SpyGlass® Built-In Rules Reference Guide

Version N-2017.12-SP2, June 2018

SYNOPSYS®

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Boost Process

Project homepage: *http://www.highscore.de/boost/process0.5/index.html*

Project license:

Boris Schaeling

Report an Error

The SpyGlass Technical Publications team welcomes your feedback and suggestions on this publication. Please provide specific feedback and, if possible, attach a snapshot. Send your feedback to *spyglass_support@synopsys.com*.

Contents

Preface

About This Book

The SpyGlass® Built-In Rules Reference Guide describes SpyGlass Built-In messages infrastructure. The document describes the types of Built-In messages, the processing stages when they are generated, and how to handle them.

Contents of This Book

The SpyGlass Built-In Rules Reference has the following chapters:

Chapter	Describes...
About SpyGlass Built-in Messages	Overview of SpyGlass Built-In Rules
Language Analysis Built-In Rules	Rules checked when a design is analyzed
Elaboration Built-In Rules	Rules checked when a design is elaborated
SpyGlass Design Constraints Built-In Rules	Rules checked on user-specified SGDC files
Constraints-Specific Built-In Rules	Rules checked for different SGDC constraints
Basic HDL Built-In Rules	Rules checked on user-specified HDL source files
Synthesis Built-In Rules	Rules checked when a design is synthesized
Liberty File Parsing Built-In Messages	Rules checked on the gate libraries
LEF Parsing Messages	Rules checked on the LEF files
DEF Parsing Messages	Rules checked on the DEF files
SDC Parsing Messages	Rules checked on SDC files
PLIB Parsing Built-In Rules	Rules checked on PLIB files
Commands and Rule Parameters Sanity Checking Rules	Rules to identify issues pertaining to user input and issues related to software configuration or versions.
SpyGlass Informational Messages	Informational messages generated
Flattening Rules	Rules reported during scenario, such as flattener API called with a NULL argument or a combinational loop detected during simulation
SDC-to-SGDC Translation Built-In Rules	Rules checked when design attributes are translated from the SDC format to the SGDC format

Chapter	Describes...
SPEF Checking Rules	Rules related to issues in an SPEF file.
SpyGlass Abstraction Flow Messages	Rules checked during the SoC flow.

Typographical Conventions

This document uses the following typographical conventions:

To indicate	Convention Used
Program code	`OUT <= IN;`
Object names	`OUT`
Variables representing objects names	*<sig-name>*
Message	Active low signal name '<sig-name>' must end with _X.
Message location	OUT <= IN;
Reworked example with message removed	OUT_X <= IN;
Important Information	**NOTE:** This rule...

The following table describes the syntax used in this document:

Syntax	Description
[ ] (Square brackets)	An optional entry
{ } (Curly braces)	An entry that can be specified once or multiple times
| (Vertical bar)	A list of choices out of which you can choose one
. . . (Horizontal ellipsis)	Other options that you can specify

About SpyGlass Built-in Messages

While analyzing or synthesizing RTL designs, SpyGlass® generates a number of standard error or warning messages known as *built-in messages*. These built-in messages are different from the rule violation messages generated during rule-checking.

There are the following classes of such built-in messages:

Message Type	Message Prefix
Syntax errors	STX_
Language Warnings	WRN_
Synthesis warnings	SYNTH_
Synthesis errors	SYNTH_
Post-elaboration syntax errors (VHDL only)	ELAB_

Syntax Errors and Language Warning messages are language-specific. See *Language Analysis Built-In Rules* for list of these messages. Synthesis warnings and errors are language-neutral for the most part. See *Synthesis Built-In Rules* for list of this message set.

The ELAB_ messages may appear when a VHDL design is being processed after elaboration. These messages relate to syntax issues and have the same number and content as the syntax messages except they have the

ELAB_ prefix.

NOTE: *Some rules in SpyGlass products are mapped to certain SpyGlass built-in messages. So when you run SpyGlass with rule-checking, some standard warning and error built-in messages are suppressed and equivalent rule violation messages of SpyGlass products are reported in their place.*
However, if a product rule overriding a warning built-in message has been disabled by the set_goal_option ignorerules <*rule-name*> *project file command, the corresponding built-in message is also not reported. But if the same product rule is disabled because of some other reason, such as it is not a mandatory rule and some other rule is specified by the* set_goal_option rules <*rule-names*> *project file command, the corresponding original built-message is reported.*

Processing HDL Designs

This section describes processing of Verilog and VHDL designs under the following topics:

- *Processing Verilog Designs*
- *Processing VHDL Designs*

Processing Verilog Designs

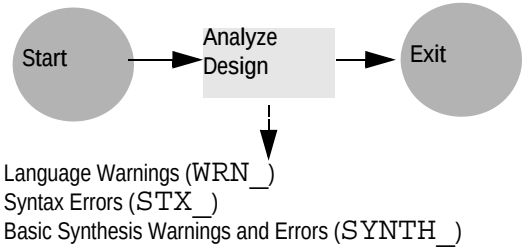
SpyGlass follows different processes while processing Verilog designs in the rule-checking mode and without the rule-checking mode.

For details, refer to the following topics:

- *Processing Verilog Designs Without Rule-Checking*
- *Processing Verilog Designs With Rule-Checking*

Processing Verilog Designs Without Rule-Checking

When you run SpyGlass *without* rule-checking (`set_goal_option norules yes`) on a Verilog design, the following process is followed:



The SpyGlass Verilog processing *without* rule-checking has the following steps:

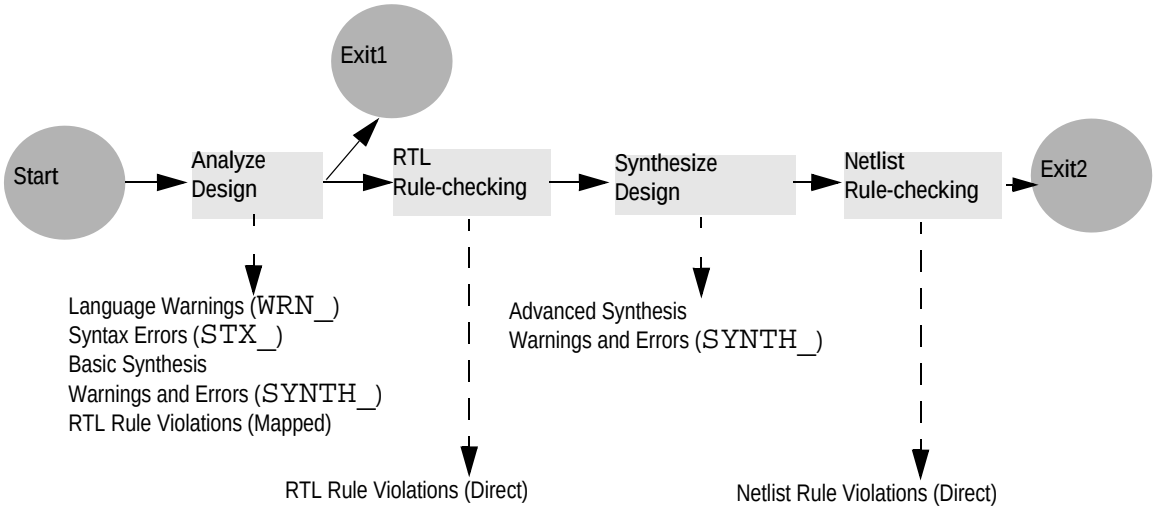
1. SpyGlass analyzes the design and generates the following types of standard built-in messages:
   - ❒ Language Warning messages (`WRN_` messages)

- ❒ Syntax Error messages (STX_ messages)
- ❒ Basic Synthesis Error and Warning messages (SYNTH_ messages)

2. SpyGlass exits.

Processing Verilog Designs With Rule-Checking

When you run SpyGlass *with* rule checking on a Verilog design, the following process is followed:



The SpyGlass Verilog processing *with* rule-checking has the following steps:

1. SpyGlass analyzes the design and generates the following types of standard built-in messages:
   - ❒ Language Warning messages (WRN_ messages)
   - ❒ Syntax Error messages (STX_ messages)
   - ❒ Basic Synthesis Error and Warning messages (SYNTH_ messages)

   SpyGlass also generates Rule Violation messages (instead of built-in messages mapped to the rules in the selected products)

2. If the design has syntax errors (STX_ messages) after initial processing, SpyGlass exits (shown as **Exit1**).
3. If the design does not have any syntax errors, SpyGlass performs RTL rule-checking and generates Rule Violation Messages as applicable.
4. SpyGlass synthesizes the design and generates the advanced synthesis error and warning messages (SYNTH_ messages).
5. SpyGlass performs Netlist rule-checking and generates Rule Violation Messages as applicable.
6. SpyGlass exits (shown as **Exit2**).

Processing VHDL Designs

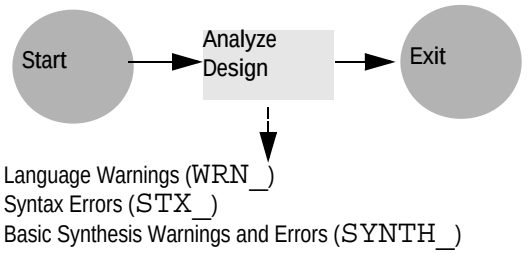
SpyGlass follows different processes while processing Verilog designs in the rule-checking mode and without the rule-checking mode.

For details, refer to the following topics:

- *Processing VHDL Designs Without Rule-Checking*
- *Processing VHDL Designs With Rule-Checking*

Processing VHDL Designs Without Rule-Checking

When you run SpyGlass *without* rule-checking (`set_goal_option norules yes`) on a VHDL design, the following process is followed:

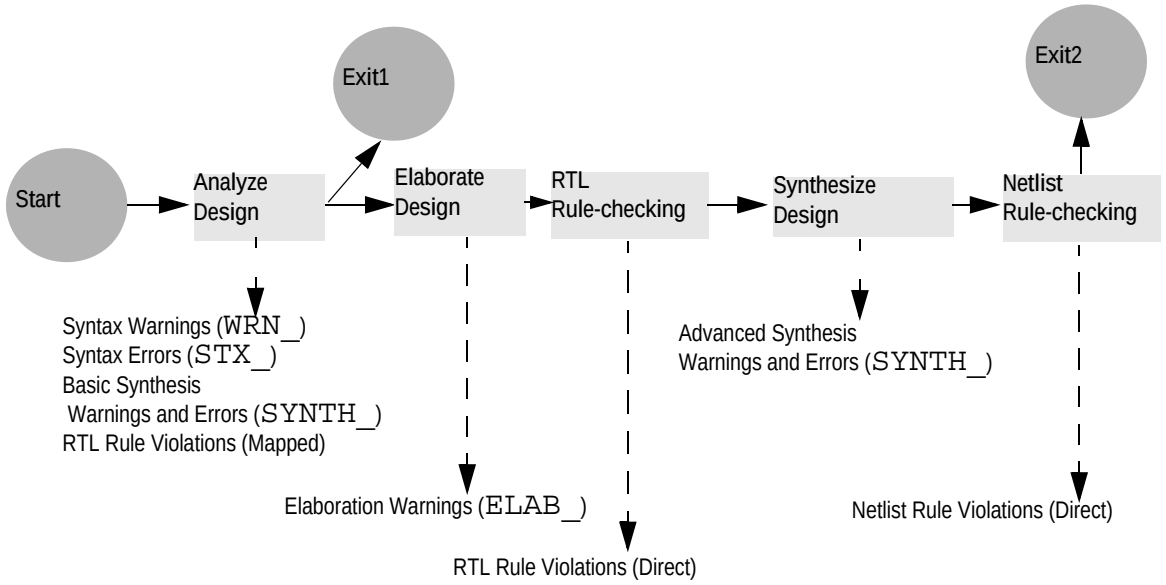


The SpyGlass VHDL processing *without* rule-checking has the following steps:

1. SpyGlass analyzes the design and generates the following types of standard built-in messages:
   - ❒ Language Warning messages (WRN_ messages)
   - ❒ Syntax Error messages (STX_ messages)
   - ❒ Basic synthesis error and warning messages (SYNTH_ messages)
2. SpyGlass exits.

Processing VHDL Designs With Rule-Checking

When you run SpyGlass *with* rule checking on a VHDL design, the following process is followed:



The SpyGlass Verilog processing *with* rule-checking has the following steps:

1. SpyGlass analyzes the design and generates the following types of standard built-in messages:
   - ❒ Syntax warning messages (WRN_ messages)

- ❒ Syntax error messages (STX_ messages)
- ❒ Basic synthesis error and warning messages (SYNTH_ messages)

SpyGlass also generates Rule Violation messages (instead of built-in messages mapped to the rules in the selected products)

2. If the design has syntax errors (STX_ messages) after initial processing, SpyGlass exits (shown as **Exit1**).
3. If the design does not have any syntax errors, SpyGlass may perform elaboration of the RTL design and generate elaboration time errors or warnings (ELAB_ messages)
4. SpyGlass performs RTL rule-checking and generates Rule Violation Messages as applicable.
5. SpyGlass synthesizes the design and generates the advanced synthesis error and warning messages (SYNTH_ messages).
6. SpyGlass performs Netlist rule-checking and generates Rule Violation Messages as applicable.
7. SpyGlass exits (shown as **Exit2**).

Handling SpyGlass Built-In Messages

You should prioritize and handle a built-in message based on its severity, such as error, language warning, synthesis warning, and synthesis error. This is described in the following topics:

- *Handling Syntax Error Messages*
- *Handling Language Warning Messages*
- *Handling Synthesis Warning Messages*
- *Handling Synthesis Error Messages*

Handling Syntax Error Messages

If you encounter a syntax error message after design analysis, you must fix that error before SpyGlass can process the design any further. Most rule checks will not be run if the design contains syntax errors.

Handling Language Warning Messages

If you encounter a language warning message, you should check that the potential problem indicated is expected and not a concern. SpyGlass will continue processing if language warnings are reported, although the nature of the analysis may be affected. For example, if you see a warning that the size of an expression does not match the size of the object to which it is assigned, you may decide that this is known but not important. On the other hand, you may realize that due to this problem, a value may be truncated or extended where no such modification was expected.

Handling Synthesis Warning Messages

If you encounter a synthesis warning message, either the specified construct is not synthesizable and therefore the design unit containing that construct cannot be synthesized, or (in a few cases) the construct will be ignored during synthesis. This condition is immediately interesting if you want to check a design for synthesizability, but it is also important to understand that SpyGlass runs most complex connectivity and functionality checks on a design by (automatically) synthesizing the design internally.

Design units which cannot be synthesized are skipped in this process and therefore will be ignored in analysis of those connectivity and functionality rules. SpyGlass will let you know which design units have been skipped for this reason, but you should be aware that analysis of those design units will necessarily be incomplete.

Handling Synthesis Error Messages

If you encounter a synthesis error message, it indicates that a design unit could not be synthesized due to an un-synthesizable construct. Also, the design unit will be replaced by a black box in the resulting netlist. Presence of such design units affects the flattening stage also and the resulting rule-checking will be inaccurate.

Language Analysis Built-In Rules

Atrenta® SpyGlass Language Built-In rules are categorized as follows:

- *VHDL Syntax Error Rules*
- *Verilog Syntax Error Rules*
- *Syntax Warning Rules*
- *Syntax Informational Rules*
- *Elaboration Rules (Analysis Stage)*
- *Synthesis Rules (Analysis Stage)*

VHDL Syntax Error Rules

Rules of this category report a violation for any syntax error in the VHDL code.

STX_VH_2

Syntax error detected.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it encountered a symbol or sequence of symbols in the VHDL source file that is illegal for the context in which it appears.

The syntax of VHDL is a set of rules defining the expected order of characters, words and constructs in a source file. If your VHDL source file does not conform to these rules, SpyGlass will generate this message along with an indication of where in the source file the error was first detected.

Message Details

```
Syntax error at or near <location>
```

Severity

Syntax

STX_VH_3

Cannot open file for writing log.

Language

VHDL

Rule Description

SpyGlass generates this syntax error when it cannot open a file for writing log.

Message Details

```
Cannot open file "<file-name>" for writing log
```

Severity

Syntax

STX_VH_4

Syntax error detected as per VHDL-87.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it detected syntax error as per VHDL-87 standards.

As per VHDL-87 standards, the syntax of VHDL is a set of rules defining the expected order of characters, words, and constructs in a source file. If your VHDL source file does not conform to these rules, SpyGlass will generate this message along with an indication of wherein the source file the error was first detected.

Message Details

```
Syntax error on "<error>" (invalid syntax as per VHDL-87)
```

Severity

Syntax

STX_VH_5

File not found.

Language

VHDL

Rule Description

SpyGlass generates this syntax error if the file is not found in its appropriate path.

Message Details

`File "<file-name>" not found`

Severity

Syntax

STX_VH_6

File could not be opened for writing.

Language

VHDL

Rule Description

SpyGlass generates this syntax error when the needed file could not be opened for writing.

Message Details

```
File "<file-name>" could not be opened for writing
```

Severity

Syntax

STX_VH_7

Label name given at the end must be the same as that given in the beginning.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of mismatch between a label that appeared at the end of a block statement and the block label.

The block label present at the beginning of the block statement is necessary. However, the label appearing at the end of the block statement is optional, and if present, must be the same as the one used at the beginning of the block

Message Details

```
End label '<error-name>' does not match the starting label
'<starting-label>' of this <block-name>
```

Severity

Syntax

STX_VH_8

Invalid operator detected.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an invalid operator.

Any operator used in your code must belong to one of following class of operators: Logical operators, Relational operators, Shift operators, Adding operators, Sign operators, Multiplying operators, and Miscellaneous operators.

Use only that operator which is defined by the language.

Message Details

```
Invalid operator <operator>
```

Severity

Syntax

STX_VH_10

Too many errors... Exiting.

Language

VHDL

Rule Description

SpyGlass generates this syntax error when it finds too many errors and has to exit.

Message Details

```
Too many errors... Exiting
```

Severity

Syntax

STX_VH_11

Identifier must be declared before usage.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not find the declaration of an identifier used in your VHDL code.

An identifier can be used only after its declaration.

For example:

```
entity TEST is
  constant a:integer:=16#54321#A1;
-- A is undeclared
-- use the correct syntax 16#54321#E1
-- where E is used as exponent
end TEST;
```

Message Details

```
Use of undeclared identifier '<identifier>'
```

Severity

Syntax

Suggested Fix

Declare an identifier before using it.

STX_VH_12

Start and terminating labels must match.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a mismatch between the start and terminating labels.

The language requires that the start and terminating labels denoting the starting and ending points of a region encapsulating a set of statements must match.

Message Details

```
Start and terminating labels do not match
```

Severity

Syntax

Suggested Fix

Make the start and terminating labels same.

STX_VH_13

Unexpected end label.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an unexpected end label.

Message Details

`Unexpected end label`

Severity

Syntax

Suggested Fix

Use legal end label in your code.

STX_VH_14

Binary expression couldn't be created.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a binary expression that couldn't be created.

Message Details

```
Cannot create binary expression
```

Severity

Syntax

STX_VH_15

Unary expression couldn't be created.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a unary expression that couldn't be created.

Message Details

```
Cannot create unary expression
```

Severity

Syntax

STX_VH_16

Unexpected end of file.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an unexpected end of file.

The syntax of VHDL is a set of rules defining the expected order of characters, words and constructs in a source file. If your VHDL source file does not conform to these rules, spyglass will generate this message along with an indication of wherein the source file the error was first detected.

For example

```
entity ent is
  port(
  in1,in2: bit_vector ( 0 to 1);
  output : out bit_vector ( 0 to 1));
end;

architecture arch of ent is
begin
  output <= in1 and in2;
--ERROR unexpected end of file,
-- it is expecting the "end;"
```

Message Details

```
Unexpected end of file
```

Severity

Syntax

Suggested Fix

End the file properly.

STX_VH_17

Generic interface list can have only constant declarations.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of object(s) of class other than the constant-class in the generic interface list.

The generic interface list is like any other interface list, but with the restriction that we can only include constant-class objects, which must be of mode IN (that is, the value can only be read).

For example:

```
entity ent is
  generic ( signal c1 : in integer ) ;
-- ERROR as signal declaration not allowed
-- only constants are allowed
end ent;
```

Message Details

```
Generic interface list can have only constant declarations
```

Severity

Syntax

Suggested Fix

Make sure that generic interface list in your program contains constant class object declarations only.

STX_VH_18

Interface declaration for a generic must have mode IN.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of mode other than IN used in an interface declaration of a generic.

The generic interface list declares constant-class objects which can only be of mode IN (that is, the value can only be read).

For example:

```
entity ent is
end ent;
architecture arch of ent is
  component A2 generic (constant G2 : out BOOLEAN);
-- ERROR: the only mode allowed in
-- a local generic list is in.
  end component;
begin
end;
```

Message Details

```
An interface declaration for a generic must have mode IN
```

Severity

Syntax

Suggested Fix

Use mode IN only for object declarations in an interface list of a generic.

STX_VH_19

Port interface list can have only signal declarations.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an object class other than the signal class used in a local port interface list. The only object class allowed in a local port interface list is the signal class.

For example:

```
package pack is
  type file_type is file of integer;
end;

use work.pack.all;
entity ent is
  port(file in1: file_type);
-- ERROR port can only be a signal
end;
```

Message Details

`Port interface list can have only signal declarations`

Severity

Syntax

Suggested Fix

Declare objects of only the signal class in port interface list.

STX_VH_20

Default expression for a LINKAGE mode interface declaration not expected.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of default expression for a LINKAGE mode interface declaration.

It is an error if a default expression appears in an interface declaration and any of the following conditions hold:

- The mode is linkage.
- The interface object is a formal signal parameter.
- The interface object is a formal variable parameter of mode other than in.
- The subtype indication of the interface declaration denotes a protected type.

Message Details

`Default expression for a LINKAGE mode interface declaration not expected`

Severity

Syntax

STX_VH_21

Parameters belonging to a function interface list must be of mode IN.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a mode other than the mode IN used in a function formal parameter list.

The parameters belonging to a function interface list can only be of signal or constant objects and the only mode allowed is IN.

For example:

```
entity ent is
  port (d : IN bit; q : OUT bit);
end ent;

architecture arch of ent is
  function func1 (signal p1 : buffer bit) return bit;
-- ERROR: parameter can only of mode IN for functions.

function func1 (signal p1 : buffer bit) return bit is
  variable v1 : bit;
    begin
      v1 := p1;
      return (v1);
    end;
begin
end;
```

Message Details

The parameters belonging to a function interface list cannot be of any mode other than IN

Severity

Syntax

Suggested Fix

Use mode IN for parameters belonging to a function interface list.

STX_VH_22

Interface list of a function can have only constant or signal parameters.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of object(s) of a class other than constant or the signal class in a function interface list.

The language allows only objects of signal and constant classes in a function interface list and the default object class is constant.

For example:

```
entity ent is
end ent;
architecture arch of ent is
  function exp_type_check (variable c1: in integer)
-- ERROR: parameter type has be either signal
-- or constant
  return boolean is
  begin
    return true;
end exp_type_check;
begin
end;
```

Message Details

```
The interface list of a function can have only constant or
signal parameters
```

Severity

Syntax

Suggested Fix

Use Constant or Signal parameters in interface list of a function.

STX_VH_24

Interface list of a procedure cannot have a parameter of BUFFER/LINKAGE port type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of parameter(s) of BUFFER/LINKAGE port types in the interface list of a procedure.

In a procedure, the interface list parameters may be constants, variables or signals and their modes may be IN, OUT, OR INOUT. If the object class of a parameter is not explicitly specified, then the Object class is by default a constant if the parameter is of mode IN, else it is a variable if the parameter is of mode OUT or INOUT.

For example:

```
entity ent is
  port ( d : in bit; q : out bit );
end ent;

architecture arch of ent is
  procedure proc1 (signal p1 : buffer bit);
-- ERROR:Parameter of BUFFER/LINKAGE port type not allowed

procedure proc1 (signal p1 : buffer bit) is
    variable v1 : bit;
begin
    v1 := p1;
end;
begin
end;
```

Message Details

```
Interface list of a procedure cannot have a parameter of
BUFFER/LINKAGE port type
```

Severity

Syntax

Suggested Fix

Do not use parameters of BUFFER/LINKAGE port type in the interface list of a procedure.

STX_VH_25

Constant parameter part of procedure interface list must be of mode IN.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find the IN mode for constant parameter(s) in interface list of a procedure.

In a procedure interface list, the parameters may be constants, variables, or signals and their modes may be IN, OUT, or INOUT. If the object class of a parameter is not explicitly specified then the Object class is by default a constant, if the parameter is of mode IN, else it is a variable if the parameter is of mode OUT or INOUT.

For example:

```
entity ent is
    port ( d : IN bit; q : OUT bit);
end ent;

architecture arch of ent is
    procedure proc1 ( constant p1 : inout bit);
--ERROR:Constant parameter part of procedure interface list
must be of mode IN
    procedure proc1 ( constant p1 : inout bit) is
      variable v1 : bit;
  begin
  end;
begin
end;
```

Message Details

```
Constant parameter part of procedure interface list must be of
mode IN
```

Severity

Syntax

STX_VH_26

Default expression cannot be specified for an interface variable declaration, which is not of mode IN.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a default expression specified for an interface variable declaration, which is not of mode IN.

For example:

```
entity ent is
    port ( d : IN  bit;
           q : OUT bit);
end ent;

architecture arch of ent is
    procedure proc1 ( p1 : inout bit:='1');
--ERROR:Constant parameter part of procedure interface list
must be of mode IN
    procedure proc1 ( p1 : inout bit:='1') is
      variable v1 : bit;
  begin
  end;
begin
end;
```

Message Details

```
Default expression cannot be specified for an interface
variable declaration, which is not of mode IN
```

Severity

Syntax

STX_VH_27

Overloaded operator function must have either 1 or 2 parameters in its interface list.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an overloaded operator function which had neither one nor two parameters in its interface list. An overloaded operator function must have either 1 or 2 parameters in its interface list depending on whether it is overloading a unary operator or a binary operator.

For example:

```
entity ent is
end ent;

architecture arch of ent is
    function "+" (a1 : integer ; b1 : integer:=
12;c1:integer) return integer is
--ERROR : Overloaded operator function "+" can have only one
or two parameter(s) in its interface list
    begin
      return 12;
    end "+
begin
end ;
```

Message Details

```
Overloaded operator function "<func-name>" must have either 1
or 2 parameters in its interface list
```

Severity

Syntax

Suggested Fix

Check the type of operator overloaded by a function and then insert the correct number of parameter(s).

STX_VH_28

Overloaded operator function for a unary operator must have exactly 1 parameter in its interface list.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an overloaded operator function for a unary operator that did not have one parameter in its interface list.

An overloaded operator function must have the same number of parameters in its interface list as the number of operands of an operator it is overloading.

For example:

```
entity ent is
end ent;

architecture arch of ent is
    function "not" (a1 : real; b1 : integer:= 12)
      return integer is
-- ERROR : Overloaded operator function "NOT" can have
-- only one parameter in its interface list
    begin
      return 12;
    end "not"
begin
end;
```

Message Details

```
Overloaded operator function "<func-name>" must have exactly 1
parameter in its interface list
```

Severity

Syntax

Suggested Fix

Check the type of operator overloaded by a function and then insert the correct number of parameter(s).

STX_VH_29

Overloaded operator function for binary operator must have exactly 2 parameters in its interface list.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an overloaded operator function for a binary operator that did not have two parameters in its interface list.

An overloaded operator function must have the same number of parameters in its interface list as the number of operands of an operator it is overloading.

For example:

```
entity ent is
end;

architecture arch of ent is
    function "ror"(in1: integer) return boolean is
-- ERROR:Overloaded operator function "ROR" must have
-- exactly 2 parameters in its interface list
    begin
      return true;
    end;
begin
end;
```

Message Details

```
Overloaded operator function "<func-name>" must have exactly 2
parameters in its interface list
```

Severity

Syntax

Suggested Fix

Check the type of operator overloaded by a function and then insert the correct number of parameter(s).

STX_VH_30

Enumeration literal must not be redeclared.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found a character literal or an identifier declared more than once in an enumeration type declaration.

Different enumeration types may include the same identifier as a literal (called overloading) so, the context of use must make it clear which type is meant. It is an error if a literal is declared more than once in same enumeration type declaration.

For example:

```
entity ent is
  port(in1,in2: integer; output : out integer);
  signal a: integer;
end;

architecture arch of ent is
  type enum is( a, b, c, d);
--ERROR "a" is already declared
begin
  output <= in1 + in2;
end;
```

Message Details

```
Enumeration literal "<literal>" already declared in file
'<file-name>' at line <line-num>
```

Severity

Syntax

Suggested Fix

See the enumeration type declaration(s) and verify that the identifiers used in any enumeration type declaration are all different.

STX_VH_31

Subprogram declaration must have corresponding body.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the absence of subprogram body of subprogram declaration.

A subprogram declaration has to have corresponding body in VHDL code.

In the following example, no function body is defined for function `func1`:

```
entity ent is
end ent;
architecture arch of ent is
   function func1 (i,l:integer) return boolean;
begin
end;
```

Message Details

```
No body defined for subprogram declaration '<declaration>'
```

Severity

Syntax

Suggested Fix

Define body for subprogram declared in your code.

STX_VH_32

Illegal type or subtype declaration detected.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it detected an illegal type or subtype declaration.

For example:

```
package pkg is
  type x is range 1 to 10;
end pkg;

use work.pkg.all;

entity ent is end;

architecture arch of ent is
  begin
    x : block
      use work.pkg.all;
      signal s : x;
-- ERROR x is declared as label of the block and
-- hence it does not denote a Type or SubType
    begin
end block;
end;
```

Message Details

```
<invalid type> does not denote a Type or Subtype declaration
```

Severity

Syntax

Suggested Fix

Use a valid identifier which shows a type or subtype declaration.

STX_VH_33

Resolution function name should be legal (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an illegal resolution function name.

Message Details

```
'<function-name>' does not denote a resolution function name
```

Severity

Syntax

Suggested Fix

Use valid resolution function name in your code.

STX_VH_34

Re-declaration of identifier is not allowed.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of multiple declarations of an identifier in your VHDL code.

For example:

```
entity ent is
end;
architecture arch of ent is
    signal first : integer;
begin
    first:block
--ERROR: label first already defined as a signal
    begin
    end block;
end;
```

Message Details

```
Identifier "<identifier-declaration>" is already declared in
file '<file-name>' at line <line-num>
```

Severity

Syntax

Suggested Fix

Use an identifier which is not used before.

STX_VH_35

SHARED variable declaration is not allowed in the declarative region of a process or a subprogram body.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of declaration of a shared variable inside a subprogram body or the process statement.

A variable that is declared outside of a process or a subprogram is called a shared variable. A shared variable can be read and updated by more than one process.

We can declare shared variables only in the places in a model where we cannot declare normal variables; namely, in entity declarations and architecture bodies, in block and generate statements and in packages.

For example:

```
entity ent is
end;
architecture arch of ent is
    impure function funcTest(constant fval : integer) return
bit is
      shared variable funcFhVar : integer := 25;
--ERROR: SHARED variable declaration not allowed here
    begin
      return('1');
    end;
begin
end;
```

Message Details

```
SHARED variable declaration is not allowed in the declarative
region of a process or a subprogram body
```

Severity

Syntax

Suggested Fix

Do not declare SHARED variables in a declarative region of a process or a subprogram body.

STX_VH_36

Declaration of only SHARED variables is allowed in present context.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of declaration of variables in region where only shared variable declaration is allowed.

A variable that is declared outside of a process or a subprogram is called a shared variable. A shared variable can be read and updated by more than one process.

For example:

```
entity ent is
end ent;

architecture arch of ent is
  variable err : boolean := true;
--ERROR :illegal location for variable declaration
-- Only shared variables are allowed here
begin
end arch;
```

Message Details

```
Only SHARED variables may be declared in this declarative
region
```

Severity

Syntax

Suggested Fix

Do not declare variables outside the process or the subprogram body.

STX_VH_37

Only predefined attribute is allowed on a range.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that only a predefined attribute (by context) is allowed on a range.

Message Details

```
Only predefined attribute '<attribute-name>' is allowed on a range
```

Severity

Syntax

STX_VH_38

Use legal object as demanded by present construct

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an illegal object in a construct. In any construct the object used must be valid as per the context in which that object appears.

For example, this error will occur if target as function call is used in variable assignment. So, the following code will produce this error:

```
function check (x : integer) return integer is

begin
return (10 * x);
end;
variable k : integer := 0;
variable p : integer := 12;
check(k) := check(p) + 24;
```

--Failure here: function call cannot be used in this construct.

Message Details

```
<object-name> cannot be used in this construct
```

Severity

Syntax

STX_VH_39

Type must be declared before usage.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of undeclared type.

Message Details

```
Undeclared Type '<type-name>'
```

Severity

Syntax

Suggested Fix

Declare type before usage.

STX_VH_40

Re-declaration of subprogram is not allowed.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of multiple declarations of a subprogram in your VHDL file.

If you are overloading a subprogram then make sure that the number of parameters and/or their types must differ. However, if you are not overloading a subprogram then use only one declaration of subprogram in your code.

For example:

```
entity ent is
end;

architecture arch of ent is

    function f2(in1:bit) return bit;
    function f2(in1:bit) return bit;
 --ERROR sub program f2 is already declared
    function f2(in1:bit) return bit is
  begin
    return '1';
  end function;
begin
end;
```

Message Details

```
Subprogram "<program-name>" is already declared in file
'<file-name>' at line <line-num>
```

Severity

Syntax

STX_VH_41

Return type must be declared.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find the return type of a function.

Unlike a procedure subprogram, a function calculates and returns a result that can be used in an expression. Hence the function declaration must specify the type of the result it is going to return.

For example:

```
entity e is
end e;
architecture a of e is
  function Q(X :integer := 2) return X is
-- Failure here: Function return type must be of
-- predefined type.
  begin
    end Q;
  begin
end;
```

Message Details

```
Undeclared return type '<return-type>'
```

Severity

Syntax

STX_VH_42

Procedure name must not denote a predefined operator.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an illegal procedure name.

The VHDL language does not allow the usage of an operator or operator name as a procedure designator.

For example:

```
entity ent is
end ent;
architecture arch of ent is
    procedure "+" (x: in integer; y: out boolean);
--ERROR "+" is a predefined operator
-- Therefore cannot be used as procedure name
begin
end arch;
```

Message Details

"<procedure-name>>" denotes a pre-defined operator: hence it cannot be used as a procedure name

Severity

Syntax

Suggested Fix

Check that no operator or operator name has been used as a procedure designator.

STX_VH_43

Overloaded operator function must have a parameter interface list.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not find the parameter interface list of an overloaded operator function. An overloaded operator function must have parameter interface list and at least one parameter must be present in that interface list.

Suggested Fix

Write the parameter interface list with the appropriate number of parameter(s) for overloaded operator function.

Message Details

```
Overloaded operator function must have a parameter interface
list
```

Severity

Syntax

STX_VH_44

Redefining of subprogram body is not allowed.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of multiple bodies of a subprogram in your VHDL file.

If you are overloading a subprogram then make sure that the number of parameters and/or their types in their respective parameter lists must differ because that is the only way spyglass would be able to resolve the ambiguity of calling one of them when that subprogram is called.

For example:

```
entity ent is
end;

architecture arch of ent is
    subtype int is integer range 0 to 10000;
  procedure proc (in1,in2:integer) is
    begin
    end;
  procedure proc (in1,in2:int) is
 --ERROR procedure proc is already defined above
  begin
  end;
begin
end;
```

Suggested Fix

Define only one body for a subprogram if you are not overloading it and in case of overloading, make their parameter list different by either changing the types of parameters or their numbers or both.

Message Details

```
Body of subprogram "<sub-program>" has already been defined in
file '<file-name>' at line <line-num>
```

Severity

Syntax

STX_VH_45

Undefined operator symbol detected (possibly typing mistake).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an undefined operator symbol in your VHDL code.

The language requires that the operator symbol used in VHDL code must represent an operator belonging to one of the six classes of operators-a logical operator, a relational operator, an adding operator, a sign operator, a multiplying operator, and a miscellaneous operator.

So any operator symbol in VHDL code must represent predefined operators in the language. Possibly you might have made a typing mistake while writing the operator symbol.

For example:

```
entity ent is
end ent;

architecture arch of ent is
    function "ans" return BOOLEAN is
--ERROR "ans" is not a operator symbol
--correct usage remove Quotes
    variable abc:boolean;
  begin
    return abc;
  end;
begin
end arch;
```

Suggested Fix

Do not use any undefined operator symbol in your VHDL code.

Message Details

'<undefined-operator>' does not denote an operator symbol

Severity

Syntax

STX_VH_46

Re-declaration of a character enumeration literal is not allowed.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a character literal declared more than once in an enumeration type declaration.

Different enumeration types may include the same identifier as a literal (called overloading) so, the context of use must make it clear which type is meant. But it is an error if a literal is declared more than once in the same enumeration type declaration.

For example:

```
package pack is
    type enum1 is ('a','b','c');
    type enum2 is ('d','e','d');
--ERROR enumeration literal 'd' is declared twice
end;
```

Suggested Fix

See the enumeration type declaration(s) and verify that the literals declared in any enumeration type are all different.

Message Details

```
Character enumeration literal '<literal>' has been declared previously
```

Severity

Syntax

STX_VH_47

Two Overloaded Subprograms are valid in this Context.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of two subprograms having same names, same number and type of parameters in their parameter list visible within the same region.

In such a case, a subprogram call may be ambiguous, and hence an error, if it is not possible to determine which of the overloaded subprograms is being called.

For example:

```
entity ent is
end;

architecture arch of ent is
    procedure proc(in1: integer) is begin end;
begin
    process
     procedure proc(in2: integer;in1: integer:=10) is begin
end;
    begin
      wait;
      proc(10);
--ERROR: Two overloaded procedures are visible here
    end process;
end;
```

Suggested Fix

Do not make ambiguous subprograms visible within the same region.

Message Details

```
Two Overloaded Subprograms are Valid in this Context at line
no= '<variable-type1>::<file-name1>' and at line no=
'<variable-type2>::<file-name2>'
```

Severity

Syntax

STX_VH_48

Either Type mismatch detected or no visible function found.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it either detected a type mismatch or did not find any visible function.

For example:

```
 entity ent is
 end ent;

 architecture arch of ent is
     begin
       test_1: process
     begin
    (0, 0, 0) := (0, 0, 0);
-- Failure here: Target of a variable assignment
--can only be a name or an aggregate.
    end process;
end arch;
```

Message Details

```
Either Type mismatch or no visible function for this case
```

Severity

Syntax

STX_VH_49

Type name should be legal (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an illegal type name. Any identifier that is used as a type name must denote a type.

For example:

```
entity ent is
end;

architecture arch of ent is
    type rec is
    record
      f1,f2: bit;
    end record;
    signal sig: rec;
begin
    sig <= arch.sig('1','0');
-- Sig is not a type

end;
```

Suggested Fix

Use valid type name in your code.

Message Details

```
'<name>' is not a type name
```

Severity

Syntax

STX_VH_50

Expression type must match type required in present context.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the mismatch between the expression type and type as is required in the context.

The result of the evaluation of an expression in a context must return a value whose type should match the type as is required in that context.

For example:

```
type A_REC is record
    E : integer;
end record;
type B_REC is record
    E : integer;
end record;
    function F2 ( PARAM : A_REC ) return B_REC is
begin
    return B_REC'(PARAM);
-- SEMANTIC ERROR: type of expression does not match
-- type mark.
end F2;
```

Message Details

```
Expression type "<type1>" does not match type of "<type2>"
required in this context
```

Severity

Syntax

STX_VH_51

Selected name should be valid (by context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the usage of invalid selected name.

Possible scenarios:

(a) Invalid use clause

(b) Invalid expression type

(c) The expression should be of type simplename

(d) The last expression should be a simple name

(e) The expression should have a name/simplename

(f) Unit type is ENTITY but entity name not matching

(g) Selected Name does not correspond to the USE clause

(h) The third expression can be ALL only for the USE clause

(i) For the USE clause the maximum number of elements in a selected name can be 3

(j) If the expression is not the ALL clause, access type must point to a record type

For example:

(1)

```
package pkg is
    type MY_TYPE is range 1 to 2;
end pkg;
    use work.pkg.all;
entity ee is
  port(sig1 : in bit);
    end ee;

architecture arch of ee is
```

```
      subtype my_sub is pkg.unknown;
-- ERROR (Selected Name does not correspond to the USE
clause)
begin
end arch;


(2)
p1 : process
    type my_record is
      record
    x1,x2 : integer;
    y : boolean;
    end record;
  variable rec1,rec2 : my_record;
    begin
  rec1.x1 := 5;
    rec1.x2 := 10;
    rec1.y  := true;
      wait for 1 ns;
    rec2 := rec1.all;
--ERROR (Selected Name does not correspond to the USE clause)
     wait for 1 ns;
end process p1;
```

Message Details

```
Invalid selected name (<name>)
```

Severity

```
Syntax
```

STX_VH_52

RECORD type object expected.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an object used as record type object but not declared as an object of that type.

For example, this error will be flagged when a selected name is used to access an element of a record and the prefix of that selected name does not denote an object of that record type.

For example:

```
test_1: process
     type R1 is record
       RE1: BOOLEAN;
     end record;
     type R2 is record
       RE2: BOOLEAN;
     end record;
     type ONE is range 1 to 1;
     type A1 is array (ONE) of BOOLEAN;
       variable V1: R1 ;
       variable V2: R2 ;
       variable V5: A1 ;
       variable V10: BOOLEAN;
     begin
     V10 := V5.ONE;
-- ERROR: NO SUCH RECORD ELEMENT;
end process;
```

Suggested Fix

Use record type object where it is expected.

Message Details

```
Object "<object>" is not a RECORD type object. Object
declaration is in file '<file-name>' at line <line-num>
```

Severity

Syntax

STX_VH_53

Element of RECORD type object expected.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an object used as an element of a record but not present in the declaration of that record type.

For example, this error will be flagged when a selected name is used to access an element of a record but that element name is not present when that record type is declared. When the selected names are used for accessing elements of a record type object, then the suffix must denote an element declared in a record type object denoted by prefix.

For example:

```
test_1: process
    type R1 is record
      RE1: BOOLEAN;
    end record;
    variable V1: R1 ;
    variable V10: BOOLEAN;
  begin
    V10 := V1.RE2;
-- ERROR: NO SUCH RECORD ELEMENT;
end process;
```

Suggested Fix

Use only those as elements of record object whose names figure in record type declaration.

Message Details

```
Object "<object>" is not an element of RECORD type "<type>"
defined in file '<file-name>' at line <line-num>
```

Severity

Syntax

STX_VH_54

Use of label outside its scope is not allowed.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the usage of a label outside its scope.

The scope of a label ends with the end of the body of the construct in which it is defined. A label indicates the start of a region in which a set of statements are enclosed and when that region ends, it is illegal to use a label associated with that region afterwards.

Suggested Fix

Do not use label outside its scope.

Message Details

```
'<label>' does not denote a Label in the current scope
```

Severity

Syntax

STX_VH_55

RHS Expression type should match target type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the mismatch between the types of RHS expression and target object.

For example:

```
entity ent is
end ent;

architecture arch of ent is
    begin
      test_1: process
      variable k : integer := true;
--Failure here:Type mismatch in variable declaration.
    begin
    end process test_1;
end arch;
```

Suggested Fix

Use RHS expression and target of same types in VHDL code.

Message Details

```
RHS Expression type does not match target type "<type>"
```

Severity

Syntax

STX_VH_56

Floating point value is not supported.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that a floating point value is not supported.

Message Details

```
Floating point value is not supported
```

Severity

Syntax

STX_VH_57

Base must be in range 2 to 16.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of invalid base for a based literal.

In a based literal, the base can be any value between 2 and 16. The syntax for a based literal is

```
Base#based-value#                --form 1
Base#based-value#E exponent      --form 2
```

SpyGlass will flag an error if base in based literal is either less than '2' or greater than '16'.Also the base and the exponent values in a based literal must be in decimal notation.

For example:

```
(a)
entity Eis is
    constant m: integer := 17#F09#;
-- ERROR
end Eis;
(b)
test_1: process
    variable I : INTEGER;
      begin
    I := 0#121#E2;
-- ERROR
end process;
```

Suggested Fix

Use base in the range 2 to 16.

Message Details

```
Base "<invalid-base>" must be in range 2 to 16
```

Severity

Syntax

STX_VH_58

Attribute name must be legal (in context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an identifier used as an attribute name but not declared as an attribute.

Any identifier that is used as an attribute name must first be declared in an attribute declaration which is of the form:

```
attribute <attribute-name> : <value-type>;
```

Now you can use an attribute name in an attribute specification which is used to associate a user-defined attribute with a name and to assign a value to the attribute. The syntax for an attribute specification is:

```
attribute <attribute-name of items-names> : name-class is expression;
```

In case of predefined attributes, the attribute must belong to one of five classes of predefined attributes - Value attribute, Function attribute, Signal attribute, Type attribute and Range attribute.

Message Details

```
'<identifier>' is not an attribute name
```

Severity

Syntax

STX_VH_59

Null literal or aggregate or string literal is not allowed as an operand of type conversion.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of invalid operands of type conversion. Operands of type conversion cannot be the literal null, an allocator, an aggregate, or a string literal. Type conversions are allowed between closely related types and type of the operand of a type conversion must be determinable independent of the context.

For example:

```
test_1: process
    subtype Grapes is STRING;
      constant Green : Grapes := Grapes ("CLSI");
-- ERROR
begin
end process;
```

Suggested Fix

Use correct operand of type conversion.

Message Details

`Null literal or aggregate or string literal "<literal>" is not allowed as an operand of Type Conversion`

Severity

Syntax

STX_VH_60

Type of the operand of type conversion must be determinable independent of the context.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it cannot determine type of operand of type conversion independent of the context in which it appeared.

The type conversion statement has the following form-:

```
type_conversion ::= type_mark(expression);
```

An expression enclosed by parentheses is allowed as an operand of type conversion only if the expression alone is allowed. Furthermore the type of operand of type conversion must be determinable independent of the context (in particular, independent of the target type).

For example:

```
test_1: process
    type Grapes is (Sweet, Sour);
    type Oranges is (Sweet, Bitter);
      variable Green : Grapes;
      variable Seville, valencia : Oranges;
begin
  := Grapes (Sweet);
 -- ERROR
end process;
```

Message Details

```
The type of the operand "<operand-type>" of type conversion
must be determinable independent of the context
```

Severity

Syntax

STX_VH_61

Type conversion operand and the result type must be closely related types.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the mismatch between the types of operand and target in type conversion. A type conversion provides for explicit conversion between closely related types. These include those between integer and real, between array types that have the same dimensions and whose index types are closely related and those whose elements types are the same.

For example:

(1)

```
type SIGNED is array(NATURAL range <>)of BIT;
type BIT_VECTOR is array (NATURAL range <>)of BIT;
    signal FCR:SIGNED(0 to 7);
    signal EMA:BIT_VECTOR(0 to 7);
```

Without type conversions, `FCR<=EMA;` is illegal, since signals FCR and EMA are of different types. However, using type conversion, `FCR<=SIGNED(EMA);` makes the assignment legal. Type conversion is allowed in this case since types SIGNED and BIT_VECTOR are closely related, that is, they are both vectors of the same element type, and their index types are closely related.

(2)

```
test_1: process
     type century is array (1 to 1000) of real ;
     type millenia is array (1 to 100 ) of real ;
      variable hundreds : century  ;
      variable thousand : millenia ;
begin
    thousand := millenia (hundreds);
-- Failure_here : dimensionality not same.
```

```
end process;
```

Message Details

```
Operand "<operand>" of type conversion cannot be converted to
type '<type>'
```

Severity

Syntax

STX_VH_62

Prefix of a slice name must denote a one-dimensional array.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of prefix of a slice name that is not a one-dimensional array object. A slice name is defined as:

```
prefix ( discrete_range )
```

The prefix of a slice name cannot be an array element unless the array element is a one-dimensional array.

For example:

(1)

```
test_1: process
  type BIT_VECTOR is range 1 to 10;
  variable NUM1 : BIT_VECTOR;
begin
  NUM1(0 to 1) := 0;
end process;
```

(2)

```
test_1: process
  type FIVE is range 1 to 5;
  type A1B is array (FIVE range <>) of BOOLEAN;
  subtype A1 is A1B(FIVE);
  type A2B is array(FIVE range<>, FIVE range<>) of A1;
  subtype A2    is A2B(FIVE, FIVE);
  function G return A2 is
    begin
      return (others => (others => (others => false)));
  end G;
  variable V1: A1;
  begin
    V1(2 to 4) := G(3 to 5);
```

```
end process;
```

Suggested Fix

Change the prefix of the slice so that it is appropriate for a one-dimensional array object.

Message Details

```
Slice name prefix "<prefix>" does not denote a one-dimensional array
```

Severity

Syntax

STX_VH_63

Prefix of an indexed name must be appropriate for an array type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error when it tries to interpret a VHDL fragment of the form "prefix (optional_association_list)".

In that context, the prefix should be an array. But SpyGlass was unable to find a valid interpretation for that name as an array. Check that the prefix indicated by the error message is an array name, correct the VHDL code, and re-analyze the file(s).

For example:

```
test_1: process
    variable k : integer := 0;
  begin
      if k(1) = 1 then
 -- ERROR
      NULL;
    end if;
      end process;
```

Message Details

```
Index name prefix "<prefix>" does not denote an array
```

Severity

Syntax

STX_VH_64

Prefix of an index name must denote an array of appropriate dimension (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error when it tries to interpret a VHDL fragment of the form "prefix (optional_association_list)".

In that context, the prefix should be an array of appropriate dimension by context. But SpyGlass was unable to find a valid interpretation for that name as an array of appropriate dimension. Check that the prefix indicated by the error message is an array name of appropriate dimension, correct the VHDL code, and re-analyze the file(s).

For example:

```
test_1: process
    type THREE    is range 1 to 3;
    type A21    is array (THREE, THREE) of BOOLEAN;
      variable V1   : BOOLEAN;
      variable V21: A21 ;
  begin
    V1 := V21(2);
-- ERROR
end process;
```

Message Details

```
Index name prefix "<prefix>" does not denote a
<num>-dimensional array
```

Severity

Syntax

STX_VH_65

Use valid aggregate value (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an aggregate value used in a context where no aggregate value is valid.

An aggregate is a basic operation that combines one or more values into a composite value of a record or array type.

Message Details

```
No aggregate value is valid in this context
```

Severity

Syntax

STX_VH_66

Illegal usage of an index detected at a position of index name.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of illegal usage of index at a position of index name.

An index used to access an element of an array must conform to the range constraint and dimension of array declared in array declaration.

For example:

```
type sting is array (1 to 5, 1 to 5) of character;
    variable str : sting;
    str(1 to 3, 1 to 3) := str(3 to 5, 3 to 5);
-- slice of a two dimensional array is illegal.
```

Message Details

```
Invalid index used at position <position> of index name
```

Severity

Syntax

STX_VH_67

Illegal lvalue found-cannot update the target specified.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of illegal value and hence it could not update the target specified.

If the target of a variable assignment statement is a name, then the name must denote a variable, and the base type of the expression on the right-hand side must be the same as the base type of the variable denoted by that name.

For example:

```
entity ent is
end entity;

architecture arch of ent is
    begin
      test_1 : process
    variable v1 : integer := 0;
    begin
      arch := v1;
 -- Failure here : Target of a variable assignment cannot be
the name of an architecture body.
    end process test_1;
end arch;
```

Message Details

```
Invalid lvalue: cannot update the target specified in this
assignment
```

Severity

Syntax

STX_VH_68

Target of a variable assignment, must denote a variable.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a target object in variable assignment statement that does not belong to the variable class. If the target of the variable assignment statement is a name, then the name must denote a variable, and the base type of the expression on the right-hand side must be the same as the base type of the variable denoted by that name.

For example:

```
architecture arch of ent is
    signal s : integer := 0;
begin
    test_1: process
    variable v : integer := 12;
begin
   s := v;
   -- ERROR
  end process;
  end architecture;
```

Suggested Fix

Make sure that target of variable assignment statement must belong to variable class only.

Message Details

`The target of a variable assignment, must denote a variable`

Severity

Syntax

STX_VH_69

Target of a signal assignment, must denote a signal.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of illegal target for a signal assignment statement.

If the target of the signal assignment statement is a name, the name must denote a signal (that is, must belong to signal class), and the base type of the value component of each transaction produced by a waveform element on the right-hand side must be the same as the base type of the signal denoted by that name.

For example:

```
architecture arch of ent is
    signal s : integer := 0;
begin
    test_1: process
    variable v : integer := 12;
  begin
    v <= s;
-- ERROR
  end process;
end architecture;
```

Suggested Fix

Use object of only signal class as target of signal assignment statement.

Message Details

`The target of a signal assignment, must denote a signal`

Severity

Syntax

STX_VH_70

A port of mode 'IN' cannot be assigned any value.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of IN port as target of the signal assignment.

In the signal assignment statement, the target must be either name or an aggregate. A port (or component or subcomponent of a port or an alias of any of these) whose mode is 'IN' or 'LINKAGE' cannot be on the left-hand side of signal assignment statement.

For example:

```
entity EE is
    port (sig1 : in bit);
end entity;

architecture arch of ent is
    signal sig2 : bit := '1';
begin
    sig1 <= sig2;
-- ERROR
end architecture;
```

Message Details

```
Cannot assign to IN port <port-name>
```

Severity

Syntax

STX_VH_71

Ports of mode OUT or LINKAGE cannot be read.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it cannot read an output port or a linkage port.

Typically, an element is read if used on the right-hand side of an assignment or if used in port association as an actual for formal port of mode IN/INOUT. If the interface object is of mode OUT, the value of the interface object can be updated. You can read attributes of the interface object, unless the attributes are predefined, but no other reading is allowed.

For example:

```
entity EE is
    port (sig1 : LINKAGE BIT);
end entity;

architecture arch of ent is
     signal sig2 : bit := '1';
begin
    sig2 <= sig1;
-- ERROR
end architecture;
```

Message Details

```
Cannot read output/linkage port <port-name>
```

Severity

Syntax

STX_VH_72

Non-resolved signal cannot have multiple sources.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a signal having multiple sources but not resolved.

The language requires that a signal with more than one driver must have a resolution function associated with it, otherwise it is an error. A resolution function is associated with a signal by specifying its name in the signal declaration.

For example:

(a)

```
   signal BUSY:WIRED_OR BIT;
   --is one way of associating the resolution function
WIRED_OR with the signal BUSY.
```

(b)

```
entity ent is
end;

architecture arch of ent is
  type rec is
    record
      f1,f2:integer;
  end record;
    signal sig:rec;
begin
    sig.f1<= 10;
    sig.f2<= 10;

-- ERROR
end;
```

Suggested Fix

Use resolution function for a signal having multiple sources.

Message Details

```
Unresolved signal <signal> cannot have multiple sources
```

Severity

Syntax

STX_VH_73

Character of string must be defined as a character literal for the corresponding enumerated type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a character (in string) which is not defined as a character literal for the corresponding enumerated type.

VHDL allows only those identifiers and character literals to be used as enumeration literals which are listed in corresponding enumeration type definition. Using any identifier as an enumeration literal that is not present in the list of identifiers and character literals in the corresponding enumeration type definition is illegal.

For example:

```
type arr is array (1 to 5) of bit;
    constant C1 : arr := "00_1_11
 -- ERROR
```

Message Details

```
Character '<character>' of string "<string>" is not defined as
a character literal for the enumerated type <type>
```

Severity

Syntax

STX_VH_74

Positional association cannot be used after first named association.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of positional association used after a named association in an association list. Named associations can be given in any order, but if both positional and named associations appear in the same association list, then all positional associations must occur first at their normal position. Hence once a named association is used, the rest of the association list must use only named associations.

For example:

```
entity ee is
  port ( signal a    : IN bit;
    signal b    : IN integer;
    signal c    : IN boolean;
    signal d    : IN time;
    signal e    : IN real;
    signal oint    : INOUT integer);
end ee;

architecture arch of ee is
      function funct1(   fpar1:bit   :='1';
        fpar2:integer   :=455;
        fpar3:boolean   :=true;
        fpar4:time   :=55.77 ns;
        fpar5:real   :=34.558) return integer is
begin
    return 1;
    end funct1;
begin
    test_1: process
begin
```

```
    wait for 1 ns;
    oint    <= funct1(fpar3=>c,fpar2=>b,fpar1=>a,d,e); --
ERROR at position 4
end process;
end architecture;
```

Suggested Fix

Do not use positional association once you have started using named association in an association list.

Message Details

```
Association <positional-association> cannot be positional,
because of a previous named association
```

Severity

Syntax

STX_VH_75

Procedure name must be valid (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an invalid procedure name.

Use an identifier as a procedure name which appears in your procedure definition.

For example:

```
entity ee is
end ee;
architecture arch of ee is
    signal some    : integer := 12;
begin
      test_1: process
      procedure check (x : in  integer; y : out boolean) is
  begin
    if x = 1 then
      y := true;
    else
      y := false;
    end if;
  end;
    variable p : integer := 3;
    variable q : boolean := true;


  begin
      some (p,q);
 -- ERROR
     end process;
 end architecture;
```

Message Details

<proc-name> does not denote a Procedure

Severity

Syntax

STX_VH_76

Element of record expected in element association.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the identifier used in element association does not denote an element of the record. If the type of an aggregate is a record type, the element names given as choices must denote elements of that record type. When record aggregate is evaluated, a check is made that the value of each element of the aggregate belongs to the subtype of this element. It is an error if this check fails.

For example:

```
  test_1: process
     type rec is record
       ele_1 : integer;
      ele_2 : real;
      ele_3 : boolean;
      ele_4 : integer;
    end record;
      variable p : rec :=
        (ele_3 => true,
        ele_1 => 1,
        ele_2 => 3.4,
        ele_5 => 12);
-- ERROR
end process;
```

Suggested Fix

Use valid record element name in element association of a record aggregate.

Message Details

```
<identifier> used in element association does not denote an
element of the record '<record>'
```

Severity

Syntax

STX_VH_77

Only the record field name is allowed as the selector of an element association.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of element associations by simple expressions in record aggregates.

Unlike array aggregates, we cannot use a range of values to identify elements in a record aggregate, since the elements are identified by name, not indexed by a discrete range. Only the record field name is allowed as the selector of an element association in record aggregates.

For example:

```
test_test_1: process
 type rec_type is record
   ele_1 : integer;
   ele_2 : integer;
 end record;
     variable v20 : rec_type;
   begin
   v20 := (1 + 1 => 20, ele_2 => 0);
-- ERROR
end process;
```

Suggested Fix

Use record field name to select element of a record in element association.

Message Details

```
Only the record field name is allowed as the selector of an
element association
```

Severity

Syntax

STX_VH_78

Elements of record must be used only once.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an element (part of composite type) used more than once when initializing an object of composite type (of which that element is a part). It is an error if you are covering an element of object of composite type (arrays or record) more than once in an aggregate during initialization of composite type data objects.

For example:

```
package pack is
  type rec1 is
record
    f1: integer;
    f2: bit;
end record;
  type rec2 is
record
    f1: integer;
    f2: bit;
end record;
  type rec3 is
record
    f1: rec1;
    f2: rec2;
end record;
constant const: rec3 := (f1=>(10,'1'),f1=>(10,'1'));
-- ERROR (f1 used more than once)
end pack;
```

Suggested Fix

Use each element of record only once in a record aggregate.

Message Details

```
Record element '<record-element>' has been used more than once
```

Severity

Syntax

STX_VH_79

Prefix of the pre-defined attribute (by context) must denote an array.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a predefined attribute whose prefix does not denote an array. The following predefined attributes require a prefix that is appropriate for an array object, or an alias thereof, or that denotes a constrained array subtype.

(Here A denotes any prefix that is appropriate for an array object, or an alias thereof, or that denotes a constrained array subtype and N denotes the index range).

A'LEFT[(N)]

A'RIGHT[(N)]

A'HIGH[(N)]

A'LOW[(N)]

A'RANGE[(N)]

A'REVERSE_RANGE[(N)]

A'LENGTH[(N)]

A'ASCENDING[(N)]

Suggested Fix

Use array as a prefix if you are using any one of above predefined attributes.

Message Details

`The prefix of the pre-defined attribute <attribute> must denote an array`

Severity

Syntax

STX_VH_80

Prefix of the pre-defined attribute (by context) must denote a signal.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a predefined attribute whose prefix does not denote a signal. The following predefined attributes require signal name as a prefix:

(Here S denotes a static signal name and T denotes units of TIME)

```
S'DELAYED[(T)]

S'STABLE[(T)]

S'QUIET[(T)]

S'TRANSACTION[(T)]

S'EVENT[(T)]

S'ACTIVE[(T)]

S'LAST_EVENT

S'LAST_ACTIVE

S'LAST_VALUE

S'DRIVING

S'DRIVING_VALUE
```

Suggested Fix

Use signal name as a prefix if you are using any one of above predefined attributes.

Message Details

```
The prefix of the pre-defined attribute <attribute> must denote
```

a signal

Severity

Syntax

STX_VH_81

Label used to denote a component instance must first be declared as an instance of that component.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a label used to denote a component instance that did not appear in the corresponding component instantiation statement(s).

This error might occur if you are using a label to denote a component instance in configuration declaration for late binding of that component, but that label does not appear in component instantiation statement(s) in the architecture body for which that configuration declaration is written.

For example:

```
entity ee is
end ee;

architecture aa of ee is
    component child2 is
    end component;
begin
    c1 : child2;
end aa;
configuration conf of ee is
    for aa
    for c2: child2 use entity WORK.child;
-- ERROR
    end for;
    end for;
end conf;
```

Message Details

```
Label <label-name> does not denote a component instance
```

Severity

Syntax

STX_VH_82

Deferred constant can appear only in a package declaration.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of constant declaration without providing an initialization expression. VHDL allows the declaration of deferred constants only in a package declaration and actual value is specified in corresponding package body.

For example:

```
test_1: process
          constant x: bit;
-- ERROR
 begin
 end process;
```

Suggested Fix

Initialize a constant if you are declaring it in a design unit other than package body.

Message Details

`Deferred constant <constant> may appear only in a package declaration`

Severity

Syntax

STX_VH_83

Guarded signal used must have a resolution function.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of declaration of guarded signal without providing resolution function for it.

In VHDL, a general form of guarded signal (declared to be of a register or bus kind) declaration is-:

```
signal list-of-signals:resolution function signal-type

signal-kind[:=expression];
```

The language requires that a guarded signal must be resolved signal, that is, it must have a resolution function associated with it.

For example:

```
entity ee is
end ee;

architecture arch of ee is
    signal s1 : bit bus;
-- ERROR : a guarded signal, but is not a resolved signal.
    signal s2 : bit register;

-- ERROR : a guarded signal, but is not a resolved signal.
begin
end architecture;
```

Suggested Fix

Provide resolution function for guarded signal in its declaration.

Message Details

```
No resolution function exists for the guarded signal
```

<signal-name>

Severity

Syntax

STX_VH_85

Locally static range constraint expected.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a range constraint which is not locally static in the context in which it is used.

A locally static range is either a range whose bounds are locally static expressions, or a range whose prefix denotes either a locally static subtype or an object that is of a locally static subtype. A locally static range constraint is a range constraint whose range is locally static. For example, there might be ambiguous reference to a range constraint as you may be using two or more package declarations each declaring that range constraint (with same name).

For example:

```
entity ee is
end ee;

architecture arch of ee is
    type a is range (1+1) to (10.0 + 20.0);
-- ERROR
begin
end architecture;
```

Suggested Fix

Use locally static range constraint as required in this context.

Message Details

`The range constraint must be locally static in this context`

Severity

Syntax

STX_VH_86

Constrained array type declaration expected (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of unconstrained array type declaration.

This error occurs for example, when array elements are declared as unconstrained array. Hence the following code will produce this error:

```
type I1 is range 1 to 1;
type A1 is array (I1 range <>) of BOOLEAN;
type A2 is array (I1'(1) to I1'(1)) of A1;

--Failure here:Array element cannot be an
--unconstrained array.
```

Suggested Fix

Use constrained array type declaration.

Message Details

```
Unconstrained array type is not allowed in <declaration>
```

Severity

Syntax

STX_VH_87

Return statement is allowed only within a subprogram body.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of the return statement outside a subprogram body.

The return statement, which is a sequential statement, is a special statement that is allowed only within subprograms (Functions or Procedures) and it applies to the innermost enclosing function or procedure.

For example:

```
test_1: process
    variable k : integer := 0;
    variable j : boolean := true;
begin
    if j then return k;
-- ERROR
end process;
```

Suggested Fix

Make sure that return statement is present only within the body of a function or procedure.

Message Details

`Return statement is allowed only within a subprogram body`

Severity

Syntax

STX_VH_88

Return statement in a procedure body cannot have a return value.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a return statement having a return value in procedure body.

VHDL does not allow the use of an expression in any return statement present within a procedure body. For procedures, objects of mode OUT and INOUT return their values to the calling program.

For example:

```
test_1: process
  variable i : integer := 0;
    procedure return_exp_check is
  begin
    i := 10;
      return i;
 -- ERROR
  end;
begin
    return_exp_check;
end process;
```

Suggested Fix

Make sure that you are not using an expression in a return statement in a procedure body.

Message Details

`Return statement in a procedure body cannot have a return value.`

Severity

Syntax

STX_VH_89

Return statement in a function body must have a return value.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the absence of the return statement with return value in the function body.

VHDL allows a function to return only one value and hence requires that all functions must have a return statement with an expression and the value of that expression is returned to the calling program

For example:

```
test_1: process
  function return_exp_check return integer is
    variable k : integer := 0;
  begin
    k := 10;
    return;
  end;
    variable i : integer := 0;
  begin
    i := return_exp_check;
end process;
```

Suggested Fix

Make sure that a return statement in a function body contains an expression.

Message Details

`Return statement in a function body must have a return value`

Severity

Syntax

STX_VH_90

Next/Exit statement is only allowed within a loop statement.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of EXIT/ NEXT statement outside a loop statement.

The exit statement is a sequential statement that can be used only inside a loop. It causes execution to jump out of the innermost enclosing loop (whether labeled or not) or the loop whose label is specified. The Next statement is also a sequential statement that can be used only inside a loop. The next statement results in skipping the remaining statements in the current iteration of the specified loop; execution resumes with the first statement in the next iteration of this loop, if one exists. If no loop label is specified, the innermost loop is assumed.

For example:

(a)

```
test_1: process
begin
    L : for i in 1 to 10 loop
end loop;
    next L;
end process;
```

(b)

```
test_1: process
    begin
    for i in 1 to 10 loop
      end loop;
    exit;
end process;
```

Suggested Fix

Use Exit/Next statements inside a loop only.

Message Details

<exit | next> statement is only allowed within a loop statement

Severity

Syntax

STX_VH_91

Exit/Next statement should be present in the loop statement denoted by the label.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an Exit/Next statement with a label that doesn't denote any labeled loop.

An Exit/Next statement with a loop label is only allowed within the labeled loop and applies only to that loop.

For example:

```
test_1: process
    begin
      L1: for i in 4 to 5 loop
    exit test_1;
-- ERROR
end loop L1;
end process;
```

Suggested Fix

Make sure that the label used in Exit/Next statement corresponds to a labeled loop in which that Exit/Next statement should be present.

Message Details

```
<Exit | Next> statement is not contained in the loop statement
denoted by the label <label>
```

Severity

Syntax

STX_VH_92

Label name must be valid (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an identifier used in place of a label but does not denote any label.

It is an error if any identifier that is not a label is used as a label. For example the following code will produce this error:

```
variable i : integer := 0;
variable K : integer := 1;
  L : for i in 1 to 10 loop
    next K when i = 3;
--Failure here : K does not denote a loop label.
end loop L;
```

Message Details

<identifier> does not denote a Label

Severity

Syntax

STX_VH_93

Discrete and same type bounds expected for a range.

Language

VHDL

Rule Description

SpyGlass reports this error to indicate the absence of discrete range bounds and same base type.

Lower and upper bounds of a range must be discrete and must belong to the same base type to form a continuous range. For example, SpyGlass reports this error for the following codes:

For different types

```
entity top is
port(input1 : in integer);
end;
architecture arch of top is
  type a is array (integer range -100 to 100) of integer;
begin
  process(input1)
    variable i1, i2, i3 : integer := 1;
  begin
    i2 := 11;
    i3 := 3;
    L1 : for c in real(i2)/i3 to i2 loop  --parameters
     --must be of same discrete type.
    end loop L1;
    end process;
end;
```

For discrete types

```
entity top is
port(input1 : in integer);
end;
architecture arch of top is
```

```
  type a is array (integer range -100 to 100) of integer;
begin
  process(input1)
    variable i1, i2, i3 : integer := 1;
  begin
    i2 := 11;
    i3 := 3;
    L1 : for c in real(i2) to real(i2) loop   --parameters
    --must be discrete.
    end loop L1;
    end process;
end;
```

For example:

```
type MVL1 is ('0', '1');
type MVL2 is ('X', 'Z');
type MVL3 is array(MVL1'LOW to MVL2'HIGH) of Integer;
```

Suggested Fix

Make both bounds of a range discrete and of same base type.

Message Details

```
The bounds of this range must be discrete and of the same type
```

Severity

Syntax

STX_VH_94

Discrete type expected.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the absence of discrete type in context.

For example:

(a)

```
  type inc_type;
  type acc is access  inc_type range 0 to 100;
 -- ERROR ('inc_type' is not a discrete type)
```

(b)

```
entity ee is
end ee;

architecture arch of ee is
    subtype XX is natural RANGE 30 DOWNTO 26;
    type Y is array(XX) of bit;
  begin
    process
  variable a : Y;
  variable b : XX;
    begin
  a(y) := "11111 -- ERROR ('y' is not a discrete type)
    wait;
  end process;
end architecture;
```

Suggested Fix

Use discrete type as per the context.

Message Details

'<type>' is not a discrete type

Severity

Syntax

STX_VH_95

Case expression cannot be of a multi-dimensional array type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of case expression which was neither a discrete nor one-dimensional character array type.

VHDL requires that the selector expression of a case statement must result in a value of a discrete type, or a one-dimensional array of character elements, such as a character string or bit string literals. This type must be determinable independently of the context in which the expression occurs, but using the fact that the expression must be of a discrete or one-dimensional character array type

For example:

```
entity ee is
end ee;

architecture arch of ee is
    type arr is array (0 to 1,0 to 1,0 to 1) of character;
begin
    process(in1,in2)
    variable var : arr;
begin
  case (var) is
-- ERROR
    when others => output <= in1 + in2;
  end case;
  end process;
end architecture;
```

Suggested Fix

Make sure that selector expression of a case statement is a discrete or one-dimensional character array type.

Message Details

Case expression cannot be of a multi-dimensional array type

Severity

Syntax

STX_VH_96

Case expression can denote an array only of character type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a case expression which was neither a discrete nor one-dimensional character array type.

VHDL requires that the selector expression of a case statement must result in a value of a discrete type, or a one-dimensional array of character elements, such as a character string or bit string literals. This type must be determinable independently of the context in which the expression occurs, but using the fact that the expression must be of a discrete or one-dimensional character array type

For example:

```
test_1: process
  type i_array_type is array (1 to 5) of integer;
    variable a1 : i_array_type := (others => 0);
  begin
    case a1 is
 -- ERROR
  end case;
end process;
```

Suggested Fix

Make sure that selector expression of a case statement is a discrete or one-dimensional character array type.

Message Details

`Case expression can denote an array only of character type`

Severity

Syntax

STX_VH_97

Either static signal name or readable name expected in sensitivity clause.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a non-static signal used in the sensitivity list.

The language requires that each signal name in the sensitivity list must be a static signal name, and each name must denote a signal for which reading is permitted.

For example:

(a)

```
test_1: process
    variable ii : integer;
  begin
    k <= 5 after 5 ns;
      wait on ii;
 -- ERROR
end process;
```

(b)

```
entity ee is
end ee;

architecture arch of ee is
    type   SWORD is ARRAY(0 to 31) of integer;
      signal   Res   : SWORD;
  begin
    test_1: process
      variable k : integer;
  begin
    for k in 0 to 31 loop
      wait on Res(k);
```

```
    end process;
end architecture;
```

Suggested Fix

Make sure that each signal name in a sensitivity list must denote a static signal.

Message Details

`The name used in the sensitivity clause is either not a static signal name or not readable`

Severity

Syntax

STX_VH_98

Wait statement cannot be used in a process, which has a sensitivity list.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of wait statement used in a function or a process which had a sensitivity list.

The presence of a sensitivity list in a process implies the presence of an implicit "wait on sensitivity-list" statement as the last statement in the process. It is an error if a wait statement appears in an explicit process statement that includes a sensitivity list, or in a procedure that has a parent that is such a process statement.

For example:

```
entity ee is
  port (s1,s2 : IN BIT);
end ee;

architecture arch of ee is
begin
test_1: process(s1,s2)
  begin
    wait on s1, s2;
  end process;
end architecture;
```

Suggested Fix

Make sure that wait statement is not present in a function or a process which has a sensitivity list.

Message Details

```
Wait statement cannot be used in a process which has a
```

```
sensitivity list
```

Severity

Syntax

STX_VH_99

Element expression of the aggregate must be a locally static name.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the absence of a locally static name for an element expression in an aggregate.

Each element expression in an association list must be evaluated when the design unit in which it is present is analyzed.

For example:

```
entity ee is
  port (s1,s2 : IN BIT);
end ee;

architecture arch of ee is
    subtype   BV2 is BIT_VECTOR(0 to 1);
    signal S : BV2;
    signal T : BV2;

begin
  test_1: process
      variable BITV : BV2     := B"11
      variable I    : integer := 1;
  begin
      (S(I), T(I)) <= BITV after 5 ns;
 -- ERROR
      wait for 10 ns;
end process;
```

Message Details

```
Each element expression of the aggregate must be a locally
static name
```

Severity

Syntax

STX_VH_100

Choice in target aggregate must be valid (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of invalid choice in target aggregate.

Suggested Fix

Use valid choice in target aggregate.

Message Details

```
Choice <invalid-choice> is not allowed in target aggregate
```

Severity

Syntax

STX_VH_101

OTHERS is allowed as a choice only for the last association element

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of an aggregate containing one or more element associations following an OTHERS association. OTHERS choice must be the final element in the association list. Hence, if keyword others is used, it must be the last choice in the aggregate. Nothing can follow an others association.

Message Details

```
OTHERS is allowed as a choice only for the last association
element
```

Severity

Syntax

STX_VH_102

Configuration name must be defined before usage

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of undefined configuration name.

Message Details

```
No configuration name "<undefined-config-name>" has been
defined
```

Severity

Syntax

STX_VH_103

Errors found in Port Map

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of errors in port map. This error might occur, for example if a local port is associated more than once in port map or a signal in port map does not denote a static signal name.

For example:

```
function increment ( bv : bit_vector )
  return bit_vector is
  begin
    return bv;
  end;

  component mycomp
    port (portIn : in bit_vector(3 downto 0);
          portOut : out bit_vector(3 downto 0));
  end component;

  myinst : mycomp
  port map (increment(next_count),
            portOut => next_count);
  myinst1 : mycomp
  port map (
  portIn => increment(next_count & next_count),
   portOut => next_count);
```

Message Details

```
Errors found in Port Map
```

Severity

Syntax

STX_VH_104

Errors found in Generic Map

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of errors in generic map. This error might occur, for example, if a local generic is associated more than once in generic map as in the following example:

```
component input2
  generic (input_1 : in  bit;
           input_2 : in  bit;
           output  : in  bit);
end component;

constant A1 : bit := '1';
constant X : bit_vector(0 to 1) := '11'
BEGIN
  G1: input2
    generic map (input_1 => X(0), input_1 => X(1),
            output => A1);
END;
```

NOTE: *One of the possible causes of the STX_VH_104 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Message Details

```
Errors found in Generic Map
```

Severity

Syntax

STX_VH_105

Identifier must be declared in the scope of subprogram.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of an identifier that is used but not declared in the scope of subprogram. Any identifier that is used in the scope of subprogram must first be declared in the subprogram before it is used.

Message Details

```
'<identifier>' is not declared in the scope of the subprogram
'<sub-program>'
```

Severity

Syntax

STX_VH_106

The actual associated with the variable parameter must be a variable

Language

VHDL

Message Details

```
Actual associated with the <variable-type1> parameter
<variable-name> must be a <variable-type2>
```

Severity

Syntax

STX_VH_107

Port/generic of mode OUT cannot be associated with the IN parameter.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that OUT port/generic cannot be associated with the IN parameter. After a given description is completely elaborated, if a formal port is associated with an actual that is itself a port, then the following restrictions apply depending upon the mode of the formal port:

1. For a formal port of mode IN, the associated actual may only be a port of mode IN, INOUT, or BUFFER.
2. For a formal port of mode OUT, the associated actual may only be a port of mode OUT or INOUT.
3. For a formal port of mode INOUT, the associated actual may only be a port of mode INOUT.

Message Details

```
Cannot associate OUT port/generic with the IN parameter
<parameter>
```

Severity

Syntax

Suggested Fix

Use correct association as per the restrictions described above.

STX_VH_108

Port/generic of mode IN and constant objects cannot be associated with OUT parameter.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that constants, IN port/generic or expression cannot be associated with OUT parameter. Constant parameter cannot be associated with OUT parameter. As far as port association is concerned, the language applies the following restrictions depending upon the mode of the formal port:

1. For a formal port of mode IN, the associated actual may only be a port of mode IN, INOUT, or BUFFER.
2. For a formal port of mode OUT, the associated actual may only be a port of mode OUT or INOUT.
3. For a formal port of mode INOUT, the associated actual may only be a port of mode INOUT.

Message Details

```
Cannot associate constant, IN port/generic or expression with
OUT parameter <parameter>
```

Severity

Syntax

Suggested Fix

Do not associate constant, IN port/generic or expression with OUT parameter.

STX_VH_109

Package declaration must be compiled before compiling corresponding package body.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the erroneous order of analysis/compilation of package body (secondary unit) before corresponding package declaration (primary unit). VHDL has "order of analysis" which contains rules (direct consequences of visibility rules) defining the order in which design units can be analyzed. In particular

1. A primary unit whose name is referenced within a given design unit must be analyzed prior to the analysis of the given design unit.
2. A primary unit must be analyzed prior to the analysis of any corresponding secondary unit.

Message Details

```
Body defined for package <package> which has not yet been compiled
```

Severity

Syntax

Suggested Fix

Compile package declaration before compiling corresponding package body.

STX_VH_110

Entity must be compiled first before compiling corresponding architecture.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the erroneous order of analysis/compilation of architecture (secondary unit) before corresponding entity (primary unit). VHDL has "order of analysis" which contains rules (direct consequences of visibility rules) defining the order in which design units can be analyzed. In particular

1. A primary unit whose name is referenced within a given design unit must be analyzed prior to the analysis of the given design unit.
2. A primary unit must be analyzed prior to the analysis of any corresponding secondary unit.

Message Details

```
Cannot compile architecture <architecture> until corresponding
entity <entity> has been compiled
```

Severity

Syntax

Suggested Fix

Compile Entity before compiling corresponding architecture.

STX_VH_112

Deferred constant usage must be legal (by context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the illegal usage of deferred constant. Within a package declaration, that contains the declaration of a deferred constant, and within the body of corresponding package body before the end of the corresponding full declaration, the use of a name that denotes the deferred constant is only allowed in the default expression for a local generic, local port, or formal parameter. The result of evaluating an expression that references a deferred constant before the elaboration of the corresponding full declaration is not defined by the language.

Message Details

```
Invalid use of deferred constant <deferred-constant>
```

Severity

Syntax

Suggested Fix

Use deferred constant only in default expression for a local generic, local port, or formal parameter before full constant declaration in package body.

STX_VH_113

Only static expression allowed in present context.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the static expression is required in the context. There are two categories of static expression. Certain forms of expression can be evaluated can be evaluated during the analysis of the design unit in which they appear; such an expression is said to be locally static. Certain forms of expression can be evaluated as soon as the design hierarchy in which they appear is elaborated; such an expression is said to be globally static. For example, the if-generate statement allows for conditional selection of concurrent statements based on the value of an expression. This expression must be globally static expression.

For example:

```
entity gen5 is
  port ( in1 : bit_vector ( 0 to 7);
         in2 : bit_vector ( 0 to 7);
         in11 : boolean;
         in12 : boolean;
         out1 : out bit_vector ( 0 to 7) );
end gen5;

architecture gen5 of gen5 is
begin
  l1: if ( in11 xor  in12 )
        generate
          out1<= in1;
        end generate;
  l2: if ( not ( in11 xor in12 ))
        generate
          out1<= in2;
        end generate;
end;
```

Message Details

```
Only static expression allowed in this context
```

Severity

Syntax

STX_VH_114

Architecture corresponding to referenced entity does not exit.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that the architecture (by context) corresponding to an entity does not exist. Either you have defined an architecture but not using the same name for it or you have not defined an architecture at all. Check your code, use correct name for an architecture (in case you have defined it) or define architecture (in case you have not defined it) and then reanalyze your design.

Message Details

```
Architecture <architecture> (<reference-entity>) does not exist
```

Severity

Syntax

STX_VH_115

Reference used must be valid in present context.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that it could not find an identifier which has been used in your code as a reference. A reference is an access to a named entity. Every appearance of a designator (a name, character literal or operator symbol) is a reference to the named entity denoted by the designator, unless the designator appears in a library clause or use clause. SpyGlass flags an error if a reference is used to access any entity that does not exist.

Message Details

```
No reference <identifier> found
```

Severity

Syntax

STX_VH_116

Entity referenced either does not exist or has not been compiled before usage.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that an entity (by context) does not exist or a configuration declaration and corresponding entity declaration do not reside in the same library. For a configuration of a given design entity, both the configuration declaration and the corresponding entity declaration must reside in the same library. SpyGlass flags an error if configuration declaration and corresponding entity declaration reside in different libraries or if entity referenced has not been defined.

Message Details

`No Entity <entity1>.<entity2> Exists`

Severity

Syntax

STX_VH_118

Configuration referenced either does not exist or has not been compiled before usage.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that a configuration does not exist or it exists but is not compiled before usage.

Message Details

```
No Configuration <configuration> exists
```

Severity

Syntax

STX_VH_119

Configuration referenced either does not exist or has not been compiled before usage.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that a configuration does not exist or it exists but is not compiled before usage.

Message Details

`No Configuration <configuration1>.<configuration2> exists`

Severity

Syntax

STX_VH_120

Illegal component instantiation.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of an identifier used to denote a component instance but does not appear in component instantiation statement. A specification applies to those instances of the specified component declaration whose labels are declared in the immediately enclosing declarative part. It is an error if the component instance label is used but that label does not appear in corresponding component instantiation statement.

Message Details

```
<identifier> does not denote an instance of the component
<component>
```

Severity

Syntax

STX_VH_121

Index constraint may be applied only on an unconstrained array type.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of index constraint applied on an object of constrained array type. When unconstrained array is declared, the number of elements in the array is not specified in the type declaration. Instead, an object declaration for an object of that type declares the number of elements of the array. A subtype declaration constraining the array type may also specify the index constraint for an unconstrained array type.

For example:

```
type week  is array (positive range <>) of integer;
type a is access week;
subtype weekend1 is week (10 to 20);
subtype weekend2 is a (10 to 20);
type week2    is array (1 to 10) of integer;
type b is access week2;
subtype weekend3 is week2 (1 to 2);
```

SpyGlass flags the above example as an index constraint is not allowed in the subtype declaration of `weekend3`.

Message Details

```
An index constraint may be applied only on an unconstrained
array type.
```

Severity

Syntax

STX_VH_122

Arrays must not have more than appropriate number of range constraints (by context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of too many range constraints specified than the base array type. Array subtype cannot have more constraints than base array type. Therefore, the following code will produce this error:

```
type A is ARRAY (NATURAL RANGE <>) of INTEGER;
SUBTYPE A8 IS A (1 TO 8,1 TO 8,1 TO 8);
```

SpyGlass flags the above example as too many range constraints are specified.

Message Details

`Too many range constraints specified for this <num>-dimensional array`

Severity

Syntax

STX_VH_123

Arrays must not have less than appropriate number of range constraints (by context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of too few range constraints specified for an array. When an object of an array type is declared, then the declaration of that object must specify as many range constraint as the dimension of base array. For example, the following code will produce this error:

```
type BIT_VECTOR is array (natural range <>,
   positive range <>) of BIT;
variable NUM1 : BIT_VECTOR(0 to 1) :=
         ( ('0', '0'), ('1', '1'),
           ('0', '1'), ('1', '1'),
           ('0', '1'), ('0', '1'),
           ('1', '0'), ('1', '0')  );
```

Message Details

```
Too few range constraints specified for this <num>-dimensional
array
```

Severity

Syntax

STX_VH_124

Component instance must be configured at most once using a configuration specification.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of a component instance which was configured more than once. Component instance must be configured only once in a configuration specification.

Message Details

```
Component instance <instance> has already been configured using
a prior configuration specification
```

Severity

Syntax

STX_VH_125

Configuration specification must have an entity aspect.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the absence of entity aspect in configuration specification. A binding indication associates instances of a component declaration with a particular design entity. The entity aspect of a binding indication, if present, identifies the design entity with which the instances of a component are associated. When a binding indication is used in a configuration specification, it is an error if the entity aspect is absent.

Message Details

```
Configuration specification must have an entity aspect
```

Severity

Syntax

STX_VH_126

Expression value must not be out of range (in context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of an out of range expression value.

Message Details

`Expression value "<value>" is out of range "<range>"`

Severity

Syntax

Suggested Fix:

Evaluate expression value within the range by context.

STX_VH_129

Others clause must be used here (aggregate has unspecified record fields).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of record aggregate which did not specify all the record fields. If a record aggregate does not specify all the record fields, use OTHERS clause to cover the remaining fields.

Message Details

```
OTHERS clause must be used here (aggregate has unspecified
record fields).
```

Severity

Syntax

STX_VH_130

OTHERS clause is not allowed after all members have been assigned.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of OTHERS clause after all members of composite type have been assigned in an aggregate. The choice OTHERS is only allowed for the last alternative and as its only choice; it stands for all values (possibly none) not given in the choices of previous alternatives. If the previous alternatives cover all the members of composite type object, then there is no need for OTHERS clause. It is an error if OTHERS clause is used in such cases.

Message Details

```
OTHERS clause not allowed as all members have already been
assigned
```

Severity

Syntax

STX_VH_131

Record elements covered by the OTHERS clause must not have different type.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that the record elements covered by the OTHERS clause have different types. If the choice OTHERS is given as a choice of a record aggregate, it must represent at least one element. An element association with more than one choice, or with the choice OTHERS, is only allowed if the elements specified are all of the same type. The expression of an element association must have the type of the associated record elements.

Message Details

`The record elements covered by the OTHERS clause have different types`

Severity

Syntax

STX_VH_132

Wait statement is not allowed in a function, or in a procedure invoked from a function.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of wait statement used in a function, or in a procedure invoked from a function. Functions are used to compute values that are available instantaneously. Therefore, a function cannot be made to wait; for example, it cannot call a procedure with a wait statement in it. Hence, do not use wait statement inside function, or in a procedure invoked from a function.

Message Details

```
Wait statement is not allowed in a function, or in a procedure
invoked from a function
```

Severity

Syntax

STX_VH_133

Wait statement cannot appear in a procedure which has been invoked from a process with sensitivity list.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of wait statement in a procedure which is invoked from a process with sensitivity list. A process that calls a procedure with a wait statement cannot have a sensitivity list. This follows from the fact that a process cannot be sensitive to signals and also be made to wait simultaneously. Therefore, do not use wait statement in a procedure which has been invoked from a process with a sensitivity list.

Message Details

```
A wait statement cannot appear in a procedure which has been
invoked from a process with sensitivity list.
```

Severity

Syntax

STX_VH_134

Signal assignments are not allowed in a process statement inside an entity.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of signal assignments in a process statement inside an entity. A process statement inside an entity must be passive. A process statement is said to be a passive process if neither the process itself, nor any procedure of which the process is a parent, contains a signal assignment statement. Such a process, or any concurrent statement equivalent to such a process, may appear in the entity statement part of an entity declaration.

Message Details

```
Signal assignments not allowed in a process statement inside an
entity
```

Severity

Syntax

STX_VH_135

Procedure call in an entity must be passive.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of non-passive procedure call in an entity. A procedure call is said to be passive if it does not contain any signal assignment statements. Such a procedure may appear in the entity statement part of an entity declaration.

Message Details

```
Procedure call in an entity must be passive
```

Severity

Syntax

STX_VH_136

End Label usage must be valid (by context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the invalid presence of end label as there is no corresponding start label.

Message Details

```
End Label <invalid-end-label> not allowed for this <name> which
does have a Start Label
```

Severity

Syntax

STX_VH_137

No more choices are permitted after OTHERS clause.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the invalid presence of choices used after OTHERS clause. The choice OTHERS is only allowed for the last alternative and as its only choice; it stand for all values (possibly none) not given in the choices of previous alternatives. Hence when OTHERS clause is used, it must be the last alternative in the list of alternatives and it must be expressed as a single value, not as a range of values by using | (vertical bar).

Message Details

```
No more choices permitted after OTHERS clause
```

Severity

Syntax

STX_VH_138

Choice expression should be locally static.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that choice expression should be locally static. A locally static expression is an expression which can be evaluated during the analysis of the design unit in which it appears. The language requires that selector expression of a case statement must result in a value of a discrete type, or a one dimensional array of character elements, such as a character string or bit string and it must be locally static.

For example, the STX_VH_138 rule flags a violation in the following case:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_unsigned.ALL;

entity myentity is
  port (
    inputA : in std_logic_vector(3 DOWNTO 0);
    inputB : in std_logic_vector(3 DOWNTO 0);
    inputSEL : in std_logic_vector(1 DOWNTO 0);
    outputZ : out std_logic_vector(3 DOWNTO 0)
  );
end entity myentity;

architecture myarch of myentity is
  signal ROT_0 : std_logic_vector(1 DOWNTO 0) :="00";

Begin
  myproc:
  process (inputSEL)
  begin
    case inputSEL is
      when  ROT_0 =>  outputZ <= inputA;
```

```
      -- ROT_0 is not statically evaluable
      when others => outputZ <= inputB;
    end case;
  end process myproc;
end myarch;
```

Message Details

```
Choice expression should be locally static
```

Severity

Syntax

STX_VH_139

Each choice must be covered in list of case statement alternatives.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that case statement alternatives do not cover each and every possible value of selector expression. VHDL requires that each and every possible value of the selector expression must be covered exactly once either through listing all possible choices (after when keyword) in the case statement body or using OTHERS clause to cover the remaining values of selector expression which are not covered by previous alternatives.

Message Details

```
Choice <choice> not covered in list of case statement
alternatives
```

Severity

Syntax

STX_VH_140

All choices must be distinct from each other in CASE statement.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of a choice which overlaps with other entries in case statement. VHDL requires that all possible values of the expression must be covered in the case statement exactly once. There must be exactly one choice for each possible value.

Message Details

```
Choice <choice> overlaps with other entries in case statement
```

Severity

Syntax

STX_VH_141

OTHERS clause must be present in CASE statement when choice list is incomplete.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the absence of OTHERS clause when choice list is incomplete for CASE expression. In CASE statement, if the value of expression is outside the range of the choices given, then the OTHERS clause matches the expression and the statement following the OTHERS clause is executed. It is an error if an OTHERS clause does not exist, and the choices given do not cover every possible value of the expression type.

Message Details

```
Choice list is incomplete for CASE expression : needs OTHERS clause.
```

Severity

Syntax

STX_VH_142

Prefix for the Slice name must be legal (that is, the prefix must be a static name, one dimensional array object, function or type mark).

Language

VHDL

Rule Description

SpyGlass reports this error to indicate presence of an illegal prefix for a slice name in the following context:

- The prefix for the slice name is not a static name.
- The prefix is a type a conversion expression.

For example, this rule reports a violation in the following case:

```
LIBRARY IEEE;
USE     IEEE.numeric_std.ALL;
USE     IEEE.std_logic_1164.ALL;


entity pack is
  port(
  in1          : in     std_logic;
  out1         : out    std_logic_vector(23 DOWNTO 0)
  );
end pack ;


architecture rtl of pack IS
  signal   sig1  : std_logic_vector(31 DOWNTO 0);
  signal   sig2  : std_logic_vector(31 DOWNTO 0);
  signal   sig3  : unsigned(31 DOWNTO 0);
  begin
    out1 <= std_logic_vector(sig3)(23 DOWNTO 2) & "00";
  end rtl;
```

To remove the violation in the above example, modify the code so that you can take a slice and then change its type by type convention, as shown

below:

```
out1 <= std_logic_vector(sig3(23 DOWNTO 2)) & "00";
```

Message Details

```
<prefix> does not denote a legal prefix for the Slice name
```

Severity

Syntax

STX_VH_143

Slice direction must match with type definition.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the mismatch between the slice direction and type definition. The bounds of the discrete range (of the slice) define those of the slice and must be of the type of the index of the array. It is an error if the direction of the discrete range is not the same as that of the index range of the array denoted by the prefix of the slice name.

Message Details

```
Slice direction does not match with type definition
```

Severity

Syntax

STX_VH_144

Slice length in expression must be valid (by context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of invalid slice length in expression.

Message Details

```
Invalid slice length in expression "<expression>"
```

Severity

Syntax

Suggested Fix

Use valid slice length as per the context.

STX_VH_145

Invalid operands found in expression.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate usage of invalid operands in expression.

Message Details

```
Invalid operands in expression "<expression>"
```

Severity

Syntax

Suggested Fix

Use valid operands as per the operator.

STX_VH_146

Operands of different base types are not allowed in a logical operation.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of operands of different base types used in a logical operation. For the binary operators AND, OR, NAND, NOR, XOR, and XNOR, the operands must be of the same base type.

Message Details

```
Operands of different base types not allowed in a logical
operation [ LHS="<LHS-expression>", RHS="<RHS-expression>" ]
```

Severity

Syntax

STX_VH_147

At least two operands are required in the expression (in context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that it expects at least two operands in expression (by context).

Message Details

```
At least two operands are required in the expression
"<expression>"
```

Severity

Syntax

STX_VH_148

Operands of different lengths are not allowed in a logical operation.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of operands of different lengths used in a logical operation. For the binary operators AND, OR, NAND, NOR, XOR, and XNOR, the operands must be of the same base type. Moreover, for the binary operators AND, OR, NAND, NOR, XOR, and XNOR defined on one-dimensional array types, the operands must be arrays of the same length, the operation is performed on matching elements of the arrays, and the result is an array with the same index range as the left operand. For the unary operator NOT defined on one-dimensional array types, the operation is performed on each element of the operand, and the result is an array with the same index range as the operand.

Message Details

```
Operands of different lengths not allowed in a logical
operation [ LHS="<LHS-expression>", RHS="<RHS-expression>" ]
```

Severity

Syntax

STX_VH_149

Slice direction must match the index range direction of prefixing array in expression (by context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the mismatch between slice direction and index range direction of prefixing array in an expression. When we use slice notation, we specify the left and right index values of part of an array object. These ranges specified in the slice must have the same direction as the original array.

For example:

```
type array1 is array(1 to 100)of integer;
type array2 is array(100 downto 1)of integer;
variable a1:array1;
variable a2:array2;
```

Now if we take array slices such as a1(20 downto 11) and a2(11 to 20), these slices have ranges whose direction does not match as that of original array.array1 and array2 has increasing and decreasing range while their slices have decreasing and increasing ranges respectively.

Message Details

```
Slice direction does not match the index range direction of
prefixing array in expression "<expression>"
```

Severity

Syntax

Suggested Fix

Match slice direction with index range direction of original array.

STX_VH_150

No more element associations are allowed after OTHERS clause.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the invalid presence of element associations used after OTHERS clause in an association list. When named association (alone or after positional association) is used in an association list, the OTHERS clause can be used for elements not covered in previous element associations in the association list. However if OTHERS clause is used, it must be the last named association in an association list.

For example:

```
type point is array(1 to 3) of real;

variable view_point:
  point:=(1=>10.0, others=>20.0,3=>0.0);
```

Message Details

```
No more element associations allowed after OTHERS clause
```

Severity

Syntax

STX_VH_151

Positional association is not allowed after the first named association.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of positional association after the first named association in an association list. The language allows the usage of both positional association and named association in same association list (for record aggregates and OTHERS clause as the only named association for array aggregates) but with a restriction that positional association cannot be used in an association list after the first named association in that association list.

Message Details

```
Positional association not allowed after the first named
association
```

Severity

Syntax

STX_VH_152

Duplicate association in association list is not allowed.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of an element covered more than once in an association list. Each element of the value defined by an aggregate must be represented once and only once in the aggregate.

Message Details

```
Duplicate association at position <position> of association
list
```

Severity

Syntax

STX_VH_154

Size of LHS must match the size of RHS for operator.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of size mismatch between LHS and RHS operands of a binary operator used in your code. The following is the list of operators, which require both their operands to be of same type and size:

- Multiplication
- Division
- Modulo
- Remainder
- Addition
- Subtraction
- Concatenation
- Equality
- Less than
- Logical operators like ‘and’.

Hence, the following code will produce this error:

```
type array_one is array(positive range <>)of boolean;
 variable x:array_one(1 to 10);
 variable y:array_one(1 to 5);
 variable z:array_one(1 to 10);
 z:=(x and y); --Failure here: x and y do not have same size.
```

Suggested Fix

Use left and right operands of same size for operator used.

Message Details

```
Size mismatch between LHS and RHS for operator "<operator>" in
expression [ LHS="<LHS-expression>", RHS="<RHS-expression>" ]
```

Severity

Syntax

STX_VH_155

Use only locally defined identifiers in declarative regions.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of an identifier in a declarative region in which it was not defined. VHDL requires that any identifier should be used in a region in which it is defined. For example, the following code will produce this error:

```
architecture TEST_BEHAVIOR of TEST is
  --Define res function for SIG:
  function RESFUNC(S:BIT_VECTOR)return BIT is
  begin
   return '1';
  end RESFUNC;
  --Define the signal.
  subtype RBIT is RESFUNC BIT;
  signal SIG:RBIT bus;

  begin
   L:block
   disconnect SIG:RBIT after 0ns. --Error here:SIG is not
   -- defined in this declarative region.
   begin
   end block L;
end TEST_BEHAVIOR;
```

Message Details

```
<identifier> is not defined in this declarative region
```

Severity

Syntax

STX_VH_156

Identifier (in context) must be of the entity class specified in the attribute specification.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of an identifier which is not of the entity class specified in an attribute specification. An attribute specification associates a user-defined attribute with one or more named entities and defines the value of that attribute for those entities. The syntax of attribute specification is:

```
attribute_specification::=

attribute attribute_designator of entity_specification is expression;

entity_specification::=

entity_name_list:entity_class
```

The attribute designator must denote an attribute. The entity name list identifies those named entities, both implicitly and explicitly defined, that inherits the attribute, described as follows:

If a list of entity designators is supplied, then the attribute specification applies to the named entities denoted by those designators. It is an error if the class of those names is not the same as that denoted by the entity class.

Message Details

```
<identifier> is not of the entity class specified in the attribute specification
```

Severity

Syntax

STX_VH_157

Attributes cannot be overloaded.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of more than one attribute associated with same named entity in the description. It is an error if a given attribute is associated more than once with a given named entity. Similarly, it is an error if two different attributes with the same simple name (whether predefined or user-defined) are both associated with a given named entity. For example, the following code will produce this error:

```
 architecture arch of ent is
   function func (in1: integer:=10) return integer is
   begin
     return 10;
   end;
   function func (in1: integer:=10) return bit is
   begin
    return '1';
   end;
   type xx is ('a','b');
   type xx1 is ('a','b');
 attribute att : integer;
 attribute att of func [integer return bit] : function is 10;
   attribute att of func : function is 10;  //Error here:
attribute att has already been associated with func.
 begin
end;
```

Message Details

```
Attribute "<attribute-name>" has already been associated with
"<entity-name>" in file '<file-name>' at line <line-num>
```

Severity

Syntax

STX_VH_158

Size of string must match the required size.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that string size does not match the size required in the context in which it appeared.

Strings represent one-dimensional array of characters. The number of characters in the string is the size of that string. For example, SpyGlass reports this error in the following case:

```
ENTITY ent IS
END ent;


ARCHITECTURE arch of ent IS
  type string4 is array( 1 to 4 ) of CHARACTER;
  signal x : string4;
  signal y : integer;
  BEGIN
    with x select
    y <= transport  1 when "one",
    -- Select expression in a selected assignment
    -- statement is  not the same type of a choice.
    2 when "two",
    0 when others;
  END arch;
```

This rule also reports violation in the following case:

```
entity ent1 is
port (

in_clk             : in bit;
emususp_out        : out bit
);
```

```
end ent1;

architecture rtl of ent1 is
signal cnt         : bit_vector(2 downto 0);
constant int1      : integer := 10;
begin
p_cnt: process (in_clk) begin
  if (int1=1) then
    cnt <= "00";
  end if;
end process;
end rtl;
```

NOTE: *The WRN_158 warning has been changed to the STX_VH_158 syntax error. SpyGlass reports this syntax error for target and source width mismatch as done by the synthesis tools.*

Message Details

```
Size '<string-size>' of string "<string>" does not match the
required size '<required-size>'
```

Severity

Syntax

STX_VH_159

OTHERS or ALL not allowed when specifying attributes on a design unit.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of usage of OTHERS or ALL clauses when attributes were specified on the design unit.

If the reserved word OTHERS is supplied, then the attribute specification applies to named entities of the specified class that are declared in the immediately enclosing declarative part, provided that each such entity is not explicitly named in the entity name list of a previous attribute specification for the given attribute.

If the reserved word ALL is supplied, then the attribute specification applies to all named entities of the specified class that are declared in the immediately enclosing declarative part. An attribute specification for an attribute of a design unit (that is, an entity interface, an architecture, a configuration, or a package) must appear immediately within the declarative part of that design unit. Hence, OTHERS or ALL are not allowed when specifying attributes on a design unit.

Message Details

```
OTHERS or ALL not allowed when specifying attributes on a
design unit
```

Severity

Syntax

STX_VH_160

Parameter for attribute must be valid in present context.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of invalid parameter for attribute in suffix expression in your code. If the attribute designator denotes a predefined attribute, the expression either must or may appear, depending upon the definition of that attribute; otherwise it must not be present. Only certain predefined attributes may be followed by a single generic expression enclosed in parenthesis representing the attribute parameter.

Suggested Fix:

See the definition of predefined attribute and use that attribute as per the definition.

Message Details

```
Invalid parameter for attribute in suffix-expression
"<expression>"
```

Severity

Syntax

STX_VH_161

Prefix for attribute must be valid in present context.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of invalid prefix for attribute used in your VHDL code. The common prefixes of predefined attributes defined by the language are-:

- Any prefix that is any scalar type or subtype, any discrete or physical type or subtype.
- Any prefix that is appropriate for an array object, or an alias thereof, or that denotes a constrained array subtype.
- Any prefix that denotes a static signal name
- Any prefix that is any named entity or any named entity other than a local port or a local generic of a component declaration.

Suggested Fix

See the definition of attribute and then use valid prefix as per the definition.

Message Details

```
Invalid prefix "<prefix>" for attribute "<attribute>"
```

Severity

Syntax

STX_VH_162

Prefix TYPE for attribute must be valid in present context.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of invalid prefix type for attribute used in your VHDL code. The common prefixes of predefined attributes defined by the language are-:

- Any prefix that is any scalar type or subtype, any discrete or physical type or subtype.
- Any prefix that is appropriate for an array object, or an alias thereof, or that denotes a constrained array subtype.
- Any prefix that denotes a static signal name
- Any prefix that is any named entity or any named entity other than a local port or a local generic of a component declaration.

Suggested Fix

See the definition of attribute and then use valid prefix type as per the definition.

Message Details

```
Invalid prefix type for attribute "<attribute>"
```

Severity

Syntax

STX_VH_163

Attribute parameter must be a locally static expression.

Language

VHDL

Rule Description

This error occurred because attribute parameter for attribute was not a locally static expression. A locally static expression is an expression that can be evaluated during the analysis of the design unit in which it appears. The language requires that attribute parameter should be a locally static expression.

For example, the following code will produce this error:

```
    entity lll is
    end lll;
    architecture kkk of lll is
        type my_array is array(1 to 10, 4 to 10) of boolean;
        signal bb : my_array;
        signal aa : integer;
        procedure proc(in1 : in integer) is
            variable var1 : integer;
        begin
           for i in bb'range(in1) loop  --Error here:in1 in
attribute bb'range(in1) is not locally static.
            end loop;
        end;
    begin
end;
```

Message Details

```
Attribute parameter "<parameter>" must be a locally static
expression
```

Severity

Syntax

STX_VH_164

Parameter type should be of type UNIVERSAL INTEGER (in context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of parameter which was not of type UNIVERSAL_INTEGER but was required to be of that type as per the context. This error might occur, for example if you are using predefined attributes, which require parameters as locally static expression of type universal_integer. For example, the following code will produce this error:

```
type small_int is range 0 to 7;
type cmd_bus is array (small_int) of bit;
signal s_int : small_int := 0;
signal s_bus : cmd_bus;
s_int <= s_bus'right(small_int); --Error here:small_int
--must be of type universal_integer.
```

Message Details

```
Parameter type for "<parameter>" should be of type UNIVERSAL
INTEGER
```

Severity

Syntax

STX_VH_165

Parameter must not be out of range (by context).

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of parameter out of range. This error might occur for example, when predefined attribute is used and its parameter does not lie within the range as specified by the language. So, the following code will produce this error:

```
    type opcode is
(Add,Add_with_carry,Sub,Sub_with_carry,complement);

    subtype Adding_opcode is opcode range add to
Add_with_carry;

    variable a : opcode;
    variable b : natural;
    variable c : Adding_opcode;

    b := integer'rightof(2);               --CORRECT
    b := integer'rightof(integer'high);    --ERROR:
-- Parameter integer'high does not belong to range
-- integer'low to integer'high.

    b := integer'rightof(integer'right);   --ERROR:
```

Message Details

```
Parameter "<parameter>" is out of range
```

Severity

Syntax

STX_VH_166

Case expression must be a discrete or one-dimensional character array type.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of case expression, which did not denote a discrete type. The language requires that selector expression of a case statement must result in a value of a discrete type or a one-dimensional array of character elements such as character string or bit string. The type must be determinable independent of the context in which the expression occurs but using the fact that expression must be a discrete type or one-dimensional character array type. For example, the following code will produce this error:

```
entity ent is
end entity;
architecture arch of ent is
  begin
   process
    variable r1:real :=0.1;
    begin
     case r1 is
      when 0.0 to 1.0 =>
       --
      when others =>
       --

     end case;
  end process;
end arch;
```

Message Details

```
Case expression "<expression>" does not denote a discrete type
```

Severity

Syntax

STX_VH_167

Element of the multidimensional array must be aggregate.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it could not find aggregate for accessing multidimensional array element. The language requires that to access any element of a multidimensional array, an aggregate must be formed in which as many numbers (separated by commas) as the dimensions of the array must be present.

Message Details

```
The Element of the multidimensional array must be aggregate
```

Severity

Syntax

STX_VH_168

Choice list for aggregate must be complete.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of incomplete choice list for aggregate. There must be one-to-one correspondence between values in the aggregate and the elements of composite type. Hence, if some elements are left un-initialized in the aggregate either initialize them individually or use an OTHERS clause. But it is an error if some elements are left un-initialized in the aggregate.

Message Details

```
Incomplete choice list for aggregate "<aggregate>"
```

Severity

Syntax

STX_VH_171

Constraint must form a continuous range for aggregate.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of a constraint, which did not form a continuous range for aggregate. Constraint used for aggregate must form discrete and continuous bounds (and hence range) for an aggregate.

Message Details

```
Constraint must form a continuous range for aggregate
"<aggregate>"
```

Severity

Syntax

STX_VH_172

Physical type not supported.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that physical type was not supported in your code.

Message Details

```
Physical type not supported
```

Severity

Syntax

STX_VH_173

Actual signal associated with signal parameter must denote a static name.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate it did not find static name for actual signal which was associated with signal parameter. Actual signal corresponding to signal parameter must be evaluated during the analysis of design unit in which it is present.

Message Details

```
Actual associated with the signal parameter <parameter> must
denote a static name
```

Severity

Syntax

STX_VH_174

Ordering operator (in context) must be defined for RECORD type.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of ordering operator used for RECORD type. The ordering operator is defined for any scalar type and for any discrete array type.

Message Details

```
Ordering operator "<operator>" not defined for RECORD type
```

Severity

Syntax

STX_VH_175

Divide by zero error.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of division by zero in your code. Avoid division by zero in your code.

Message Details

```
Divide by zero error
```

Severity

Syntax

STX_VH_176

Attribute for a formal signal parameter cannot be read.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it could not read an attribute for a formal signal parameter. It is an error if signal-valued attributes 'STABLE,'QUIET,'TRANSACTION, AND 'DELAYED of formal signal parameters of any mode are read within a subprogram. Hence, the following code will produce this error:

```
     ARCHITECTURE arch OF ent IS
     procedure proc1 (signal S1: inout bit) is
          variable V1 : boolean;
     begin
      -- Failure_here : attribute STABLE may not be read
within a procedure
          V1 := S1'STABLE;
     end proc1;
     BEGIN
END;.
```

Message Details

```
Cannot read attribute for a formal signal parameter
"<parameter>"
```

Severity

Syntax

STX_VH_177

Port/Generic map aspect must be valid (by context).

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that the Port/Generic map aspect was not valid in the context in which it appeared.

Suggested Fix:

Use Port/Generic map aspect in a context where it is valid.

Message Details

```
Port/Generic map aspect "<aspect>" is not valid in this context
```

Severity

Syntax

STX_VH_178

Attribute name may have a signature only if the prefix denotes an entity that can be overloaded.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of signature used in an attribute name but prefix of the attribute did not denote an entity that can be overloaded. In the case of attributes decorating subprograms or enumeration literals, it may be necessary to use a signature to distinguish between a number of alternative names. However, if the name of the item is unambiguous, we can simply write an apostrophe and attribute name after the item name.

Message Details

```
Attribute name may have a signature only if the prefix denotes
an entity that can be overloaded
```

Severity

Syntax

STX_VH_179

OTHERS clause must be the only choice in an association.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it did not find OTHERS clause alone in an association. We can use the symbol "|" to include more than one choices in same association. But if OTHERS is used it must be the only choice in association. For example, the following code will produce this error:

```
        type ARRAY_TYPE is array (INTEGER range <>) of
BOOLEAN;
        type RECORD_TYPE is record
                E1,E2,E3 : BOOLEAN;
        end record;
        signal S2 : RECORD_TYPE;
        S2 <= (E2 => TRUE, others | E1 => FALSE);
-- Failure_here
-- SEMANTIC ERROR:  "others" must be only choice in an
association.
```

Message Details

```
OTHERS clause must be the only choice in an association [
"<expression>" ]
```

Severity

Syntax

STX_VH_180

RECORD type elements covered by a choice list must all be of the same type.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it did not find all record type elements of the same type covered by a choice list. In record aggregate, an element association with more than one choice is only allowed, if the elements specified are all of the same type. The expression of an element association must have the type of the associated record elements. For example, the following code will produce this error:

```
      type rec is record
                ele_1 : integer;
                ele_2 : real;
                ele_3 : boolean;
        end record;
constant p  :rec := (ele_1 | ele_2 | ele_3 => 4.5);
-- Failure_here.
```

Message Details

```
RECORD type elements covered by a choice list must all be of
the same type "<type>"
```

Severity

Syntax

STX_VH_181

Invalid component aspect specified with a procedure call.

Language

VHDL

Rule Description

Message Details

```
Invalid component aspect specified with procedure call
"<procedure-call>"
```

Severity

Syntax

STX_VH_182

Association must be valid with a component instance.

Language

VHDL

Rule Description

Message Details

```
Association not valid with component instance
"<component-instance>"
```

Severity

Syntax

STX_VH_183

Resolution function must be valid (by context).

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of invalid resolution function.

Suggested Fix:

Use valid resolution function as per the context.

Message Details

`<function> is not a valid resolution function`

Severity

Syntax

STX_VH_184

Invalid use of undefined attribute.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of invalid use of undefined attribute. The language does not allow such a use of attribute that may lead to ambiguity. For example, the following code will produce this error:

```
     entity ent is
     end;
     architecture arch of ent is
        function func (in1: integer) return integer is
            begin
               return 10;
            end;
        function func (in1: integer) return bit is
            begin
               return '1';
            end;
        attribute att : integer;
        attribute att of func [integer return integer] :
function is 10;
       attribute att of func [integer return bit] : function
is 20;
        constant const :integer := func(10)'att;
--ERROR two functions are visible here
      begin
end;
```

Message Details

```
Invalid usage or use of undefined attribute '<attribute>'
```

Severity

Syntax

STX_VH_185

Default generic does not match.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that default generic did not match. The default generic map aspect associates each local generic in the corresponding component instantiation (if any) with a formal of the same simple name. It is an error if such a formal does not exist or if its mode and type are not appropriate for such an association. Any remaining unassociated formals are associated with the actual designator open.

Message Details

```
Default generic does not match : "<generic>"
```

Severity

Syntax

STX_VH_186

Default port does not match.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that default port did not match. The default port map aspect associates each local port in the corresponding component instantiation (if any)with a formal of the same simple name. It is an error if such a formal does not exist or if its mode and type are not appropriate for such an association. Any remaining unassociated formals are associated with the actual designator open.

Message Details

```
Default port does not match : "<port>"
```

Severity

Syntax

STX_VH_187

TIMEOUT/REJECT/AFTER clause should have an expression of predefined type TIME.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it did not find an expression of predefined type TIME in TIMEOUT/REJECT/AFTER clauses. The language requires that the expression following TIMEOUT/REJECT/AFTER reserved words must evaluate to a result of predefined type TIME.

Message Details

```
<TIMEOUT | REJECT | AFTER> clause should have an expression of predefined type TIME
```

Severity

Syntax

STX_VH_188

Both formal and actual cannot be function calls.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it found function calls used for both formal and actual parameters.

Suggested Fix

Do not use function calls for formal and its actual simultaneously.

Message Details

```
Both formal and actual cannot be function calls
```

Severity

Syntax

STX_VH_189

Formal name in association list must not be ambiguous.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it found ambiguous formal name in association list.

Suggested Fix

Do not use ambiguous formal name in an association list.

Message Details

`Ambiguous formal name <name> in association list`

Severity

Syntax

STX_VH_190

Only one parameter is allowed for function call at formal.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of more than one parameter used for function call at formal. The formal part of a named element association may be in the form of function call if and only if the function name denotes a function and takes a single parameter whose type is of the type of formal and function return type is the type of actual.

Message Details

```
Only one parameter allowed for function call at formal
```

Severity

Syntax

STX_VH_191

Index must be locally static for formal.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it did not find a locally static index for formal. The language requires that expression for index for formal must be evaluated during the analysis of design unit in which it appears. For example, the following code will produce this error:

```
package pack is
 type rec1 is
  record
   f1: bit_vector(0 to 0);
   f2: bit_vector(0 to 1);
end record;

type arr is array (0 to 0 ) of rec1;

type rec is
  record
    f1: arr;
    f2: integer;
  end record;
end;

use work.pack.all;
entity ent is
  port(in1: in bit_vector;in2: integer;
      output : out bit_vector);
end;

architecture arch of ent is
  component comp
     port(in1 : rec);
end component;
```

```
  begin
     l1: comp port map (
             in1.f1(0 ).f1(in1'range) => in1,
             in1.f2 => in2);
--Failure here: index in1'range is not locally static for
in1.f1(0).f1(in1'range).
end;
```

Message Details

```
Index "<index>" must be locally static for formal "<formal>"
```

Severity

Syntax

STX_VH_192

Duplicate formal in association.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of duplicate formal in association. Sub elements of an association list may only be assigned once. For example, the following code will produce this error:

```
        package pack is
           type rec_type is
                   record
                     a,b,c : integer;
           end record;
          procedure P1(p : in rec_type; q : in integer; r : out
integer);
           end pack;
```

Message Details

```
Duplicate formal in association
```

Severity

Syntax

STX_VH_193

Access Type not supported.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that access type was not supported in your code.

Message Details

`Access Type not supported`

Severity

Syntax

STX_VH_194

Selector for element association should be static.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it did not find static selector for element association. The language requires that the selector (or expression) for element association in any position in an aggregate must be a locally static name. For example, the following code will produce this error:

```
variable i : integer := 2;
p ( t'(i => 12, others => 0));  --Failure here: i is
                                --not locally static.
```

Message Details

```
Selector for element association at position <position> should
be static
```

Severity

Syntax

STX_VH_195

File Type not supported.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that file type was not supported in your code.

Message Details

`File type is not supported`

Severity

Syntax

STX_VH_196

Alias declaration not supported.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that alias declaration was not supported in your code.

Message Details

```
Alias declaration not supported
```

Severity

Syntax

STX_VH_197

Only signal attribute name can be associated with signal parameter.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it did not find signal attribute name for signal parameter. Only signal attribute name can be associated with signal parameter.

Message Details

```
Only signal attribute name can be associated with signal
parameter <parameter>
```

Severity

Syntax

STX_VH_198

Only one type of operators should be used to combine relations.

Language

VHDL

Rule Description

Message Details

```
Only one type of operators to be used to combine relations
```

Severity

Syntax

STX_VH_199

Invalid combination of operators found

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of invalid combination of operators.

Message Details

```
Operator <operator> cannot be combined with any operator
```

Severity

Syntax

STX_VH_200

OTHERS clause not allowed for unconstrained array.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it found OTHERS clause used for unconstrained array. An OTHERS clause is allowed only in aggregates of constrained array types. Since an OTHERS clause is used to assign values to remaining elements of an array which are left unassigned in previous element associations in an aggregate and hence number of remaining elements must be finite, which is not in the case of unconstrained array.

Message Details

```
OTHERS clause not allowed for unconstrained array
```

Severity

Syntax

STX_VH_201

Object (by context) in expression must be legal.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of illegal object used in an expression. Use legal object as per the context in an expression.

Message Details

```
Invalid object <object> used in expression
```

Severity

Syntax

STX_VH_202

Design units must be current with respect to their dependent units; otherwise require recompile.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of out-of-date design units.

If a library unit has changed (say, by reanalysis of the corresponding design unit), all the library units that are potentially affected by such change become out-of-date and must be reanalyzed before they can be used again. For example, a secondary unit is affected by a change in its corresponding primary unit.

Example

In the following example, the existence of entity `ent` twice makes the architecture out-of-date with respect to the entity.

```
entity ent is --primary unit
port(
    a: in bit
    );
end ent;

architecture arch of ent is --secondary unit
begin
end arch;

entity ent is --primary unit
end ent;
```

Consider another example of a package, as follows:

pack.vhd

```
package pack is --primary unit
```

```
   CONSTANT NUM : NATURAL := 3;
end;
```

Compile the above package in library lib1

The following file uses the library lib1

top.vhd

```
library lib1;
use lib1.pack.all;
entity top is --secondary unit
    port(a : in bit_vector (NUM downto 0));
begin
end top;
```

Compile the file top.vhd and again compile pack.vhd.

Now, an attempt to restore the entity top by using the `set_option top <top-name>` command will give recompile design unit error.

Message Details

```
The <design-unit-type1> '<design-unit-name1>' in library
'<library-name1>' is out-of-date because it has been compiled
earlier than the <design-unit-type2> '<design-unit-name2>' in
library '<library-name2>'; Please recompile it. (<design-unit-
name3> (<unit-string1>) TimeStamp (<time-stamp1>);<design-unit-
name4> (<unit-string2>) TimeStamp (<time-stamp2>))
```

Severity

Syntax

Note

SpyGlass reports the *STX_VH_202* message when the same design is complied twice in a specific sequence of steps, as shown in the following example:

1. Package P1
   Function F1_FROM_P1

2.
```
Use work.p1.all
Use work.p2.all  //Nothing from this package is being
                 //used in E1
Entity E1
```

3.
```
Use work.p1.all
Package P2
Constant : MYCONST : unsigned (31 downto 0) :=
F1_FROM_P1(6, 32);
```

When you compile the above code in the first SpyGlass run, SpyGlass reports the *WRN_384* warning for `work.p2.all`, because it is not present in `work`.

NOTE: *This warning is actually a syntax error as per LRM and all other industry tools report an error in this case.*

If nothing is used from the package in a design unit (`E1` in this case), other industry tools report an error. However, SpyGlass compiles `E1` with a warning.

In the re-entrant flow, when you compile the same code again, `P1` is compiled freshly and `p2` is used from the previous run. Therefore, SpyGlass reports the *STX_VH_202* message to indicate an out of date situation, because the `P2` package uses function from `P1` that was compiled later.

The **solution** to this problem is to clean the WORK directory, and recompile the design.

STX_VH_203

Circular dependency/recursive hierarchy between design units found.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of circular dependency/recursive hierarchy between design units.

For example:

```
entity ee1 is
    port (in1 : in BIT);
end ee1;

architecture aa1 of ee1 is
  component ee2
    port(in2 : in BIT);
  end component;
begin


label_2 : entity WORK.ee2(aa2) port map(in1);
 --ERROR : circular/recursive hierarchy
end aa1;

entity ee2 is
    port (in2 : in BIT);
end ee2;

architecture aa2 of ee2 is
  component ee1
    port(in1 : in BIT);
  end component;

begin
```

```
    label_1 : entity WORK.ee1(aa1) port map(in2);
--ERROR : circular/recursive hierarchy

end aa2;
```

If such recursive hierarchy is intentional, it can be specified within a finite For-Generate statement to avoid this error.

Message Details

```
Circular dependency found (library name=<library-name>, unit
name=<unit-name>, primary unit name=<primary-unit-name>)
```

Severity

Syntax

STX_VH_204

Choice can be a NULL range only if there is a single element association in the aggregate.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of a choice as NULL range even when there were more than one element association in the aggregate. A named association of an array aggregate is allowed to have a choice that is not locally static, or likewise a choice that is a null range, only if the aggregate includes a single element association and this element association has a single choice.

Message Details

```
Choice can be a NULL range only if there is a single element
association in the aggregate
```

Severity

Syntax

STX_VH_205

Invalid REPORT clause is detected for assert statement.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an invalid report clause for assert statement. If the report clause is present, it must include an expression of predefined type STRING that specifies a message to be reported.

Message Details

```
Invalid REPORT clause for assert statement
```

Severity

Syntax

STX_VH_206

Invalid SEVERITY clause is detected for assert statement.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an invalid SEVERITY clause for assert statement. If the severity clause is present, it must specify an expression of predefined type SEVERITY_LEVEL that specifies the severity level of the assertion.

Message Details

```
Invalid SEVERITY clause for assert statement
```

Severity

Syntax

STX_VH_207

Illegal expanded name detected.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of illegal expanded name. An expanded name denotes a primary unit contained in a design library if the prefix denotes the library and the suffix is the simple name of a primary unit whose declaration is contained in that library. An expanded name is not allowed for a secondary unit, particularly for an architecture body. Also the prefix of an expanded name may not be a function call.

Suggested Fix

Use legal expanded name.

Message Details

`Invalid expanded name is found.`

Severity

Syntax

STX_VH_208

Range bounds must be either integer or floating point types (by context).

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it did not find valid range bounds in a context. Range bounds can both be either integer or floating point types. It is an error if one bound is of integer type and other bound is of floating point type.

Message Details

```
Range bounds in this context "<context>" must both be either
integer or floating point types
```

Severity

Syntax

STX_VH_209

Use only expected (by context) index types in expressions.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of wrong index type for expression in which it was used. For example, the following code will produce this error:

```
entity ent is
  port(in1:bit_vector);
end ent;


architecture arch of ent is
   type arr is array(integer range <>)of bit;
   signal sig,sig1:arr(integer);
   signal s:bit;
   begin
     l1:for i in bit generate
        s <= sig1(i);
         --index type does not match
     end generate;
end arch;
```

Suggested Fix

Use valid index type in an expression.

Message Details

```
Wrong index type for expression "<expression>". Expected type
is "<expected-type>"
```

Severity

Syntax

STX_VH_210

Both bounds in the constrained array definition must have the same discrete type.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an index constraint that did not have a range of discrete type. The language requires that both bounds in the constrained array definition must have the same discrete type. It is an error if both bounds of index constraint do not belong to same discrete type.

Message Details

```
Index constraint "<constraint>" must have a range of discrete type
```

Severity

Syntax

STX_VH_211

Objects denoting physical types must be declared as a physical unit.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an object denoting physical type but was not declared as physical unit. A physical type is a numeric scalar type that is used to represent measurements of some quantity. Any value of a physical type is an integral multiple of the primary unit of measurement for that type. For example, the following code will produce this error:

```
package pack is
  constant mm : integer := 10;
  constant const : time := 10 mm;
end;
```

Suggested Fix

Declare an object of a physical type as a physical unit.

Message Details

```
<object> must be declared as a physical unit
```

Severity

Syntax

STX_VH_212

Conditional expression for ASSERT clause must be of the type BOOLEAN.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it did not find BOOLEAN type for a conditional expression for an ASSERT clause. An assertion statement checks that a specified condition is true and reports an error if it is not. Evaluation of an assertion statement consists of evaluation of the BOOLEAN expression specifying the condition. It is an error if this condition is not of type BOOLEAN.

Message Details

```
Conditional expression for ASSERT clause must be of type
BOOLEAN
```

Severity

Syntax

STX_VH_213

Expression with a negative value is not allowed in "TIMEOUT","REJECT" and "AFTER" clauses.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of a negative value for an expression of TIME. Expression that evaluates to a negative value is not allowed in TIMEOUT, REJECT, and AFTER clauses.

Message Details

```
An expression "<expression>" with a negative value '<value>' is
not allowed in "<clause>" clause
```

Severity

Syntax

STX_VH_214

Wait statement is not allowed in a function body.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of `wait` statement inside function body. It is an error if a wait statement appears in a function subprogram or in a procedure that has a parent that is a function subprogram.

Message Details

```
Wait statement is not allowed in a function body
```

Severity

Syntax

STX_VH_216

Non-object aliases are not supported in VHDL-87.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that non-object aliases are not supported in VHDL-87.

Message Details

```
Non-object aliases not supported in VHDL-87
```

Severity

Syntax

STX_VH_217

Exponentiation with a negative exponent is only allowed for a left operand of a floating point type.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of exponentiation with a negative exponent but the left operand was not of a floating point type. The language allows exponentiation with a negative exponent only for a left operand of a floating point type.

Message Details

```
Exponentiation with a negative exponent is only allowed for a
left operand of a floating point type [ "<type>" ]
```

Severity

Syntax

STX_VH_218

Size must match for concatenation operator.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it found size mismatch for concatenation operator. The concatenation operator is predefined for any one-dimensional array type. The left and right operands of a concatenation operator must be either one dimensional arrays of same types or if one operand is any array type, other can be an element type.

In all cases, it is an error if either bound of the index subtype of the result does not belong to the index subtype of the type of the result, unless the result is a null array. It is also an error if any element of the result does not belong to the element subtype of the type of the result.

Message Details

```
Size does not match for concatenation operator
```

Severity

Syntax

STX_VH_219

Invalid concatenation operator detected.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an invalid concatenation operator. The concatenation operator predefined by the language is "&".

Suggested Fix

Use valid concatenation operator as predefined by the language.

Message Details

```
Invalid concatenation operator
```

Severity

Syntax

STX_VH_221

Alias base type must match subtype indication.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of alias base type, which did not match subtype indication. Since an alias declaration declares an alternate name for an existing name entity, hence alias base type must match subtype indication.

For example, an alias of a signal denotes a signal; an alias of a variable denotes a variable; an alias of a constant denotes a constant; and an alias of a file denotes a file. Similarly, an alias of a subprogram (including an operator) denotes a subprogram, an alias of an enumeration literal denotes an enumeration literal and so forth.

Message Details

```
Alias base type "<type>" does not match subtype indication
"<subtype-indication>"
```

Severity

Syntax

STX_VH_222

Number of arguments must be valid (by context).

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an invalid number of arguments by context.

Suggested Fix

Use valid number of arguments as per the context.

Message Details

`Invalid number of arguments for <context>`

Severity

Syntax

STX_VH_223

Invalid type detected.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an invalid type in the context in which it appeared.

Suggested Fix

Use valid type as per the context.

Message Details

```
<type1> is valid only for <type2> types
```

Severity

Syntax

STX_VH_224

Access type is not allowed in constant or signal declarations.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of access type used in constant or signal declaration. An object declared to be of an access type must be an object of class variable. An object designated by an access value is always an object of class variable. Hence it is an error if access type is used in constant or signal declarations.

Message Details

```
Access type is not allowed in <declaration>
```

Severity

Syntax

STX_VH_225

Range Constraint cannot be used in a NEW expression.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of a range constraint used in a NEW expression. The only allowed form of constraint in the subtype indication of an allocator is an index constraint. Hence SpyGlass will flag an error if range constraint is used in NEW expression.

Message Details

```
Range Constraint cannot be used in a NEW expression
```

Severity

Syntax

STX_VH_226

New cannot be used to allocate an object of unconstrained array type: Index Constraint must be specified.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of keyword NEW used to allocate an object of unconstrained array type. If an allocator includes a subtype indication and if the type of the object created is an array, then the subtype indication must either denote a constrained subtype or include an explicit index constraint.

Message Details

```
New cannot be used to allocate an object of unconstrained array
type: Index Constraint must be specified
```

Severity

Syntax

STX_VH_227

Allocator/New expression is not allowed if target is not of access type.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of allocator/New expression used when target was not of access type. Since an allocator creates a new data object of the specified subtype in memory and returns a pointer to it, we must have a target of access type so that the pointer value returned by allocator expression can be assigned to target of a variable assignment statement.

Message Details

```
Allocator/New expression not allowed if target is not of access type
```

Severity

Syntax

STX_VH_228

A subtype indication that is part of an allocator must not include a resolution function.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of a subtype indication that is part of an allocator including a resolution function. A subtype indication that is part of an allocator must not include a resolution function.

Message Details

```
A subtype indication that is part of an allocator must not
include a resolution function
```

Severity

Syntax

STX_VH_229

Design unit referenced not present in logical library.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it could not find design unit in logical library.

Message Details

```
Design Unit <design-unit> not found in logical library
<library>
```

Severity

Syntax

STX_VH_230

No entity class may be defined in the group template list after <>.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an entity class defined in group template list after <>.The language allows to use the box symbol <> after an item class. In fact, you may only include such a class specification once in a template but with a condition that it must be in the last position in the list of item classes.

Message Details

`No entity class may be defined in the group template list after <>`

Severity

Syntax

STX_VH_231

Group template must be declared.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it could not find group template which appeared in group declaration. Group template declaration is a first stage in grouping mechanism to identify a collection of items over which some relationship holds. It lists one or more classes of items, in order, that may constitute a group.

Suggested Fix

First declare group template and then use it in group declaration.

Message Details

`Group Template <template> not declared`

Severity

Syntax

STX_VH_232

Type of group element must match with the corresponding entity class.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of mismatch between the types of group element and corresponding entity class. A group declaration names a template to use for the group and lists the items that are to be members of the group. Each item in the list must be of the class specified in the corresponding position in the template.

Message Details

```
Type of group element <element> does not match with the
corresponding entity class
```

Severity

Syntax

STX_VH_233

Group declaration must have same number of elements as are specified in group template.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of a group declaration which had less number of elements than the number of elements specified in group template declaration. The language requires that number of items listed in group declaration must be equal to the number of classes of items listed in group template declaration.

Message Details

```
Group declaration has less than the number of elements
specified in group template
```

Severity

Syntax

STX_VH_234

Group declaration must have same number of elements as are specified in group template.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of a group declaration which had more number of elements than the number of elements specified in group template declaration. The language requires that number of items listed in group declaration must be equal to the number of classes of items listed in group template declaration

Message Details

```
Group declaration has more than the number of elements
specified in group template
```

Severity

Syntax

STX_VH_235

Subtype indication of a file declaration must define a FILE subtype.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that the subtype indication of a file declaration was not of FILE subtype. The language requires that the subtype indication of a file declaration must define a file subtype.

Message Details

```
Subtype <subtype> used in file declaration not of FILE type
```

Severity

Syntax

STX_VH_236

File name of type STRING is expected.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an identifier (as a file logical name) which was not of type STRING. The file declaration (explicitly declared file) has the following syntax:

```
file_declaration <=
  file identifier{,...}:subtype_indication
    [[open file_open_kind_expression]is
string_expression];
```

The string_expression must represent a file logical name of predefined type STRING. The value of this expression is interpreted as a logical name for a file in the host system environment.

Message Details

`Expecting file name of type STRING`

Severity

Syntax

STX_VH_237

Identifier should be a CONSTANT of type STRING (by context).

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an identifier (as a file logical name) which was not a CONSTANT of type STRING. The file declaration (explicitly declared file) has the following syntax:

file_declaration <=

file identifier{,...}:subtype_indication

[[open file_open_kind_expression]is string_expression];

The string_expression must represent a file logical name of predefined type STRING. The value of this expression is interpreted as a logical name for a file in the host system environment.

Suggested Fix:

Use a CONSTANT identifier of predefined type STRING as a file logical name.

Message Details

`Identifier <identifier> should be a CONSTANT of type STRING`

Severity

Syntax

STX_VH_238

Enumeration value of type file_open_kind is expected.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that it could not find an enumeration value of type file_open_kind. The file declaration (explicitly declared file) has the following syntax:

```
file_declaration <=
  file identifier{,...}:subtype_indication
    [[open file_open_kind_expression]is
string_expression];
```

The optional expression after the keyword open allows us to specify how the physical file associated with the file object should be opened. This expression must have a value of the predefined type file_open_kind, declared in the package standard. The declaration is:

```
type file_open_kind is (read_mode,write_mode,append_mode);.
```

Message Details

```
Expecting an enumeration value of type file_open_kind
```

Severity

Syntax

STX_VH_239

Identifier must be a guarded signal with a resolution function (in context).

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of an identifier that did not denote a guarded signal with a resolution function. For example, the signals appearing in the disconnection specification must be guarded signals. For example, the following code will produce this error:

```
function RESFUNC (S:BIT_VECTOR) return BIT is
 begin
            for I in S'RANGE loop
               if(S(I) = '1')then
                  return '1';
               end if;
            end loop
            return '0';
         end RESFUNC;
     --define the signal.
     subtype RBIT is RESFUNC BIT;
     signal SIG : RBIT;
     --define the disconnection specification
disconnect SIG:RBIT after 0ns;  --Error here.
```

Message Details

```
'<identifier>' must be a guarded signal with a resolution
function
```

Severity

Syntax

STX_VH_240

Disconnection type mark needs to be a predefined type without constraints.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of disconnection type mark, which was not a predefined type without constraints. The language requires that disconnection type mark needs to be a predefined type without constraint.

For example, if the guarded signal is an array element of an explicitly declared signal, the type mark must be the same as the element subtype indication in the (explicit or implicit) array type declaration that declares the base type of the explicitly declared signal. If the guarded signal is a record element of an explicitly declared signal, then the type mark must be the same as the type mark in the element subtype definition of the record type declaration that declares the type of the explicitly declared signal.

Message Details

```
Disconnection type mark "<mark>" needs to be a predefined type
without constraints
```

Severity

Syntax

STX_VH_241

Time expression in a disconnection specification must be valid static time expression.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the absence of a valid static time expression in a disconnection specification. The time expression in a disconnection specification must be static and must evaluate to a non-negative value.

Message Details

```
Expression "<expression>" is not a valid static time expression
for a DISCONNECTION
```

Severity

Syntax

STX_VH_242

DISCONNECTION type specification must match with the guarded signal type.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of mismatch between DISCONNECTION type specification and guarded signal type. The syntax for disconnection specification is:

```
disconnect guarded_signal_list:type_mark after time_expression;
```

If the guarded signal is a declared signal or a slice thereof, the type must be the same as the type mark indicated in the disconnection specification shown above.

Message Details

```
DISCONNECTION type specification "<specification>" does not match with the guarded signal type "<type>"
```

Severity

Syntax

STX_VH_243

Disconnection is supported only for signals without constraints.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of disconnection specification not applied to guarded signals. A disconnection specification defines the time delay to be used in the implicit disconnection of drivers of a guarded signal within a guarded signal assignment. It is an error if disconnection specification is not applied to guarded signals.

Suggested Fix

Use disconnection only for signals without constraints.

Message Details

```
Disconnection supported only for signals without constraints
```

Severity

Syntax

STX_VH_244

Resolution function with the expected parameters (by context) must be present.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of resolution function used for access type or file type in subtype indication. A subtype indication denoting an access type, a file type, or a protected type may not contain a resolution function.

Suggested Fix

Do not use resolution function for subtype indication denoting an access type, a file type or a protected type.

Message Details

`Could not find resolution function <function> with the expected parameters`

Severity

Syntax

STX_VH_245

Illegal command argument specified.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that illegal command arguments are specified.

Message Details

`Too many command arguments specified`

Severity

Syntax

STX_VH_247

.jaguarc not found in CURRENT or HOME directory

Language

VHDL

Rule Description

Message Details

```
.jaguarc not found in CURRENT or HOME directory
```

Severity

Syntax

STX_VH_248

Default logical library WORK must be mapped.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate that default logical library work was not mapped. A VHDL tool suite must provide some means of using a number of separate design libraries. When a design is analyzed, SpyGlass nominates one of the libraries as the working library, and the analyzed design is stored in this library. SpyGlass use the special library name "work" in VHDL models to refer to the current working library. Hence, this default logical library work must be mapped.

Message Details

```
Default logical library work is not mapped
```

Severity

Syntax

STX_VH_250

Selected element denoted by suffix must be declared within the construct denoted by prefix.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of a selected name in an invalid format. A selected name is a name with the format prefix.suffix. You can use a selected name to denote an element of a record, an object designated by an access value, or an entity whose declaration is contained within another named entity, particularly within a library or a package. You can also use a selected name to denote all entities whose declarations are contained in a library or a package. Modify the format of the selected name, which was invalid because of a missing or incorrectly formed use clause; a typo; or incorrect use of a record element.

Message Details

`No selected element named <name> is defined for this prefix`

Severity

Syntax

STX_VH_251

Attribute name couldn't be created.

Language

VHDL

Message Details

Cannot create attribute name <name>

Severity

Syntax

Severity

Syntax

STX_VH_252

Illegal use of character literal in group list is not allowed (by context.

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of illegal use of character literal in group list. A group declaration names a template to use for the group and lists the items that are to be members of the group. Each item in the list (name or character literal) must be of the class specified in the corresponding position in the template. It is an error if a name or character literal does not belong to the class specified in the corresponding position in the template.

Suggested Fix

Use character literal of the class specified in corresponding position in the template.

Message Details

```
Invalid use of character literal in group list
```

Severity

Syntax

STX_VH_253

Invalid name found in group constituent list (by context).

Language

VHDL

Rule Description

SpyGlass generates this error to indicate the presence of invalid name in group constituent list.

Message Details

`Invalid name <name> used in group constituent list`

Severity

Syntax

STX_VH_254

Group element TYPE must match with corresponding entity class.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of type mismatch of group element with corresponding entity class. The language requires that type of group element in group declaration must belong to one of the classes of items declared in group template declaration.

Message Details

```
Type mismatch of group element with corresponding entity class
```

Severity

Syntax

STX_VH_255

Option used is not allowed in VHDL-87 mode.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an option used, which was not allowed in VHDL-87 mode. The options which are not allowed in VHDL-87 mode are: Identifier before null statement, return statement, exit statement, next statement, case statement, if statement, procedure call, variable assignment, signal assignment, assertion statement, wait statement. Also SIGNATURE, Direct entity instantiation, direct configuration instantiation, declaration within generate are not allowed in VHDL-87 mode.

Message Details

```
'<option>' is not allowed in VHDL-87 mode
```

Severity

Syntax

STX_VH_256

Only IN and OUT modes are allowed in file declaration in VHDL-87.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it does not find IN and OUT modes used in file declaration in VHDL-87. Only IN and OUT modes are allowed in file declaration in VHDL-87.

Message Details

```
Only IN and OUT modes are allowed in file declaration in VHDL-87
```

Severity

Syntax

STX_VH_257

Invalid character encountered in based literal.

Language

VHDL

Rule Description

SpyGlass generates this syntax error when it detects invalid character in based-literal. The syntax for based-literal is:

```
Base#based-value#                --form 1
Base#based-value#E exponent      --form 2
```

SpyGlass will flag an error if based-value in based-literal has any character that does not conform to the base (any value between 2 and 16) specified in based-literal.

Message Details

```
Invalid character "<character>" in based literal "<literal>"
```

Severity

Syntax

STX_VH_258

Invalid character encountered in bit string literal.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an unexpected character in bit string literal. VHDL specifies that various items, such as literals, consist of a series of specific characters. If, while reading one of these items, a character that does not fall into this set is encountered, this error message is generated. For example, if the base specifier is 'O', the extended digits in the bit value are restricted to legal digits in the octal number system, that is, the digits 0 through 7. Similarly when the base specifier is 'B', only bit values are allowed in bit string literal.

Message Details

```
Invalid character "<character>" in bit string literal <literal>
```

Severity

Syntax

STX_VH_259

Guarded signal assignment can only occur either within a guarded block or with an explicitly declared signal called GUARD of type Boolean.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of guarded signal assignment neither within a guarded block nor with an explicitly declared signal called GUARD of type Boolean.

Message Details

```
A guarded signal assignment can only occur either within a
guarded block or with an explicitly declared signal called
GUARD of type Boolean
```

Severity

Syntax

STX_VH_260

More than one DISCONNECTION specification cannot be applied to the same signal.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of DISCONNECTION specification applied more than once to the same signal.

Suggested Fix

Do not apply DISCONNECTION specification more than once to the same signal.

Message Details

```
DISCONNECTION specification for signal "<signal>" has already
been specified in file '<file-name>' at line <line-num>
```

Severity

Syntax

STX_VH_262

Illegal association element encountered.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an illegal element association. VHDL requires that expression of each element association in an aggregate must result in a type, which should match its target type.

Suggested Fix

Use legal association element.

Message Details

`Invalid association element <element>`

Severity

Syntax

STX_VH_263

Label name should be an identifier.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it does not find an identifier for label name.

Suggested Fix

Use identifier for label name.

Message Details

```
Label name should be an identifier
```

Severity

Syntax

STX_VH_264

Use valid prefix (by context) for an attribute.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an invalid prefix for an attribute. The prefix used for an attribute must be the same prefix given in the definition of that pre-defined attribute.

Suggested Fix

Check the definition of attribute in LRM and then use appropriate prefix.

Message Details

`Invalid prefix for attribute <attribute>`

Severity

Syntax

STX_VH_266

Port must be specified only once in a port map.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a port that is specified more than once in a port map. VHDL requires that a port should be specified only once in a port map.

Suggested Fix

Specify each port only once in a port map.

Message Details

```
Port "<port-name>" is specified more than once in this port map
```

Severity

Syntax

STX_VH_267

Formal must be present corresponding to each actual.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it does not find a formal corresponding to an actual parameter. VHDL requires that the number of formals must exactly be same as the number of actuals in an interface list.

Suggested Fix

Specify missing formal corresponding to an actual parameter.

Message Details

```
Mismatch in number of <PORTS | GENERICS> defined for instance
(<association-list-length1>) as compared to <ENTITY |
COMPONENT> <block-name> block name declaration (<association-
list-length2>) at line <line-num> of file <file-name>
```

Severity

Syntax

STX_VH_268

Actual specified for formal must be valid (in context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an invalid actual specified for formal. VHDL requires that the type of actual must match the type of corresponding formal.

Suggested Fix

Specify valid actual for formal in your code.

Message Details

```
Invalid actual specified for formal <formal>
```

Severity

Syntax

STX_VH_269

Type of an actual must be same as that of the formal.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a mismatch in the types of actual and its corresponding formal. VHDL requires that the type of actual must match the type of its corresponding formal. Any mismatch in the types of actual and its formal parameter will result in generation of this error.

For example, the following code will generate this syntax error:

```
entity ent is
end;
architecture arch of ent is
    component comp is
        port(cpin1: out bit;
            cpin2: in bit);
end component;
function func(fin1:bit_vector(0 to 1)) return bit is
begin
    return '1';
end;

signal sig : bit;
begin
l1: comp port map
    ( func(cpin1) =>sig,
        cpin2=>'1');
```

```
end;
```

Message Details

```
Mismatch of type between '<variable-name>' ('<variable-type>')
and corresponding <PORT | PARAM> '<parameter-name>'
('<type-name>') of <COMPONENT | ENTITY | BLOCK | PROCEDURE |
FUNCTION> '<name>' at line <line-num> of file <file-name>
```

Severity

Syntax

STX_VH_271

Subprogram body header and corresponding declaration must match.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of mismatch between subprogram body header and its declaration. It is an error if subprogram specification that specifies the name of a subprogram and defines its interface, including the formal parameter names, their class (signal, variable, constant, or file), their type, their mode (in, out, or inout) does not match with subprogram declaration

Message Details

```
Subprogram "<sub-program>" body header and corresponding
declaration do not match
```

Severity

Syntax

STX_VH_272

Array subscript must not go out of Bounds.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of array subscript out of bounds. It is an error if array subscript exceeds the range of array specified in the declaration of that array.

Message Details

```
Array "<array>" subscript "<subscript>" out of Bounds "<range>"
```

Severity

Syntax

STX_VH_273

Labels are not permitted as primaries in an expression.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of illegal use of label in expression. The identifier, which can be used in an expression, must be a name, a literal, an aggregate, a function call, a qualified expression, a type conversion, an allocator, or an expression enclosed with parenthesis. The language does not allow use of label in an expression.

For example, the following code will produce this error:

```
type small_int is range 0 to 7;
type cmd_bus is array (small_int range <>) of small_int;
signal ibus: cmd_bus(small_int);
signal s_int: small_int;
begin
  test_1:process
    s_int <= ibus'right(small_int(test_1)) after 5 ns;
    -- process labels illegal here
  END process
END.
```

Message Details

```
Invalid use of label <label> in expression
```

Severity

Syntax

STX_VH_274

VHDL-93 does not allow use of OPEN as an actual for individual sub-element association.

Language

VHDL

Rule Description

This error occurs because SpyGlass finds usage of the OPEN keyword as an actual for individual sub-element association. VHDL-93 does not allow use of OPEN as an actual for individual sub-element association.

Message Details

```
VHDL-93 does not allow use of OPEN as an actual for individual
sub-element association
```

Severity

Syntax

STX_VH_275

A signature must not appear in an object alias declaration.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of signature in object alias declaration. An object alias is an alias whose alias designator denotes an object (that is, a constant, a variable, a signal or a file). An alias declaration has the following form:

```
alias_declaration ::=

      alias alias_designator [:subtype_indication ] is name
[signature];
```

VHDL requires that a signature should not appear in a declaration of an object alias. Hence, it is an error if signature is included in declaration of object alias.

Message Details

```
A signature may not appear in an object alias declaration
```

Severity

Syntax

STX_VH_276

Non-object alias declaration cannot have a subtype indication.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of subtype indication in non-object alias declaration. An alias declaration has the following form:

```
alias_declaration ::=

  alias alias_designator [:subtype_indication ] is name [signature];
```

A nonobject alias is an alias whose alias designator denotes some named entity other than an object. VHDKL requires that a subtype indication should not appear in a nonobject alias. Hence, it is an error if subtype indication is included in declaration of nonobject alias.

Message Details

```
A non-object alias declaration cannot have a subtype indication
```

Severity

Syntax

STX_VH_277

Aliased name corresponding to signature couldn't be found.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not find aliased name corresponding to signature used in alias declaration.

When we declare an alias for a subprogram, the signature identifies which overloaded version of the subprogram name is aliased. The signature lists the types of each of the subprogram's parameters in the same order that they appear in the subprogram's declaration. However it is an error if signature parameter list does not match with all of the overloaded versions of the subprogram's parameters.

Similarly, in case of enumeration literal it is an error if signature fails to distinguish one particular meaning of overloaded enumeration literals. For example, the following code will produce this error:

```
package pack is
 procedure func (in1,in2 : bit);
 procedure func (in1,in2 :integer);
 procedure func (in1,in2 : character);
 procedure func (in1,in2 : bit_vector);
 alias a1 is func [integer,bit];    --Error here:signature
[integer,bit] does not match any of subprogram parameters.
end;
```

Message Details

```
Cannot find aliased name corresponding to signature
```

Severity

Syntax

STX_VH_278

Signature not allowed in alias declaration (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of signature which was not allowed in an alias declaration. We can declare aliases for other named items that do not represent stored data, such as types, subprograms.packages, and generate parameters. The syntax rule for alias declarations for non data items is:

```
alias_declaration <=
alias (identifier | character_literal | operator_symbol)
is name [signature];
```

we can use character literals as aliases for enumeration literals, and operator symbols as aliases for function subprograms. The optional signature part in an alias declaration is only used in aliases for subprograms and enumeration literals.

Message Details

```
Signature not allowed in this alias declaration
```

Severity

Syntax

STX_VH_279

POSTPONED keyword not allowed for a process which was not declared postponed.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a process not declared postponed but postponed keyword was used to end that process. A process is made postponed by including the keyword postponed as shown by the full syntax rule for a process:

```
process_statement<=
 [process_label:]
 [postponed] process [( signal_name{,..})][is]
   {process_declarative_item}
 begin
   {sequential_statement}
 end [postponed] process [process_label];
 (Items in [ ] are optional)
```

It is an error if postponed keyword is present at the end of process (when the process is ended) but not present at the start of the process.

Message Details

```
POSTPONED keyword not allowed for a process which was not
declared postponed
```

Severity

Syntax

STX_VH_280

Label must be associated with a statement before it can be declared.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a label declared that is not associated with any statement.

Message Details

```
Label <label> declared but not associated with any statement
```

Severity

Syntax

STX_VH_282

Library WORK already contains a design unit by the name specified.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of duplicate primary unit names in the same library. Each primary unit in a given library must have a simple name that is unique within the given library, and each architecture body associated with a given entity declaration must have a simple name that is unique within the set of names of the architecture bodies associated with that entity declaration.

Message Details

```
Library WORK already contains a design unit by the name <name>
```

Severity

Syntax

STX_VH_284

Range Bounds in a physical type definition must be of integer type and locally static.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the presence of range bounds in a physical type definition which were not integer type and locally static. The language requires that each bound of a range constraint that is used in a physical type definition must be a locally static expression of some integer type, but the two bounds need not have the same integer type. (Negative bounds are allowed).

Message Details

```
Range Bounds in a physical type definition must be of integer
type and locally static
```

Severity

Syntax

STX_VH_285

Declaration of unit for physical type expected.

Language

VHDL

Rule Description

This error occurred because SpyGlass could not find the declaration of a unit for a physical type used in your code. The language requires that unit must first be declared before usage. Hence the following code will produce this error:

```
  ENTITY ent IS
  END ent;
  ARCHITECTURE arch OF ent IS
       type time is range 0 to 1E8 units
           fs;
           ps = 10 min;--Failure_here: min not defined
        end units;
  BEGIN
END arch
```

Message Details

Unit <unit> has not been previously declared for this physical type

Severity

Syntax

STX_VH_286

Language

VHDL

Rule Description

This error occurs when SpyGlass does not find integer literal multiplier for a secondary unit. The language requires that Unit names declared in secondary unit declarations must be directly or indirectly defined in terms of integral multiples of the primary unit of the type declaration in which they appear. The abstract literal portion (if present) of a physical literal appearing in a secondary unit declaration must be an integer literal. Hence the following code will produce this error:

```
ENTITY ent IS
END ent;
ARCHITECTURE arch OF ent IS
    type J is       -- physical type decl
        range 0 to 1000
        units
                A;
                B = 10 A;
                C = 10.1 B;     -- Failure_here
                D = 10 C;
        end units;
    BEGIN
END arch;
```

Message Details

```
The multiplier for a secondary unit must be an integer literal
```

Severity

Syntax

STX_VH_287

Signals cannot be declared in a subprogram body.(VHDL)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that signals were declared in a subprogram body. Subprogram declarative items include type declaration, subtype declaration, constant declaration, variable declaration, file declaration, alias declaration, attribute declaration, attribute specification, use clause, group template declaration and group declaration. The subprogram declarative part, if not empty, must consist of the above subprogram declarative items. However, it is illegal to declare signals within subprogram declarations.

Message Details

```
Signals cannot be declared in a subprogram body
```

Severity

Syntax

STX_VH_288

Subprogram body not allowed in package declarative region.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of subprogram body in package declarative region. In package declarative region, subprogram declarations are allowed and bodies of these subprograms are included in corresponding package body. It is an error if subprogram body is included in package declarative region.

Message Details

```
Subprogram body not allowed in package declarative region
```

Severity

Syntax

STX_VH_289

Only a simple identifier can be used to define a Procedure name.

Language

VHDL

Rule Description

Message Details

```
Only a simple identifier can be used to define a Procedure name
```

Severity

Syntax

STX_VH_290

Illegal use of incomplete type encountered before its actual definition.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of illegal use of incomplete type before its actual definition. An incomplete type declaration has the form:

```
type type-name;
```

Once an incomplete type has been declared, the type-name can now be used in any mutually dependent or recursive access type and a corresponding full type declaration must follow later. However it is an error if an incomplete type is used to declare an object of its own type before its full type declaration. Hence the following code will produce this error:

```
   entity ent is
   end;
   architecture arch of ent is
     type inc_type;
     attribute att : inc_type;
     begin
end;
```

Message Details

```
Invalid use of incomplete type <type> encountered before its
actual definition
```

Severity

Syntax

STX_VH_291

Subtype indication given in the full declaration must conform to that given in the deferred constant declaration.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that subtype of a deferred constant did not conform to that given in full declaration. If a given package declaration contains a deferred constant declaration, then a constant declaration with same identifier must appear as a declarative item in the corresponding package body. Also, the subtype indication given in the full declaration must conform to that given in the deferred constant declaration.

Message Details

`Subtype Indication mismatch for deferred constant <constant>`

Severity

Syntax

STX_VH_292

Deferred constant must have corresponding full declaration in package body

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find full declaration of deferred constant in the package body. If a given package declaration contains a deferred constant declaration, then a constant declaration with same identifier must appear as a declarative item in the corresponding package body. Also, the subtype indication given in the full declaration must conform to that given in the deferred constant declaration.

Message Details

```
Deferred constant <constant> not initialized in package or in package body
```

Severity

Syntax

STX_VH_293

File type is not allowed in signal or constant or variable declarations.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of file type used in signal or constant or variable declarations.

The following are some points about signal or constant or variable declaration:

- It is an error if a signal declaration declares a signal that is of a file type, an access type, a protected type, or a composite type having a sub element that is a file type, an access type, or a protected type.
- It is an error if a variable declaration declares a variable that is a file type.
- It is an error if a constant declaration declares a constant that is of a file type, an access type, a protected type, or a composite type that has a subelement that is a file type, an access type, or a protected type.

Message Details

```
File Type not allowed in signal or constant or variable declarations
```

Severity

Syntax

STX_VH_294

Element of an array cannot be of FILE type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of file type used as an element of array (composite type). A composite type (that is, arrays of values and record of values) may only contain elements that are of scalar, composite, or access type. Elements of file types are not allowed in a composite type.

Message Details

```
Element of an array cannot be of FILE type
```

Severity

Syntax

STX_VH_295

Element of a record cannot be of a FILE type (in context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of file type used as an element of record (composite type). A composite type (that is, arrays of values and record of values) may only contain elements that are of scalar, composite, or access type. Elements of file types are not allowed in a composite type.

Message Details

```
Element <element> of record cannot be of a FILE type
```

Severity

Syntax

STX_VH_296

Element type of a file cannot be an access or FILE type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of access type or file type as an element of a file type. A file can only contain one type of object, but that type can be almost any VHDL type, including scalar types, records and one-dimensional arrays. The only types that cannot be stored in files are multidimensional arrays, access types, protected types and other files.

Message Details

```
Element type of a file cannot be an access or FILE type
```

Severity

Syntax

STX_VH_297

Only one-dimensional array is allowed as an element of a file type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find one dimensional array as an element of a file type. A file can only contain one type of object, but that type can be almost any VHDL type, including scalar types, records and one-dimensional arrays. The only types that cannot be stored in files are multidimensional arrays, access types, protected types and other files.

Message Details

```
Only one-dimensional array is allowed as element of a file type
```

Severity

Syntax

STX_VH_298

Type designated in an access type definition cannot be a file type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of file type designation in an access type. Values belonging to an access type are pointers to dynamically allocated objects of some other type. In general we can write access type declarations referring to any VHDL type except file types or protected types.

Message Details

```
The type designated in an access type definition cannot be a
file type
```

Severity

Syntax

STX_VH_299

Undeclared Attribute found.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not find declaration of an attribute used in your code. An attribute must first be associated with a named entity before it can be used. First declare an attribute using the following attribute declaration:

```
attribute_declaration::=
        attribute identifier:type_mark;
```

`type_mark` denotes the named entity with which an attribute is desired to be associated. It is an error if an attribute is used before its declaration.

Message Details

`<declaration> has not been previously declared as an Attribute`

Severity

Syntax

STX_VH_300

Label used to denote a component instance must first be declared as an instance of component.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a label used to denote a component instance but did not appear in corresponding component instantiation statement(s). This error might occur if you are using a label to denote a component instance in configuration declaration for late binding of that component, but that label does not appear in component instantiation statement(s) in architecture body for which that configuration declaration is written.

Message Details

```
Label <label> does not denote an instance of component
<component>
```

Severity

Syntax

STX_VH_301

Component instances must be configured only once in a configuration.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of multiple configurations for a component label in a configuration specification. The elaboration of a configuration specification results in the association of binding information with the labels identified by the instantiation list. A label that has binding information associated with it is said to be bound. It is an error if the elaboration of a configuration specification results in the association of binding information with a component label that is already bound.

Message Details

```
Component instance "<instance>" already has a configuration
specified in file '<file-name>' at line <line-num>
```

Severity

Syntax

STX_VH_302

Attribute for a signal must be specified only once.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of attribute specification of an attribute more than once in your code. It is an error if a given attribute is associated more than once with a given named entity. The language does not allow overloading of attributes.

Message Details

```
Attribute "<attribute>" has already been specified for signal
"<signal-name>"
```

Severity

Syntax

STX_VH_303

User defined attribute cannot be declared for ACCESS type and FILE type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of user defined attributes declared for ACCESS type or FILE type. User defined attributes are constants of any type except access or file type. It is an error if user defined attribute are declared for access type, a file type, a protected type, or a composite type with a subelement that is an access type, a file type, or a protected type.

Message Details

```
User-defined attribute cannot be declared for ACCESS type and
FILE type
```

Severity

Syntax

STX_VH_304

Port and port map defined in block header must match.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of mismatch between the port and port map defined in block header. Each element in the association list associates one port of the entity either with one signal of the enclosing architecture body or with value of an expression, or leaves the port unassociated using keyword `open`. It is an error if mismatch is found either because of unequal number of entries or because of different types of corresponding entries in port and port map.

Message Details

`Port and Port map defined in block header do not match`

Severity

Syntax

STX_VH_305

Generic and Generic map defined in block header must match.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of mismatch between the generic and generic map defined in the block header.

Message Details

```
Generic and Generic map defined in block header do not match
```

Severity

Syntax

STX_VH_306

Sequence of transactions must be described in ascending order of time.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that a sequence of transactions was not described in ascending order of time.

Suggested Fix

Define sequence of transactions in ascending order of time.

Message Details

```
Sequence of transactions must be described in ascending order of time
```

Severity

Syntax

STX_VH_307

NULL transaction can be assigned only to GUARDed signals.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of NULL transaction that was not assigned to GUARDed signals. A process can disconnect a driver for a guarded signal by specifying a null transaction in a signal assignment statement. A syntax rule for null transactions is:

```
waveform <=(value_expression[after time-expression] | null
[after time_expression]){,..}
```

This rule indicates that instead of specifying a value in a transaction, we can use the keyword `null` to indicate that the driver should be disconnected after the given delay. When this null transaction matures, the driver ceases to contribute values to resolution function used to compute signal's value. Hence, the size of array of values passed as an argument to the resolution function is reduced by one for each driver that currently has a null transaction determining its contribution.

Message Details

```
NULL transaction can be assigned only to GUARDed signals
```

Severity

Syntax

STX_VH_308

Default value specified in a generic must be locally static.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find locally static default value specified in a generic. If a default expression is specified for a generic, the value of this expression is the value of the generic. This value must be evaluated during the analysis of the design unit in which it appears.

Message Details

```
The default value specified in a generic must be locally static
```

Severity

Syntax

STX_VH_309

Integer overflow detected. Allowable range is -2147483648 to 2147483647.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of maximum system integer overflow in the code. The allowable range is -2147483648 to 2147483648. It is an error if any integer in your code exceeds this range.

Message Details

```
Maximum system integer overflow - allowable range is -
2147483648 to 2147483647
```

Severity

Syntax

STX_VH_310

Bounds of a range must be of the same discrete type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it does not find same discrete types for bounds of a range. The language requires that bounds of a range must be of same enumeration or an integer type. If this condition is satisfied, then a range is known as discrete range.

Message Details

```
Bounds of a range must be of the same discrete type
```

Severity

Syntax

STX_VH_311

Parameter name must match with its corresponding parameter in declaration.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of mismatch between the names of a parameter in subprogram body and subprogram declaration. The language requires that parameter names must match in the declaration and body of same subprogram. For example, the following code will produce this error:

```
entity ent is
end;
architecture arch of ent is
     procedure proc ( in1,in2: out bit);
     procedure proc (in3,in4:out bit) is
      --Error:in3,in4 do not match in1,in2 in decl
     begin
     end;
     begin
end;
```

Message Details

```
Parameter name <parameter1> does not match with corresponding
parameter <parameter2> in declaration
```

Severity

Syntax

STX_VH_312

Parameter of specified kind must match with its declaration.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of mismatch between the kinds of parameter appeared where it is used and its declaration. For example, this error will be generated when there is a mismatch between the kinds of a parameter in subprogram body and subprogram declaration. Hence the following code will produce this error:

```
  entity ent is
  end;
  architecture arch of ent is
     procedure proc (signal in1,in2: out integer);
          --here in1,in2 are of signal class.
     procedure proc (in1,in2: out integer) is
          --here in1,in2 are of variable class.
     begin
     end;
     begin
end;
```

Message Details

```
Parameter kind of <parameter-type> does not match with
declaration
```

Severity

Syntax

STX_VH_313

Port type of a parameter (by context) must match with declaration.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the port type of a parameter did not match with its declaration. It is an error if the mode (in, out, inout) of parameter does not conform to the mode declared in its declaration.

Message Details

```
Port type of parameter <parameter> does not match with
declaration
```

Severity

Syntax

STX_VH_314

Illegal declaration (by context) detected in an expression.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an illegal declaration in an expression.

Suggested Fix

Use legal declaration in an expression by context.

Message Details

`Cannot use declaration <declaration> in an expression`

Severity

Syntax

STX_VH_315

Index type of unconstrained array must denote a discrete type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that an index type of unconstrained array did not denote a discrete type. The index type of unconstrained array must be an enumeration type or an integer type. Each value of an index type of unconstrained array must have a position number that should be an integer value.

Message Details

```
Index type <type> of unconstrained array does not denote a
discrete type
```

Severity

Syntax

STX_VH_316

Full type definition for incomplete type must be specified.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not find full type definition for incomplete type used in your code. For each incomplete type declaration there must be a corresponding full type declaration with the same identifier. This full type declaration must occur later and immediately within the same declarative part as the incomplete type declaration to which it corresponds.

Message Details

```
Missing full type definition for incomplete type <type>
```

Severity

Syntax

STX_VH_317

Return type of a function must match with function declaration type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of mismatch between return type of a function and return type declared in function declaration. All functions must have a return statement with an expression and the type of the result when this expression is evaluated must match the type appearing after the keyword return in function declaration.

Message Details

```
Type mismatch between function name "<function-name>" return
type "<return-type>" and declaration type "<declaration-type>"
```

Severity

Syntax

STX_VH_318

Value should not be out of range (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a value which was out of range as per the context in which it appeared. It is an error if the variable is assigned a value, which is beyond the maximum allowable limit of base type of that variable.

Message Details

`Value <value> is out of range <range>`

Severity

Syntax

STX_VH_319

Bit string delimiters must be identical.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find identical bit string delimiters in your code. A delimiter is either one of the following special characters (in the basic character set):

```
& '() * +  , - . / : ; < = > | []
```

or

one of the following compound delimiters, each composed of two adjacent special characters:

```
=> ** := /= >= <= <>
```

Each of the special characters listed for single character delimiter is a single delimiter except if this character is used as a character of a compound delimiter or as a character of a comment, string literal, character literal, or abstract literal.

Message Details

```
Bit string delimiters must be identical
```

Severity

Syntax

STX_VH_320

Disconnect specifications must be specified only once for a signal of particular type in a scope.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of disconnect specification more than once for a signal in your VHDL code. It is an error if more than one disconnection specification applies to drivers of the same signal.

Message Details

```
No more disconnect specifications allowed in this scope for
signals of type <type>
```

Severity

Syntax

STX_VH_321

Signals of a particular type cannot be declared after the OTHERS/ALL disconnect specification of that type.(VHDL)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of signals of a particular type declared after the OTHERS/ALL disconnect specification. A disconnection specification with the signal list others or all for a given type that appears in a declarative part must be the last such specification for the given type in that declarative part. No guarded signal of the given type may be declared in a given declarative part following such a disconnection specification.

Message Details

```
Signals of type <type> cannot be declared after the OTHERS/ALL
disconnect specification
```

Severity

Syntax

STX_VH_322

Constraint on parameter (by context) must match with declaration.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found mismatch between constraint on parameter and its declaration.

Suggested Fix

Use those constraints on parameter, which conform to parameter declaration.

Message Details

```
Constraint on parameter <parameter> does not match with
declaration
```

Severity

Syntax

STX_VH_323

Use expected type (by context) in range expression.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find an expected type in range expression. The language requires that bounds of discrete ranges must be of same type.

Message Details

`Type error in range expression - expecting type <type>`

Severity

Syntax

STX_VH_324

Actual must be specified for each formal signal.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find an actual signal for a formal signal. VHDL requires that each formal signal must have an actual signal associated with it. Hence, it is an error if the mismatch is found between the number of formal and actual signals.

Message Details

```
Actual for formal signal <signal> must be specified
```

Severity

Syntax

STX_VH_325

Object GUARD cannot be declared as anything other than a signal.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an object GUARD that was not declared a signal. It is an error if the object GUARD is not declared as signal because the option `guarded` specifies that the signal assignment statement is executed when a signal GUARD changes from FALSE to TRUE (that is when an event occurs on it) or when that signal has been TRUE and an event occurs on one of the signal assignment statement's input.

Message Details

```
Object GUARD cannot be declared as anything other than a signal
```

Severity

Syntax

STX_VH_326

Explicitly declared GUARD signal must be of type BOOLEAN.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that an explicitly declared GUARD signal is not of type BOOLEAN. Guarded signals may also be assigned values where a signal GUARD has been explicitly declared. VHDL requires that explicitly declared GUARD signal must be of type BOOLEAN that is visible at the point of the concurrent signal assignment statement.

Message Details

```
Explicitly declared signal GUARD must be of type boolean
```

Severity

Syntax

STX_VH_327

Integer values must have a non-negative exponent.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of negative exponent for integer values. Exponentiation with an integer exponent is equivalent to repeated multiplication of the left operand by itself for a number of times indicated by the absolute value of the exponent and from left to right; if the exponent is negative, then the result is the reciprocal of that obtained with the absolute value of the exponent. Exponentiation with a negative exponent is only allowed for a left operand of a floating point type.

Message Details

```
Integer values must have a non-negative exponent
```

Severity

Syntax

STX_VH_328

Array index must denote a discrete type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of array index, which did not denote a discrete type. A discrete type is an enumeration type or an integer type. Each value of a discrete type has a position number that is an integer value. The name for an element of an array must use one or more index values belonging to specified discrete types.

Message Details

```
Array index must denote a discrete type
```

Severity

Syntax

STX_VH_329

Type mark in index constraint must not denote a constrained type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of type mark in index constraint, which did not denote a constrained type.

Message Details

```
Type mark in index constraint must not denote a constrained
type
```

Severity

Syntax

STX_VH_330

Signal assignment must be guarded if target is a guarded signal.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of target of signal assignment statement as guarded but signal assignment was not guarded. If the target of a concurrent signal assignment is a name that denotes a guarded signal or if it is in the form of an aggregate and the expression in each element association of the aggregate is a static signal name denoting a guarded signal, then the signal assignment must be guarded.

Message Details

```
Target is a guarded signal : hence signal assignment must be guarded
```

Severity

Syntax

STX_VH_331

Signal assignment target cannot contain both guarded and unguarded signals.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of both guarded and unguarded signals in a signal assignment target. The target of signal assignment statement must either be a guarded signal (may be its aggregate also) or an unguarded signal (may be its aggregate also) but not both. Also it is an error if the target of signal assignment is neither a guarded target nor an unguarded target.

Message Details

```
Signal assignment target cannot contain both guarded and
unguarded signals
```

Severity

Syntax

STX_VH_332

Procedure with wait statement cannot be invoked from a process with sensitivity list.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a procedure with wait statement, which was invoked from a process with sensitivity list. It is an error if a wait statement appears in an explicit process statement that includes a sensitivity list or in a procedure that has a parent that is such a process statement.

Message Details

```
Procedure with wait statement cannot be invoked from a process
with sensitivity list
```

Severity

Syntax

STX_VH_333

Procedure with wait statement cannot be invoked from a function.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a procedure with wait statement, which was invoked from function. It is an error if a wait statement appears in a function subprogram or in a procedure that has a parent that is a function subprogram. It is an error if a wait statement appears within any subprogram whose body is declared within a protected type body, or within any subprogram that has an ancestor whose body is declared within a protected type body.

Message Details

```
Procedure with wait statement cannot be invoked from a function
```

Severity

Syntax

STX_VH_334

Expanded name must be visible in the scope of construct (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an expanded name used outside the named construct. If the name of the construct appears in an expanded name, then that expanded name must appear within that named construct.

For example, the following code will produce this error:

```
    package pkg is
       procedure check (x: in integer; y: in boolean);
    end pkg;
    use work.ch0603_p01002_01_pkg.all;
    ENTITY ent IS
    END ent;

    ARCHITECTURE arch OF ent IS
        constant p: integer := 3;
        constant q: boolean := true;
        BEGIN
        TESTING: PROCESS
                variable p: integer;
                variable q: boolean;
            BEGIN

        END PROCESS TESTING;
check (TESTING.p, TESTING.q);     -- Failure_here
END arch;
```

Message Details

```
Expanded name visible only in the construct <construct>
```

Severity

Syntax

STX_VH_335

Only value attribute name can be associated with constant parameter.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find the association of value attribute name with constant parameter. Only value attributes can be associated with constant parameter because these attributes return a constant value.

Following are the value attributes defined by VHDL:

- If T is any scalar type or subtype, then

  T'LEFT,T'RIGHT,T'HIGH,T'LOW,T'ASCENDING are value attributes.

- If A is a constrained array object and N is the dimension, then

  A'LENGTH(N),A'ASCENDING(N)are value attributes.

- If E is any name other than a local port or a local generic of a component declaration, then

E'INSTANCE_NAME,E'PATH_NAME are value attributes.

Message Details

```
Only value attribute name can be associated with constant
parameter <parameter>
```

Severity

Syntax

STX_VH_336

Attribute name cannot be associated with variable parameter.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not associate attribute name with variable parameter.

Message Details

```
Attribute name cannot be associated with variable parameter
<parameter>
```

Severity

Syntax

STX_VH_337

Prefix of a name cannot be a formal parameter of mode out if the prefix is an access type.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of formal parameter of mode OUT used as prefix, which was of access type. If the type of a prefix is an access type, then the prefix must not be a name that denotes a formal parameter of mode OUT, or a subelement thereof.

For example, the following code will produce this error:

```
      type z is
        record
            y   : integer;
            p,q : boolean;
        end record;
    type ptrtype is access z;
    procedure P ( x : out Ptrtype) is
     begin
      x.y := 1;          -- The prefix is of access type
                         -- of which denotes a formal
                         -- parameter of mode
end;
```

Message Details

```
Formal parameter <parameter> of mode OUT cannot be used as
prefix since it is an access type
```

Severity

Syntax

STX_VH_338

Index specification may be applied only on a for-scheme generate statement.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of index specification not applied on a for-scheme generate statement. For a generate statement with a for generation scheme, the generate parameter specification is the declaration of the generate parameter with the given identifier. The generate parameter is a constant object whose type is the base type of the discrete range of the generate parameter specification.

Message Details

```
Index specification may be applied only on a for-scheme
generate statement
```

Severity

Syntax

STX_VH_339

Object in the group declaration must denote a group template.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a named entity (in group declaration) that did not denote a group template. A group declaration declares a group, a named collection of named entities. It is an error if the class of any group constituent in the group constituent list is not the same as the class specified by the corresponding entity class entry in the entity class entry list of the group template.

Message Details

```
<named-entity> in the group declaration does not denote a group template
```

Severity

Syntax

STX_VH_340

Default value is not permitted on physical literals.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that a default value was used for physical literals. The language does not permit the usage of default values for physical literals.

Message Details

```
Unknown value : default value not permitted on physical
literals
```

Severity

Syntax

STX_VH_341

Formal may not be associated with a discrete range as actual.

Language

VHDL

Rule Description

Message Details

```
Formal <formal> may not be associated with a discrete range as
actual
```

Severity

Syntax

STX_VH_342

Use correct index constraint in subtype indication.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of invalid index constraint in subtype indication. An index constraint is compatible with an array type if and only if the constraint defined by each discrete range in the index constraint is compatible with the corresponding index subtype in the array type. An array value satisfies an index constraint if the array value and the index constraint have the same index range at each index position.

For example, the following code will produce this error:

```
package pack is
  subtype sub is bit_vector(0 to 1);
  signal sig: bit_vector(sub) ; -- type sub needs to be
                                -- scalar type here.
end;
```

Suggested Fix

Use valid index constraint in subtype indication.

Message Details

```
Invalid index constraint in subtype indication
```

Severity

Syntax

STX_VH_343

Attribute parameter must have a non-negative value.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of negative value of an attribute parameter. Various types of attribute parameters are base type of any scalar type or subtype, an expression of type string, base type of any discrete or physical type or subtype, a locally static expression of type universal_integer, a static expression of type TIME that evaluates to a non-negative value.

For example, this error will be generated if parameter T (which should be static expression of type TIME) in predefined attribute S'STABLE[(T)](where S is any static signal name) is negative.

Message Details

```
Attribute parameter must have a non-negative value
```

Severity

Syntax

STX_VH_344

Prefix for attribute 'driving and 'driving_value cannot be an input or linkage port.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it finds a port of mode input or linkage as a prefix for attribute 'driving and 'driving_value. The 'driving and 'driving_value attributes are available only from within a process, a concurrent statement with an equivalent process, or a subprogram. If the prefix denotes a port, it is an error if the port does not have a mode of inout, out or buffer.

Message Details

```
Prefix for attribute 'driving and 'driving_value cannot be an input or linkage port
```

Severity

Syntax

STX_VH_346

Alias for subprogram must include a signature.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find signature when an alias for a subprogram was declared. When we declare an alias for a subprogram, the signature identifies which overloaded version of the subprogram name is aliased. The signature lists the types of each of the subprogram's parameters, in the same order that they appear in the subprogram's declaration.

For example, if a package arithmetic_ops declares two procedures as follows:

```
procedure increment (bv:inout bit_vector;by :in integer:=1);
procedure increment (int:inout integer;by:in integer:=1);
```

we can declare aliases for the procedure as follows:

```
alias bv_increment is
work.arithmetic_ops.increment[bit_vector,integer];
alias int_increment is
work.arithmetic_ops.increment[integer,integer];
```

If the subprogram is a function, the signature also includes the type of the return value, after the keyword `return`

Message Details

`Alias for subprogram must include a signature`

Severity

Syntax

STX_VH_347

Alias for enumeration element must include a signature.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find signature when an alias for enumeration literals is declared. If we wish to alias an individual literal of an enumeration type, we must deal with the possibility that the literal may belong to several different enumeration types. We can use a signature to distinguish one particular meaning by noting that an enumeration literal is equivalent to a function with no parameters that return a value of the enumeration type.

For example, when we write the enumeration literal '1',we can think of this as a call to a function with no parameters, returning a value of type bit. we can write an alias for this literal as follows:

```
alias high is std.standard.'1'[return bit];
```

Note that a selected name is required for a character literal, since a character literal by itself is not a syntactically valid name.

Message Details

`Alias for enumeration element must include a signature`

Severity

Syntax

STX_VH_348

Parameter type is not allowed in signature for enumeration element.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of parameter type used in signature for enumeration element. A signature is said to match the parameter and result type profile of a given enumeration literal if the signature matches the parameter and result type profile of the subprogram equivalent to the enumeration literal. For example, the following code will produce this error:

```
entity ent is
   port(in1:integer);
end;

architecture arch of ent is
   attribute att : bit;
   begin
     process(in1)
     type enum is (a,b,c);
     type enum1 is (a,b,c);
      attribute att of a [integer] : literal is '1';
      --Error here: Parameter type integer used in
      --signature for enumeration literal.
     begin
     end process;
   end;
```

Message Details

```
Parameter type not allowed in signature for enumeration element
```

Severity

Syntax

STX_VH_349

Object alias expression must denote a static name.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an object alias expression that did not denote a static name. An object alias is an alias whose alias designator denotes an object (that is, a constant, signal, variable, or file). The language requires that this object should denote a static name.

For example, the following code will produce this error:

```
entity ent is
 end ent;

 architecture arch of ent is
   type arr is array(0 to 1) of integer;
   type acc is access arr;

   begin
      process
      variable var : acc;
       alias al : integer is var(0); -- prefix var is
       -- an access type and is therefore not static
      begin
         wait;
      end process;
  end;
```

Message Details

```
Object alias expression must denote a static name
```

Severity

Syntax

STX_VH_350

Objects of multi-dimensional array type cannot be aliased.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the type specified for an ObjectAlias is multi-dimensional array type. There is no such restriction on NonobjectAlias because no type information needs to be specified for NonobjectAlias.

Message Details

```
Cannot alias objects of multi-dimensional array type
```

Severity

Syntax

STX_VH_351

Actual associated with a formal port of mode BUFFER must also be of BUFFER mode

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the VHDL language not being conformed:

For a formal port of mode BUFFER, the associated actual may only be a port of mode BUFFER. Also, the types of the formal and the actual being associated must be the same.

Message Details

```
Mismatch of direction between <PORT | PARAM> <name1>
(<actual-mode>) (expected mode BUFFER) and corresponding <PORT
| PARAM> <name2> (<port/param-type>) of <COMPONENT | ENTITY |
BLOCK | PROCEDURE | FUNCTION> <name2> at line <line-num> of
file <file-name>
```

Severity

Syntax

STX_VH_352

Actual associated with a formal port of mode INOUT must also be of INOUT mode

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the VHDL language not being conformed:

For a formal port of mode INOUT, the associated actual may only be a port of mode INOUT. Also, the types of the formal and the actual being associated must be the same.

Message Details

```
Mismatch of direction between <PORT | PARAM> <name1>
(<actual-mode>) (expected mode INOUT) and corresponding <PORT |
PARAM> <name2> (<port/param-type>) of <COMPONENT | ENTITY |
BLOCK | PROCEDURE | FUNCTION> <name2> at line <line-num> of
file <file-name>
```

Severity

Syntax

STX_VH_353

Actual associated with a formal port of mode OUT must be of either OUT or INOUT mode

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the VHDL language not being conformed:

For a formal port of mode OUT, the associated actual may only be a port of mode OUT or INOUT. Also, the types of the formal and the actual being associated must be the same.

Message Details

```
Mismatch of direction between <PORT | PARAM> <name1>
(<actual-mode>) (expected mode OUT or INOUT) and corresponding
<PORT | PARAM> <name2> (<port/param-type>) of <COMPONENT |
ENTITY | BLOCK | PROCEDURE | FUNCTION> <name2> at line <line-
num> of file <file-name>
```

Severity

Syntax

STX_VH_354

Actual associated with a formal port of mode IN cannot be of mode OUT or LINKAGE

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the VHDL language not being conformed:

For a formal port of mode IN, the associated actual may only be a port of mode IN, INOUT, or BUFFER. Also, the types of the formal and the actual being associated must be the same.

Message Details

```
Mismatch of direction between '<PORT | PARAM>' <name1> ('<IN |
OUT | INOUT>') (expected mode IN or INOUT or BUFFER) and
corresponding '<PORT | PARAM>' <name2> ('<IN | OUT | INOUT>')
of '<COMPONENT | ENTITY | BLOCK | PROCEDURE | FUNCTION>'
<name3> at line <line-num> of file <file-name>
```

Severity

Syntax

STX_VH_355

A port whose mode is IN or LINKAGE cannot be the target of assignment statement

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed.

A port whose mode is LINKAGE cannot be on the left-hand side of a signal assignment statement.

The value of a LINKAGE port can be read and updated, but only by another port of mode LINKAGE.

Message Details

```
Invalid assignment target - <rule> is of LINKAGE mode
```

Severity

Syntax

STX_VH_356

Parameter must be specified for attribute (by context)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find any specified parameter or value for the attribute.

Message Details

```
Expecting parameter for attribute <attribute>
```

Severity

Syntax

STX_VH_357

Type of the attribute parameter must match its usage

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a type mismatch in attribute parameter. For example, SpyGlass flags an error in the following case:

```
package pack is
  constant const : integer := integer'val('a');
end;
```

In the above example, `a` is a character type, whereas it should be an integer.

Message Details

```
Type error in attribute parameter '<parameter>' - expecting
type <type>
```

Severity

Syntax

STX_VH_358

Syntax error detected

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it encountered a symbol or sequence of symbols in the VHDL source file that is illegal for the context in which it appears. The syntax of VHDL is a set of rules defining the expected order of characters, words, and constructs in a source file. If VHDL source file does not conform to these rules, SpyGlass will generate this message along with an indication of wherein the source file the error was first detected.

Message Details

`Syntax error <error>`

Severity

Syntax

STX_VH_359

Use only the allowed (by context) overloaded functions for a return type

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a mismatch in the return type of Overloaded Function.

Message Details

```
Only following Overloaded Functions are visible for return type
<type>
```

Severity

Syntax

STX_VH_360

Use only the allowed (by context) functions for a return type

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a mismatch in the return type of Function

Message Details

```
Only following Function is visible for return type <type>
```

Severity

Syntax

STX_VH_361

Signature can be applied only on a subprogram or an enumeration literal

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an invalid usage of Signature. The Signature can be applied only on an overloaded subprogram or an enumeration literal

Message Details

```
Signature can be applied only on a subprogram or an enumeration literal
```

Severity

Syntax

STX_VH_362

Declaration corresponding to a signature must exist / be visible

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find any declaration visible corresponding to Signature.

Message Details

```
No declaration <declaration> visible corresponding to signature
```

Severity

Syntax

STX_VH_363

NULL literal may be used to denote a value only for access types

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of NULL literal with types other than access types.

For access type only, there is a literal NULL that has a NULL access value designating no object at all. The NULL value of an access type is the default initial value of the type.

Message Details

```
NULL literal may be used to denote a value only for access
types
```

Severity

Syntax

STX_VH_364

Attribute association with identifier must be valid by context

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an invalid association of attribute.

An attribute is a value, function, type, range, signal, or constant that may be associated with one or more named entities in a description.

A user-defined attribute of a port, signal, variable, or constant of some composite type is an attribute of the entire port, signal, variable, or constant, not of its elements.

Message Details

```
Attribute <attribute> cannot be associated with <entity>
```

Severity

Syntax

STX_VH_365

Use of UNAFFECTED keyword is not allowed in the waveform of a sequential signal assignment

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed.

It is an error if the reserved word UNAFFECTED appears as a waveform in a (sequential) signal assignment statement. The reserved word UNAFFECTED may only appear as a waveform in concurrent signal assignment statements. The use of UNAFFECTED keyword in a concurrent signal assignment causes no change to the driver for the target signal.

Message Details

```
Use of keyword UNAFFECTED is not allowed in the waveform of a
sequential signal assignment
```

Severity

Syntax

STX_VH_366

Reject time value must not be greater than the value associated with the first waveform

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the reject time value greater than the value associated with the first waveform.

Message Details

```
Reject time value must not be greater than the value associated
with the first waveform
```

Severity

Syntax

STX_VH_367

Component declaration must be valid in the present context

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an invalid component declaration statement.

Message Details

```
'<declaration>' does not denote a component declaration
```

Severity

Syntax

STX_VH_368

NULL waveform is not permitted in a concurrent signal assignment

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found NULL waveform in a concurrent signal assignment. The execution of the NULL statement has no effect other than to pass on to the next statement.

It is an error if a NULL waveform element appears in a waveform of a concurrent signal assignment statement.

Message Details

```
NULL waveform is not permitted in a concurrent signal
assignment
```

Severity

Syntax

STX_VH_369

Resolution function parameter type must denote an unconstrained array

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed.

A resolution function must be a pure function. It must have a single input parameter of class constant that is a one-dimensional, unconstrained array whose element type is that of the resolved signal. The type of the return value of the function must also be that of the signal.

Message Details

```
Resolution function parameter type must denote an unconstrained array
```

Severity

Syntax

STX_VH_370

Resolution function must not be impure

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed.

A resolution function must be a pure function. A pure function is one that returns the same value each time the function is called from same set of actuals. An impure function is one that potentially returns different values each time it is called with same set of actuals. If neither PURE nor IMPURE is present in the function specification, the function is, by default, a PURE function.

Message Details

```
Resolution function must not be impure
```

Severity

Syntax

STX_VH_371

No operator symbol visible corresponding to signature.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find specified operator visible corresponding to Signature.

Message Details

```
No operator symbol '<symbol>' visible corresponding to
signature.
```

Severity

Syntax

STX_VH_372

Alias for string must include a signature

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the following rule imposed by the language is being conformed.

A signature may not appear in a declaration of an object alias.

A signature is required for a nonobject alias if the name denotes a subprogram (including an operator) or enumeration literal.

Message Details

```
Alias for string must include a signature
```

Severity

Syntax

STX_VH_373

Use of variable parameters in concurrent procedure call is not allowed

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found usage of variable parameters in concurrent procedure call, which is not permissible.

Message Details

```
Cannot use variable parameters in concurrent procedure call
```

Severity

Syntax

STX_VH_374

String used has ambiguous type - string versus bit_vector

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found ambiguity in string usage

Message Details

`String <string> has ambiguous type - string versus bit_vector`

Severity

Syntax

STX_VH_375

Instance must not be configured more than once.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found re-configuration of component using a `generate` statement. While using `generate` statement for component binding, no generate index should be used again.

Message Details

```
Instance <instance> has already been configured for the
generate index <index>
```

Severity

Syntax

STX_VH_376

Instance must not be configured more than once

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found re-configuration of component using a previous full configuration specification.

Message Details

```
Instance <instance> has already been configured using a
previous full configuration
```

Severity

Syntax

STX_VH_377

Index specification must match with corresponding for generate range

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found mismatch in the index specification and the corresponding `for generate` range. The index specified must lie within the range of corresponding `for generate` range.

Message Details

```
Index specification does not match with corresponding for
generate range
```

Severity

Syntax

STX_VH_378

Invalid index specification in for generate block configuration

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found invalid index specification in the `for generate` block configuration.

Message Details

```
Invalid index specification in for generate block configuration
```

Severity

Syntax

STX_VH_379

Index specification must match with corresponding for generate range.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found mismatch in the index specification and the corresponding `for generate` range. The index specified must lie within the range of corresponding `for generate` range.

Message Details

```
Index <index> is not covered by the for generate discrete range
```

Severity

Syntax

STX_VH_380

Instance must have a valid configuration specification before usage (by context)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate presence of an instance with no previous valid configuration specification. Instance must have a valid configuration specification before usage (by context).

Message Details

```
Instance <instance> must first have a valid config
specification
```

Severity

Syntax

STX_VH_381

Component instances are not bound to the same design unit

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate presence of component instances not bound to the same design unit

Message Details

```
Component instances are not bound to the same design unit
```

Severity

Syntax

STX_VH_382

OPEN may not be used as actual if there is a conversion function on the formal

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found usage of OPEN construct as actual for the corresponding conversion function based formal. It is an error if an actual of open is associated with a formal that is associated individually.

Message Details

```
OPEN may not be used as actual if there is a conversion
function on the formal
```

Severity

Syntax

STX_VH_383

Appropriate architecture should be configured in present context

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found configuration of an inappropriate/invalid architecture as per present context.

Message Details

`Only architecture <architecture> can be configured here`

Severity

Syntax

STX_VH_385

Invalid expanded name in use clause

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an invalid expanded name in use clause.

Message Details

```
Invalid expanded name in use clause
```

Severity

Syntax

STX_VH_386

No more declarations allowed in the scope after OTHERS/ALL attribute specification

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed.

An attribute specification with the entity name list OTHERS or ALL for a given entity class that appears in a declarative part must be the last such specification for the given attribute for the given entity class in that declarative part. No named entity in the specified entity class may be declared in a given declarative part following such an attribute specification.

Message Details

```
No more <declaration> declarations allowed in this scope after
OTHERS/ALL attribute specification
```

Severity

Syntax

STX_VH_388

Signature needed to resolve attribute name uniquely

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not resolve attribute name uniquely without signature.

Message Details

```
Cannot resolve attribute name uniquely without signature
```

Severity

Syntax

STX_VH_389

Type conversion is not allowed for a formal of a FILE type

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the FILE type object as a parameter to conversion function. Type conversion is not allowed for a formal of FILE type.

Message Details

```
Type conversion is not allowed for a formal of a FILE type
```

Severity

Syntax

STX_VH_390

Range constraint cannot be specified for an access type

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed.

Range constraint cannot be specified for an access type.

The only form of constraint that is allowed after the name of an access type in a subtype indication is an index constraint.

Message Details

```
Range constraint cannot be specified for an access type
```

Severity

Syntax

STX_VH_391

A constant, IN port, IN generic, or expression cannot be associated with an INOUT/BUFFER formal

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed:

A constant, IN port, IN generic, or expression cannot be associated with an INOUT/BUFFER formal.

Also, the types of the formal and the actual being associated must be the same.

Message Details

```
A constant, IN port, IN generic, or expression, cannot be
associated with the INOUT/BUFFER formal <formal>
```

Severity

Syntax

STX_VH_392

Overloaded operator function and the implicit operator are both valid in this context

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that overloaded operator function and the implicit operator are both valid in this context.

Message Details

```
Overloaded operator function "<function>" and the implicit
operator <operator> are both valid in this context
```

Severity

Syntax

STX_VH_393

Attribute parameter must be static (by context)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it expected the attribute parameter to be static, but did not find so.

For example-Argument to signal-valued function attribute `DELAYED` must be static.

Message Details

```
Attribute parameter must be static in this case
```

Severity

Syntax

STX_VH_394

Explicit port map needed in a block

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find explicit port map for the block in context.

Message Details

```
This block requires an explicit port map
```

Severity

Syntax

STX_VH_395

Actual associated with generic formal must denote a constant expression

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find the actual as a constant expression. For valid match between actual and formal, the actual must be a constant expression, since the corresponding generic formal is always of constant type.

Message Details

```
Actual associated with generic formal <formal> must denote a
constant expression
```

Severity

Syntax

STX_VH_396

Interface declaration list must be complete before using its elements further in the list.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the interface declaration list must be complete before using its elements further in the list.

Message Details

```
Cannot use <declaration> till this interface declaration list is complete
```

Severity

Syntax

STX_VH_397

Internal Error

Language

VHDL

Rule Description

Internal Error

Message Details

```
Internal error: routine <routine>. Please contact Atrenta
Support.
```

Severity

INTERNAL_FATAL

STX_VH_398

Illegal signal specified in subprogram

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of an invalid signal drive statement in subprogram. A subprogram can drive only its own parameters.

Message Details

```
Cannot drive signal <signal> in this subprogram
```

Severity

Syntax

STX_VH_399

OTHERS must be the only choice in a list of choices

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that OTHERS must be the only choice in a list of choices. The use of OTHERS means it refers to all previously unassigned values. Therefore, when used it must be the last association.

For example, the following statement is invalid:

```
when 1|others => output <= 10;
```

Message Details

```
OTHERS must be the only choice in a list of choices
```

Severity

Syntax

STX_VH_400

Case expression of array type must have a static subtype

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the case expression of array type must have a static subtype.

Message Details

```
Case expression of array type must have a static subtype
```

Severity

Syntax

STX_VH_401

IN/LINKAGE port cannot be associated with formal port of mode OUT

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the VHDL language not being conformed:

For a formal port of mode OUT, the associated actual may only be a port of mode OUT or INOUT. Also, the types of the formal and the actual being associated must be the same.

Message Details

```
Mismatch of direction between <PORT | PARAM> <name1>
(<actual-mode>) (expected mode OUT or INOUT) and corresponding
<PORT | PARAM> <name2> (<port/param-type>) of <COMPONENT |
ENTITY | BLOCK | PROCEDURE | FUNCTION> <name2> at line <line-
num> of file <file-name>
```

Severity

Syntax

STX_VH_402

Cannot write to read-only file

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not write to read-only file.

Message Details

```
Cannot write to read-only file <file>
```

Severity

Syntax

STX_VH_403

Cannot read file in write mode

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not read file opened in write mode.

Message Details

```
Cannot read file <file> opened in write mode
```

Severity

Syntax

STX_VH_404

Variable declaration not permitted in this declarative region

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found variable declaration in a region not permitted for variable declaration.

Message Details

```
Variable declaration not permitted in this declarative region
```

Severity

Syntax

STX_VH_407

Full configuration specification for instance after previous partial configuration specification is not permissible

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate an invalid full configuration specification for instance after previous partial configuration specification is not permissible.

Message Details

```
Invalid full configuration specified for instance <instance>
after previous partial configuration on it
```

Severity

Syntax

STX_VH_408

Type conversion function output must be a constrained type for a formal of unconstrained type

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found type conversion function output of unconstrained type for a formal of unconstrained type. The conversion function output must be of constrained type, because for a subprogram parameter of unconstrained type, the constraint is obtained from the actual parameter passed in during the subprogram call.

Message Details

```
Since formal is of unconstrained type, type conversion function output must be a constrained type
```

Severity

Syntax

STX_VH_409

Configuration specified must be visible in scope of context

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find the configuration specified in the scope of usage. The configuration must be visible in the scope of usage

Message Details

```
Unknown configuration <configuration>, use expanded name
```

Severity

Syntax

STX_VH_410

Entity specified must be visible in scope of context

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find the entity specified in the scope of usage. The entity must be visible in the scope of usage

Message Details

`Unknown entity <entity>, use expanded name`

Severity

Syntax

STX_VH_411

An aggregate with multiple associations cannot have a non-static choice

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of non-static choice(s) in the aggregate specification. All the choices of the aggregate must be static in nature. A named association of an array aggregate is locally static, or likewise a choice that is a null range, only if the aggregate includes a single element association and this element association has a single choice. A OTHERs choice is locally static if the applicable index constraint is locally static.

Message Details

```
An aggregate with multiple associations cannot have a non-
static choice
```

Severity

Syntax

STX_VH_412

Floating point arithmetic overflow - value beyond Infinity

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of floating point arithmetic overflow - value beyond Infinity.

Message Details

```
Floating point arithmetic overflow - value beyond Infinity
```

Severity

Syntax

STX_VH_413

Named association should be valid

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate to indicate presence of an invalid named association in the given context

Message Details

```
Named association is not valid in this context
```

Severity

Syntax

STX_VH_414

Signal assignment not permitted in passive process

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found signal assignment in passive process. A process statement is said to be a passive process if neither the process itself, nor any procedure of which the process is a parent, contains a signal assignment statement. Such a process, or any concurrent statement equivalent to such a process, may appear in the entity statement part of an entity declaration.

Message Details

```
Signal assignment not permitted in passive process
```

Severity

Syntax

STX_VH_415

Element associations of an array aggregate must be all either positional or named

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule, specified by the language, not being met.

Apart from a final element association with the single choice others, the rest (if any) of the element associations of an array aggregate must be either all positional or all named.

Message Details

```
Element associations of an array aggregate must be all either
positional or named.
```

Severity

Syntax

STX_VH_416

Parameter type of identifier must match with respective declaration

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found mismatch of parameter type of identifier with respective declaration. The parameter type of identifier must be consistent with its previous declaration.

Message Details

```
Parameter type of <identifier> does not match with declaration
```

Severity

Syntax

STX_VH_417

OTHERS may not be used after a previous full configuration

Language

VHDL

Rule Description

SpyGlass generates this syntax error because OTHERS may not be used after a previous full configuration

Message Details

```
OTHERS may not be used after a previous full configuration
```

Severity

Syntax

STX_VH_418

Attributes specified for entities must be locally static

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule, specified by the language, not being met.

If the entity name list denotes an entity interface, architecture body, or configuration declaration, then the expression in the attribute specification is required to be locally static.

Message Details

```
Attributes specified for entities must be locally static
```

Severity

Syntax

STX_VH_419

Illegal use of keyword END FUNCTION to end procedure body: end procedure expected

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate illegal use of keyword END FUNCTION to end procedure body was found. END PROCEDURE expected.

Message Details

```
Invalid use of keyword END FUNCTION to end procedure body: end procedure expected
```

Severity

Syntax

STX_VH_420

Illegal use of keyword END PROCEDURE to end function body: end function expected

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate illegal use of keyword END PROCEDURE to end function body was found. END FUNCTION expected.

Message Details

```
Invalid use of keyword END PROCEDURE to end function body: end function expected
```

Severity

Syntax

STX_VH_421

Illegal use of 'RANGE/'REVERSE_RANGE attribute

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate illegal use of 'RANGE/ 'REVERSE_RANGE attribute.

For Example:

```
package pack is
  constant const1: bit_vector(0 to 1) := "11";
  constant const2 : integer := const1'range;
  -- ERROR,   'range is not allowed in this context
end;
Message Details
Invalid use of 'range/'reverse_range attribute
```

Severity

Syntax

STX_VH_423

Each component local must be associated in configuration specification

Language

VHDL

Rule Description

This error occurs when SpyGlass finds component local unassociated in configuration specification. Each component local must be associated in configuration specification.

Message Details

```
Component local <component-local> is not associated in
configuration specification
```

Severity

Syntax

STX_VH_424

Type conversion operand type bounds not consistent with target type bounds

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found type conversion operand type bounds not consistent with target type bounds.

Message Details

```
Type conversion operand type bounds not consistent with target type bounds
```

Severity

Syntax

STX_VH_425

Use operators unambiguously; multiple operators should not be visible

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate ambiguous use of operator was found by SpyGlass. Multiple operators should not be visible.

For Example:

```
process(sig)
 begin
  sig <=std_logic_vector'(x"11") >=
std_logic_vector'(('1','1','1','1'));
  sig <=bit_vector'(x"11") <=
bit_vector'(('1','1','1','1'));
  sig <=(x"11") > ('1','1','1','1');       -- error
end process;
```

Message Details

```
Operator "<operator>" is ambiguous - more than one visible
```

Severity

Syntax

STX_VH_426

Primary binding specification is must for a component before its usage in configuration declaration

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find primary binding specification for component specified.

When a binding indication is used in a configuration specification, it is an error if the entity aspect is absent. A binding indication appearing in a component configuration need not have an entity aspect under the following condition: The block corresponding to the block configuration in which the given component configuration appears is required to have one or more configuration specifications that together configure all component instances denoted in the given component configuration. Under this circumstance, these binding indications are the primary binding indications. It is an error if a binding indication appearing in a component configuration does not have an entity aspect and there are no primary binding indications.

Message Details

```
Component instance <instance> must first have a primary binding
```

Severity

Syntax

STX_VH_427

Port already associated in the primary binding must not be used

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that a port already associated in the primary binding must not be used, as per language specification.

Message Details

```
Port <port> already associated in the primary binding
```

Severity

Syntax

STX_VH_428

Static expression overflow: value is out of range for the type specified

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found static expression overflow.

For Example:

```
CONSTANT const : TIME := time'high + 1 fs;
-- Error : overflow of time constant value

CONSTANT const1 : TIME := time'low - 1 fs;
```

Message Details

```
Static expression overflow : value is out of range for the type
<type>
```

Severity

Syntax

STX_VH_429

Size of actual must match with size of formal

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found mismatch between the size of actual and the formal parameter.

For example:

```
entity ent is end;
architecture arch of ent is
  procedure proc(in1:bit_vector(0 to 3)) is
  begin end;
 subtype sub is bit_vector(0 to 2);
 constant const:sub:="111";
begin
  proc(const); --error
end;
```

Message Details

```
Actual size <size> mismatch with formal <formal> of size
<formal-size>
```

Severity

Syntax

STX_VH_430

Library name must be specified for every design unit used

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find any library name for the specified design unit.

Message Details

```
No Library name is specified for the design unit <design-unit>
```

Severity

Syntax

STX_VH_431

Cannot initialize library manager. Missing/Invalid .jaguarc

Language

VHDL

Rule Description

Message Details

```
Cannot initialize library manager. Missing/Invalid .jaguarc
```

Severity

Syntax

STX_VH_432

Static expression overflow: value out of range for the type specified

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate presence of static expression overflow.

For example:

```
entity ent is end;
architecture arch of ent is
    CONSTANT const : TIME := time'high + 1 fs;
    -- Error : overflow of time constant value
    CONSTANT const1 : TIME := time'low - 1 fs;
begin
 assert (now = (time'high - 1 fs))
 --SpyGlass gives error
    report "no error ";
end;
```

Message Details

```
Static expression overflow : value of <expression> is out of
range for the type
```

Severity

Syntax

STX_VH_436

Maximum instance recursion limit (implementation-dependent) reached while elaborating architecture

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that maximum instance recursion limit (implementation-dependent) was reached while elaborating architecture.

For Example:

```
 ENTITY EEEE IS
    port (clk : IN BIT);
 END ENTITY EEEE;
   ARCHITECTURE AAAA OF EEEE IS
     COMPONENT EEEE IS
      port (clk : IN BIT);
     END COMPONENT EEEE;
     BEGIN
      inst2:  EEEE
       PORT MAP (clk => clk);
END ARCHITECTURE AAAA;
```

Message Details

```
Maximum Instance recursion limit (Implementation Dependent)
reached while trying to elaborate architecture <architecture>
```

Severity

Syntax

STX_VH_437

Maximum feature recursion limit (implemenetation dependent) reached. Use set_option allow_recursion_limit X to increase/ decrease this limit.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the maximum recursion limit (implementation dependent) was reached while elaboration.

You can remove the violation of this rule by increasing the recursion limit to a value ranging from 10241 to 2147483647. By default, the maximum recursion limit is 10240.

To change the recursion limit, use the following command:

```
set_option allow_recursion_limit <limit>
```

The following example shows the usage of this command:

```
set_option allow_recursion_limit 2147483647
```

Message Details

```
Maximum Feature Recursion Limit (Implementation Dependent)
reached. Use set_option allow_recursion_limit <value> to
increase the limit and suppress this error.
```

Severity

Syntax

STX_VH_438

Internal - Stack : out of memory

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate

Message Details

```
Internal - Stack : out of memory. Please contact Atrenta
Support. Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VH_439

Internal - Stack: nothing to Pop

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the internal stack has nothing to pop.

Message Details

```
Internal - Stack: nothing to Pop. Please contact Atrenta
Support. Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VH_446

Index value used must not be out of elaborated range specified

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the index value not lying in the elaborated range.

For Example:

```
type ADDRESS_WORD is array(3 to 7) of BIT;
signal ADDRESS_BUS : ADDRESS_WORD;
ADDRESS_BUS(4 to 6) <= "010"; --error
```

Message Details

```
Index value <value> is out of elaborated range <r1> to <r2>
```

Severity

Syntax

STX_VH_447

Elaborated values must constitute a valid slice range

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the elaborated values do not constitute a valid slice range.

Message Details

```
Elaborated values <value1> to <value2> do not constitute a
valid slice range
```

Severity

Syntax

STX_VH_448

Return expression sizes do not match

Language

VHDL

Rule Description

Message Details

```
Return expression sizes do not match
```

Severity

Syntax

STX_VH_449

Attribute 'BASE' is allowed only as prefix to different attribute

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate an illegal usage of pre-defined attribute 'BASE. It must conform to the following rule set by the language. This attribute cannot be used in expressions as such since it returns a base type, but it can be used in conjunction with other attributes.

T'BASE

```
Kind:            Type.
Prefix:          Any type or subtype T.
Result:          The base type of T.
Restrictions:    This attribute is allowed only as the
prefix of the name of another attribute.
```

For example:

```
T'BASE'LEFT.
```

Message Details

```
Attribute 'BASE is allowed only as prefix to different
attribute
```

Severity

Syntax

STX_VH_455

Others must be only choice in aggregate of non-locally static size

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find OTHERS as the only choice in aggregate of non-locally static size.

For Example:

```
entity E is
end;
 architecture A of E is
  function ff(a: bit_vector) return integer is
       type mytype is array(a'range) of bit;
        variable j: mytype;
    begin
     j := (4=>'1', 5=>'0', others=>'1'); -- error
     return a'length;
    end;
  begin
 end A;
```

A named association of an array aggregate is allowed to have a choice that is not locally static, or likewise a choice that is a null range, only if the aggregate includes a single element association and this element association has a single choice. A OTHERs choice is locally static if the applicable index constraint is locally static.

Message Details

```
OTHERS must be only choice in aggregate of non-locally static
size
```

Severity

Syntax

STX_VH_458

Actual for formal should either be a static signal name or static expression (VHDL-93)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it expected that actual for formal should be a static signal name or static expression (VHDL-93).

For Example:

```
entity e is
end;
architecture a of e is
  signal next_count : bit_vector(3 downto 0);
  component mycomp
    port (portIn : in bit_vector(3 downto 0);
          portOut : out bit_vector(3 downto 0));
  end component;
  begin
   myinst : mycomp port map (portIn => next_count,
                   portOut => next_count); -- legal
   myinst1 : mycomp port map (portIn => (next_count
     & next_count), portOut => next_count); -- error
end;
```

Message Details

```
Actual for formal <formal> should either be a static signal
name or static expression (VHDL-93)
```

Severity

Syntax

STX_VH_459

Illegal syntax: concurrent procedure call statement expected here

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate presence of an illegal syntax. Only concurrent assertion statements, concurrent procedure call statements, or process statements may appear in the entity statement part. All such statements must be passive.

Message Details

```
Illegal syntax: concurrent procedure call statement expected
here.
```

Severity

Syntax

STX_VH_460

Expression used must be of pre-defined type BOOLEAN (by context)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the expression used must be of pre-defined type BOOLEAN (by context).

Message Details

```
<expression> expression must be of pre-defined type BOOLEAN
```

Severity

Syntax

STX_VH_464

Either component or procedure name expected

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that an undeclared, undefined component or procedure was detected by SpyGlass.

For Example:

```
entity top is
end top;
 architecture rtl of top is
   begin
    inst : test ;
 end architecture rtl;
```

Message Details

<name> denotes neither a component nor a procedure

Severity

Syntax

STX_VH_465

Bounds of range must be of Integer or enumerated type (by context)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that bounds of range must be of Integer or enumerated type (by context).

Message Details

`Bounds of <range> range must be of Integer or enumerated type`

Severity

Syntax

STX_VH_466

Value specified for attribute should be of valid type (by context)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the value specified for attribute was found to be of unexpected type.

For Example:

```
 variable a1_delay : INTEGER := 20;
 variable b1_delay : TIME := 20;
 out1 <= in1'DELAYED(a1_delay); -- error, value
-- specified for attribute should be of type TIME

out2 <= in2'DELAYED(b1_delay); -- no error
```

Message Details

```
Value specified for attribute <attribute> should be of type
<type>
```

Severity

Syntax

STX_VH_467

Waveform element type cannot be uniquely determined based on the aggregate on LHS

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that waveform element type could not be uniquely determined based on the aggregate on LHS. The type of an aggregate must be determinable solely from the context in which the aggregate appears, excluding the aggregate itself but using the fact that the type of the aggregate must be a composite type. The type of an aggregate in turn determines the required type for each of its elements.

Message Details

```
The type of the waveform element cannot be uniquely determined
based on the aggregate on the lhs
```

Severity

Syntax

STX_VH_468

Expression type must be uniquely determinable in the present context

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that expression type could not be uniquely determined. The type of an expression depends only upon the types of its operands and on the operators applied; for an overloaded operand or operator, the determination of the operand type, or the identification of the overloaded operator, depends on the context.

Message Details

```
The type of expression <expression> cannot be uniquely
determined
```

Severity

Syntax

STX_VH_469

Use of ALL not allowed in selected name not denoting an access type

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed

Use of ALL keyword not allowed in selected name not denoting an access type.

Message Details

```
Use of ALL not allowed in selected name not denoting an access type
```

Severity

Syntax

STX_VH_470

Unknown port must not be used in association list

Language

VHDL

Rule Description

This error occurred because SpyGlass found port of an unknown type being used in association list.

Message Details

`Unknown port <port>`

Severity

Syntax

STX_VH_472

Illegal access of signal/variable from within pure function is not allowed

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate illegal access of signal/variable from within pure function. A pure function returns the same value each time it is called using the same values as actual parameters.

Message Details

```
Invalid access of <signal | variable> <name> from within pure function <function>
```

Severity

Syntax

STX_VH_473

Illegal call of impure subprogram from within pure function is not allowed

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed.

A pure function must not be the parent of an impure function. Thus it is illegal to call an impure function from within a pure function. A function that returns the same value each time it is called with the same values as actual parameters is a pure function. An impure function may return a different value each time it is called, even when different calls have the same actual parameter values.

Message Details

```
Invalid call of impure subprogram <sub-program> from within
pure function <function>
```

Severity

Syntax

STX_VH_474

Could not resolve signal/variable/constant without context, multiple definitions visible

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that multiple definitions were visible for identifier/construct, so it could not resolve without context.

For Example:

```
type days is (mon, tue, wed, thu, fri, sat, sun);
type weekdays is (mon, tue, wed, thu, fri);
type startdays is array (mon to wed) of integer;
--Error
```

Message Details

```
Could not resolve <definition> without context because multiple
definition visible
```

Severity

Syntax

STX_VH_481

Report expression should be of pre-defined type STRING

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found that report expression is not of type STRING. As per the language specification, the report statement expression must be of the predefined type STRING.

For Example:

```
function test_func return Integer is
variable aa:String(1 to 38) := "In Function test_func(testing
default)";
variable bb : bit_vector(0 to 3);
begin
    report aa;        -- Valid
    report "Function test_func testing level-WARNING"
severity NOTE;
    report bb;        -- Error
    return 3;
end;
```

Message Details

```
Report expression should be of pre-defined type STRING
```

Severity

Syntax

STX_VH_482

Severity expression should be of the pre-defined enumeration type SEVERITY_LEVEL

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found severity expression is not of pre-defined enumeration type SEVERITY_LEVEL. If the severity clause is present, it must specify an expression of predefined type SEVERITY_LEVEL. The severity clause specifies a severity level associated with the report. In the absence of a severity clause for a given report, the default value of the severity level is NOTE.

Message Details

```
Severity expression should be of the pre-defined enumeration
type SEVERITY_LEVEL
```

Severity

Syntax

STX_VH_486

Unrecognized type used in qualified expression

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not recognize the type used in qualified expression. The type specified must be legal

Message Details

```
Unrecognized type used in qualified expression
```

Severity

Syntax

STX_VH_487

Procedure must be visible corresponding to the specified arguments and their types

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not detect any procedure corresponding to the specified arguments and their types

For Example:

```
entity ent is
end;
architecture arch of ent is
   unction func(signal in1: integer) return integer is
        begin
          return 10;
      end;
      procedure proc (signal in1: integer) is
        begin
      end;
 signal sig : integer;
 begin
   proc(in1=>func(sig)); -- Error, type conversion on
                         -- actual for signal formal
end;
```

Message Details

```
No procedure <procedure> is visible corresponding to the
specified arguments and their types
```

Severity

Syntax

STX_VH_488

Named entity referenced by an alias name should be defined in the current declarative region

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not detect named entity referenced by an alias name in the current declarative region.

Message Details

```
Named entity <entity> referenced by the alias name <name>
should be defined in the current declarative region
```

Severity

Syntax

STX_VH_490

Library Dump of Design Unit must be compatible with the current version.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the library dump is compiled using an earlier version of SpyGlass and is not compatible with the current version, because of changes in the SpyGlass internal data model. Delete the whole library and recompile it with the current version of SpyGlass. (These libraries are specified by the `set_option lib <logical-lib-name> <physical-path>` command in the project file)

SpyGlass generates this syntax error in the following case:

File1: pack.vhd

```
package pack is
 constant c : integer := 5;
end pack;

package body pack is
end pack;
```

File2: test.vhd

```
library abc;
 use abc.pack.all;

 entity e is
 generic (g : integer := c);
 end e;
```

```
architecture a of e is
begin
end a;
```

Compilation step1:

a) Use any older version of SpyGlass, and specify the following commands in a project file:

```
set_option work abc
set_option lib abc ./abc
read_file -type vhdl pack.vhd
```

b) Use latest version of SpyGlass, and specify the following commands in a project file:

```
set_option lib abc ./abc
read_file -type vhdl test.vhd
```

After performing the above steps the STX_VH_490 rule displays the following message:

```
Library Dump of the package 'PACK' inside the library 'abc' is not compatible with the current version of SpyGlass.
```

Message Details

```
Library Dump of the <design-unit-type> '<design-unit-name>' inside the library '<library-name>' is not compatible with the current version of SpyGlass
```

Severity

Syntax

STX_VH_492

Aliasing of ACCESS type objects is not allowed

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found aliasing of ACCESS type objects, which is not allowed as per language specification.

Message Details

```
Cannot alias object of ACCESS type
```

Severity

Syntax

STX_VH_494

Non-printable characters are not allowed in extended identifiers

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that non-printable characters were detected in extended identifiers by SpyGlass, which is not allowed as per language specification.

Message Details

```
Non-printable character <character> not allowed in extended
identifiers
```

Severity

Syntax

STX_VH_495

Signal references must elaborate statically

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it detected illegal reference of signal during static elaboration.

For Example:

```
entity ent is
  port ( a : NATURAL range 0 to 255) ;
end ent ;
architecture arch of ent is
  type array_type is array(NATURAL range <>) of NATURAL ;
  CONSTANT value : NATURAL := 10 ;
  signal sig : array_type ( a downto 0 ) ; -- Error
   begin
end ;
```

Message Details

`Invalid reference of signal - cannot elaborate statically`

Severity

Syntax

STX_VH_496

Unknown Design Unit used

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that specified design unit/ construct was not found in context.

Message Details

```
Specified <type> <name> does not exist
```

Severity

Syntax

STX_VH_498

Range of the actual must match with the formal

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate a mismatch in the range specification of formal parameter and the corresponding actual parameter. The range of formal and actual must match as per the language requirements.

Message Details

`Range of the actual does not match with the formal - <formal>`

Severity

Syntax

STX_VH_506

Aliasing of label, loop parameter, or generate parameter not allowed

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it found the following rule imposed by the language not being conformed.

An alias can be declared for all named entities except for labels, loop parameters, and generate parameters.

For Example:

```
l2: for i in 0 to 100 generate
alias al is i;              -- Error
```

Message Details

```
Cannot alias label or loop parameter or generate parameter
```

Severity

Syntax

STX_VH_512

Duplicates present in this target aggregate

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate presence of duplicates in target aggregates.

For Example:

```
      signal S  : aggsig := (bit'('0'), bit'('1'),
bit'('0'),bit'('1'));
      signal S1,S2,S3,S4 : bit;
(S1, S2, S1, S4) <= S;    --- ERROR : duplicate
-- specification of S1 in this target aggregate.
```

Message Details

```
Duplicates present in this target aggregate
```

Severity

Syntax

STX_VH_516

Cannot open dependency information file

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not open dependency information file.

Message Details

```
Could not open dependency information file <file-name>
```

Severity

Syntax

STX_VH_517

Implicit Function is not supported (by context)

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that implicit Function is not supported (by context).

Message Details

```
Implicit Function <function> is not supported
```

Severity

Syntax

STX_VH_518

Signal attributes can be associated with input ports only

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate presence of signal attributes associated with local constructs/identifiers. As per language specification, signal attributes can be associated with input ports only.

For Example:

```
entity E is
end;
architecture A of E is
component SUB
 port(AA : in integer;BB : out integer);
end component;
    signal A, B, C, D : integer;
begin
   C1: SUB port map(BB=>B'DELAYED(10 ns), AA=>A);
-- Error because B is not an input port
end A;
```

Message Details

```
Signal attributes can be associated with input ports only
```

Severity

Syntax

STX_VH_520

Cannot determine type of expression

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it could not determine the type of expression.

Suggested Hint

Qualified expression or Type-Casting may be used

Message Details

```
Cannot determine type of expression.
```

Severity

Syntax

STX_VH_521

Could not create DUMP file

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that DUMP file could not be created.

Message Details

```
Could not create DUMP file <file-name>
```

Severity

Syntax

STX_VH_522

Out of memory

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate detection of out of memory.

Message Details

```
Out of memory [<message>]. Please contact Atrenta Support.
Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VH_523

Generic map of component must match with previously created black box entity

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate mismatch between the generic map of component and the previously created black box entity.

Message Details

```
Generic Map of component does not match with previously created
black box entity <entity>
```

Severity

Syntax

STX_VH_524

Port map of component must match with the previously created black box entity

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate mismatch between the port map of component and the previously created black box entity

Message Details

```
Port Map of component does not match with the previously
created black box entity <entity>
```

Severity

Syntax

STX_VH_529

Output parameter not readable

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that output parameter could not be read.

For Example:

```
entity E is
end;
architecture A of E is
  procedure FF(x: out bit_vector) is
    variable j : bit;
   begin
     j := x(10);                    --Error
   end;
begin
  process
    begin
    wait;
  end process;
end A;
```

Message Details

Cannot read output parameter <parameter>

Severity

Syntax

STX_VH_530

Expression can only be associated with a port of mode IN

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that in the current context, expression could only be associated with a port of mode IN.

Message Details

```
Expression can only be associated with a port of mode IN.
```

Severity

Syntax

STX_VH_532

Design Unit used not found in logical library

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that it did not find that design unit used in logical library

Message Details

```
Design Unit <design-unit> not found in logical library
<library>
```

Severity

Syntax

STX_VH_537

Path for library not found

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that path of library was not found.

Message Details

```
Path for library '<library-path>' not found. Library
'<library1>' is in the dependency list of library '<library2>'
```

Severity

Syntax

STX_VH_539

OPEN is not a legal actual part of an element association for an input port without default expression

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate presence of an invalid usage of OPEN keyword. OPEN is not a legal actual part of an element association for an input port without default expression.

Message Details

```
OPEN is not a valid actual part of an element association for
an input port without default expression.
```

Severity

Syntax

STX_VH_543

Multiple design units with different port/generic-interface found

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate presence of an already defined/declared design unit with different port/generic-interface. When same design unit is specified more than once it is processed only once and warning is issued for the next one thus avoiding the reprocessing overhead

Message Details

```
<design-unit-type1> <design-unit-name1> is defined multiple
(port/generic-interface of this entity do not match to the
port/generic-interface of the <design-unit-type2> <design-unit-
name2> already encountered in File: "<file-name>" at Line
no:'<line-num>').
```

Severity

Syntax

STX_VH_544

Design Unit should not be defined more than once during sort

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that design unit should not be defined more than once during sort. When same design unit is specified more than once it is processed only once and warning is issued for the next one thus avoiding the reprocessing overhead.

Message Details

```
During sort, <design-unit-type1> <design-unit-name> is defined
multiple times. Previous <design-unit-type2> declaration found
in File: "<file-name>" at Line no:'<line-num>'.
```

Severity

Syntax

STX_VH_545

Unexpected end of Design Unit

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate an unexpected end of Design Unit.

Message Details

```
Unexpected end of Design Unit.
```

Severity

Syntax

STX_VH_548

Foreign module port expression must be an identifier. Bit or part select on formal is not allowed

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that foreign module port expression must be an identifier. Bit or part select on formal is not allowed as per language specification.

Message Details

```
Foreign module port expression must be an identifier. Bit or
part select on formal "<formal>" is not allowed
```

Severity

Syntax

STX_VH_551

Library Dump of Design Unit must be compatible with the current platform

Language

VHDL

Message Details

```
Library Dump (<platform1> bit) of <library-name>.<unit-name> is
not compatible with current platform (<platform2> bit)
```

Severity

Syntax

STX_VH_552

Library Dump of Design Unit must be compatible with the current platform

Language

VHDL

Message Details

```
Library Dump (<32 | 64> bit) of <library-name>.<primary-unit-
name>(<unit-name>) is not compatible with current platform
(<num> bit)
```

Severity

Syntax

STX_VH_555

OPEN is not a legal actual part of an element association for a subprogram call

Language

VHDL

Message Details

`OPEN is not a valid actual part of an element association for a subprogram call.`

Severity

Syntax

STX_VH_556

Formal associated with actual OPEN does not have default expression.

Language

VHDL

Message Details

```
Formal associated with actual OPEN does not have default
expression
```

Severity

Syntax

STX_VH_559

Re-declaration of package with different declarations found

Language

VHDL

Rule Description

Message Details

```
Package <package> already defined with different declaration(s)
in File: "<file-name>" at Line no:'<line-num>'
```

Severity

Syntax

STX_VH_560

Reconfiguration of component already configured in a block configuration

Language

VHDL

Message Details

```
Component <component> already configured in this block
configuration at Line no:'<line-num>')
```

Severity

Syntax

STX_VH_561

Each component local port must be associated in port map either in configuration specification or in configuration declaration if it is incremental binding

Language

VHDL

Message Details

```
Component local port (<port-name>) not associated in
<configuration-declaration/specification> (<component-label>:
<component-name>)
```

Severity

Syntax

STX_VH_562

An instance port connection has incompatible width compared to the port definition

Language

VHDL

Rule Description

SpyGlass throws syntax error when the size of a particular port is not same in component and instance.

For example, the following code snippet produces this syntax error:

```
entity mygate is
   port ( a: in std_logic_vector(0 to 1) --size 2
   );
end mygate;

entity top is
   port ( x : in std_logic_vector(2 to 5) --size 4
   );
end top;

architecture synth of top is
   component mygate is
      port
         a: in std_logic_vector(0 to 1)
      );
   end component;
begin
   U1: mygate port map(a=>x);
end top;
```

Message Details

```
During port map size <size1> of formal port '<formal-port>'
does not match size <size2> of actual port '<actual-port>'.
```

Severity

Syntax

STX_VH_565

Return type of subprogram call is not compatible with the type expected for target expression

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate a mismatch between the types of subprogram return type and the type expected of the target expression. Since VHDL is a strongly typed language, it requires that both the target and source expression belong to the exact same type, unless a type conversion function is specified.

In the following example, the return value of function `func1` is integer which is being assigned to expression `y` of type Boolean:

```
entity ent is
end ent;

architecture arch of ent is
begin
  test_1: process
    function func1
      (a1 : real; b1 : integer:= 12) return integer is
    begin
      return 5;
    end;
    variable y: boolean ;
    begin
   y := func1 (2.5, 10);
   wait;
 end process test_1;
end arch;
```

Suggested Fix

Use target expression and source subprogram call of same types in VHDL code.

Message Details

Return type mismatch for subprogram "<sub-program>". Expected type "<type>"

Severity

Syntax

STX_VH_566

Identifier used is ambiguous, having declarations at more than one visible place.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of more than one visible declaration or definition of an identifier. This erroneous situation usually comes into picture, when we include two different packages, through the 'use' clauses, and both packages having declaration or definition of the same identifier. In such scenario, none of declaration or definition is visible, and it is an error situation.

In the following example, the ARRAY_OF_INTEGER type is visible from two packages resulting in an STX_VH_566 error:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

PACKAGE pack_1 IS
  TYPE ARRAY_OF_INTEGER IS ARRAY (NATURAL RANGE <>)
    OF INTEGER;
END pack_1;

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

PACKAGE pack_2 IS
  TYPE ARRAY_OF_INTEGER IS ARRAY (NATURAL RANGE <>)
    OF INTEGER;
END pack_2;

LIBRARY work;
use work.pack_1.all;
use work.pack_2.all;

ENTITY ent is
```

```
end ent;

ARCHITECTURE arch of ent is
  CONSTANT c1  : ARRAY_OF_INTEGER(0 TO 5);
begin
end arch;
```

Message Details

```
Identifier '<identifier>' is ambiguous - more than one visible
```

Severity

Syntax

STX_VH_567

Physical path for a design unit in dependency file of some other design unit is different from the current mapping of <logical_lib> <physical_path>

Language

VHDL

Message Details

```
Physical path ("<library-path1>") corresponding to logical
library ("<library-name1>") for "<library-name2>.<design-unit-
name1>" in dependency info file of "<library-name3>.<design-
unit-name2>" is different from the current mapping of physical
path ("<library-path2>") for logical library ("<library-
name4>")
```

Severity

Syntax

STX_VH_570

Value should not be out of range of time (by context).

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the presence of a value, which is out of range as per the context in which it appears. It is an error to assign a value to a variable that is beyond the maximum allowable limit of base type of that variable. In case of time, the valid range is (-9223372036854775808) to 9223372036854775807 fs.

Message Details

```
Value <value> is outside the 64-bit range for the type time
```

Severity

Syntax

STX_VH_571

Type of range is ambiguous and two different valid types are visible.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the type of range is ambiguous as two different valid types are visible. For example, SpyGlass flags an error in the following case:

```
entity e is
end;
architecture a of e is
type myEnumType is ('A','B','C','D');
begin
  myForLabel : for I in 'A' to 'C' generate
  -- I is enum type
    end generate myForLabel;
end;
```

Message Details

```
Type of range is ambiguous, Visible types are '<type>' and
std.standard.character
```

Severity

Syntax

STX_VH_572

Invalid re-declaration of function via an alias to the same function in the same or different package.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the function or procedure is being redeclared via an alias. For example, SpyGlass flags an error in the following case:

```
package pack is
function add (a,b:bit) return bit;
end pack;

package body pack is
alias a is add [bit,bit return bit];
function a(a,b : bit) return bit is
  begin
    return (a xor b);
    end;
    end pack;
```

Message Details

```
Invalid re-declaration, function '<function1>', cannot be
declared via an alias to function '<function2>' in the same or
different package
```

Severity

Syntax

STX_VH_573

Mismatch of length during assignment of an array with an aggregate.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate the mismatch of length between array and aggregate, as shown in the following examples:

Example 1

```
entity ent is
  port(in1,in2: bit_vector(0 to 3);
  output : out bit_vector(0 to 3));
end;

architecture arch of ent is
  signal sig : bit_vector( 0 to 4):=(0=>'0',1 to
2=>'1',3=>'0');
begin
  sig <= in1 and in2;
  output<= sig;
end;
```

Example 2

```
-- Example of NULL RANGE
entity ent is
end ent;

architecture arch of ent is
begin
  P1: process
  type A1 is array (1 to 15) of integer;
  variable   XC : A1;
  begin
```

```
    XC (4 to 1) := (4,3,2,1);
-- mismatch of length due to NULL RANGE
    wait;
  end process P1;
end arch;
```

Message Details

```
Mismatch of length, array length is <array-length> whereas
aggregate length is <aggregate-length>
```

Severity

Syntax

STX_VH_574

Aggregate value is out of range.

Language

VHDL

Rule Description

SpyGlass generates this syntax error to indicate that the aggregate value was out of range. For example, SpyGlass flags the STX_VH_574 error in the following case:

```
entity ent is
end ent;
architecture arch of ent is
  subtype    A is    bit_vector (2 downto 1);
  constant   C :    A := (3 => '1', others => '0');
begin
end arch;
```

Message Details

```
Aggregate value <value> is out of range of <range> for
<declaration>
```

Severity

Syntax

STX_VH_603

Invalid assertion in PSL Boolean layer

Language

VHDL

Message Details

```
Invalid assertion in PSL Boolean layer
```

Severity

Syntax

STX_VH_614

Unexpected mode of parameter to conversion function

Language

VHDL

Rule Description

This error is generated by SpyGlass because the mode parameter to a conversion function can be either of out/linkage/buffer/inout modes.

For example, the following code will produce this syntax error:

```
entity ent is
port(portin: in bit;
     portout : out bit);
end;

architecture arch of ent is
component comp is
port(cpinIp :in bit;cpinOp: out bit)  ;
end component;

function func(fin1:bit) return bit is
begin
return '1';
end;

begin
l1: comp port map
(func(cpinIp)=>portin,cpinOp=>portout);
end;
```

Message Details

```
The PARAMETER '<parameter-name>' of mode <port-type> cannot be
argument to a conversion function
```

Severity

Syntax

STX_VH_621

The trailing underscore(s) and adjacent underscores are not allowed in any identifier

Language

VHDL

Rule Description

Identifiers with trailing underscore(s) and adjacent underscores are not allowed in VHDL. For example, SpyGlass flags the STX_VH_621 syntax error in the following case:

```
library ieee;
use ieee.std_logic_1164.all;

entity qlapbtestv1_ff0 is
port(h01_ : in std_logic; --Trailing underscore found
    h01____1 : in std_logic --Adjacent underscore found
    h01____ : in std_logic --Adjacent underscore found
);
end qlapbtestv1_ff0;

architecture RTL of qlapbtestv1_ff0 is
begin
  FF_00 :process (h01_) begin
  end process;
end RTL;
```

Message Details

```
<underscore> in the identifier '<identifier>
```

Severity

Syntax

STX_VH_622

Identifiers of size greater than 10240 are not allowed

Language

VHDL

Rule Description

Identifiers of size greater than 10240 are not allowed.

Message Details

```
Identifiers of size greater than 10240 are not allowed
```

Severity

Syntax

STX_VH_625

Internal Error raised when mismatch in the push pop count for memPlane swap stack

Language

VHDL

Rule Description

SpyGlass flags this Internal Error whenever there is a mismatch in the push pop count for the memory plane swap stack.

Message Details

```
Internal error: Mismatch of push(<push-count>) and pop(<pop-count>) count for memPlaneSwap stack. Please contact Atrenta Support. Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VH_626

Dependent Package CheckSum Error: Different version of a package used in the current session.

Language

VHDL

Rule Description

SpyGlass generates this dependent package checksum error to indicate that a design unit used a different package when it was compiled earlier, and in the current run it refers to a different package with the same name.

Consider an example in which there are two versions of a file, package.vhd. This file is compiled into a logical library `lib1` at two different physical locations. Now consider that you perform the following steps:

1. Compile package.vhd (version1) into logical library `lib1` at physical path `./dir1`:

```
package pack is
    CONSTANT VAL0 : NATURAL := 0;
end;


read_file -type vhdl package.vhd
set_option projectwdir run1
set_option work lib1
set_option lib lib1 ./dir1
```

2. Compile package.vhd (version 2) into logical library `lib1` at physical path `./dir2`:

```
package pack is
    CONSTANT VAL1 : NATURAL := 1;
    CONSTANT VAL0 : NATURAL := 0;
end;
```

```
read_file -type vhdl package.vhd
set_option projectwdir run1
set_option work lib1
set_option lib lib1 ./dir2
```

Compile middle.vhd in the libm library at the physical path ./dirm.

lib1 in this case is used from ./dir1 via set_option lib lib1 ./dir1

```
library lib1;
use lib1.pack.all;
entity ent is
    port(a : in  bit_vector (VAL0 downto 0));
begin
end ent;


architecture rtl of ent is
begin
end;


set_option work libm
set_option lib libm ./dirm
set_option projectwdir run3
read_file -type vhdl middle.vhd
set_option lib lib1 ./dir1
```

3. Elaborate the ent entity by using the top option.

   In this case, set_option lib lib1 ./dir2 and set_option lib libm ./dirm are used.

```
set_option top ent
set_option lib lib1 ./dir2
set_option lib libm ./dirm
set_option projectwdir run4
```

In the above scenario, because two different versions of the same library, lib1, is used (versions 1 and 2 above), external references of libm in SpyGlass dumps were created by using version1 and not version2. Therefore, the STX_VH_626 checkSum error is reported in this case.

Message Details

```
Checksum Error: The package '<package>' in the library
'<library>' is not the same when it was used earlier while
compiling <lib>
```

Severity

Warning

Suggested Fix

To remove this warning, specify correct library mappings by using the `set_option lib <logical-lib-name> <physical-path>` command in the project file.

STX_VH_627

Choice expression cannot be array type

Language

VHDL

Rule Description

SpyGlass flags this message if choice expressions in VHDL `case` statements have array type declarations.

For example, this rule reports a violation in the following case:

```
entity ent is
port(in1: bit_vector);
end;


architecture arch of ent is
  type  bit_temp_vector is array (integer range <>) of bit ;
  constant const : bit_temp_vector :="10";
  signal sig : integer;
begin
  process(in1)
  begin
    case const is
      when bit_temp_vector => sig <= 10;
      when others => sig <= 10;
    end case;
  end process;
end;
```

In the above example, the STX_VH_627 rule violation is reported for the usage of `bit_temp_vector` in the `case` statement.

Message Details

`Choice expression cannot be array type <type>`

Severity

Syntax

STX_VH_628

The elements of the aggregate must be of the same size when others is used for associating the same

Language

VHDL

Rule Description

SpyGlass flags this error to indicate a mismatch of length between elements of an aggregate for which `others` association is used.

Consider the following example (line numbers are also displayed):

```
1.library ieee;
2.use ieee.std_logic_1164.all;
3. entity test1 is
4. port(in1: bit;
5. in2: bit;
6. output: out bit); end;
7. architecture arch of test1 is
8. type rec is
9. record
10.f1: bit_vector(0 to 4);
11.f2: bit_vector (0 to 2);
12.end record;
13.   constant const : rec :=(others=> (1=>'0',
      others=> '1'));
        begin
         output<=const.f1(0) and in1;
        end;
```

This rule reports a violation in the above case.

You can fix this violation in any of the following ways:

- Change the size of `f2` in line number *11* to `(0 to 4)`, as shown below:

  ```
  f2: bit_vector (0 to 4);
  ```

- Specify an explicit mapping of elements in line number *13*, as shown below:

```
constant const : rec :=( (1=>'0',  others=>
'1'),(1=>'0',  others=> '1'));
```

Message Details

```
Size <record-size> of record element <record-element> does not
match with the expected size <expected-size>
```

Severity

Syntax

STX_VH_634

Library dump could not be restored for a design unit. Please recompile after removing the old dump.

Description

The *STX_VH_634* rule reports a violation to indicate that a library dump cannot be restored as it is corrupted.

Language

VHDL

Messages and Suggested Fix

This rule reports the following violation:

```
[SYNTAX] Dump of design unit may be corrupted and hence cannot be restored.
```

Consequences of Not Fixing

If you do not fix this violation by using a freshly recompiled library dump, you will not get the expected output from the design

How to Debug and Fix

To fix this violation, perform the following steps:

1. Remove the old dump (WORK and other dump libraries).
2. Recompile the design unit.

Default Severity Label

Syntax

STX_VH_638

There is a missing `protect end_protected directive

When to Use

Use this rule to check the correctness of a decryption envelope.

Description

The *STX_VH_638* rule reports a violation if the `protect end_protected directive is missing for the `protect begin_protected directive.

Messages and Suggested Fix

This rule reports the following message:

[SYNTAX] Missing "`protect end_protected" corresponding to "`protect begin_protected" directive. Please contact the vendor of this file for resolution of this problem.

Consequences of Not Fixing

If you do not fix this violation, the SpyGlass run does not proceed further.

How To Debug and Fix

Specify the missing `protect end_protected directive for the `protect begin_protected directive.

Example Code and/or Schematic

This rule reports a violation for the following example because of the missing `protect end_protected directive for the `protect begin_protected directive.

```
`protect begin_protected
`protect version = 1
`protect data_method = "aes128-cbc"
`protect key_keyowner = "Atrenta" , key_keyname =
 "ATR-SG-RSA-1" , key_method = "rsa"
`protect encoding = (enctype = "base64", line_length =
 64, bytes = 384), key_block
```

```
HcWZCZv/
LG+tKRHQ+bgrxjo4RMgLUrcGYujGD1RRhKkJJnNDHFSw3wkJufXemSyW
...
...
...
...
...
i0GN8bHJRNxPUYO/IAvpB6sA/3J2yENvsMSh2pgyv88=
```

Default Severity Label

Syntax

STX_VH_644

VHDL syntactic or semantic error / warnings detected in the encrypted source file

When to Use

Use this rule to check VHDL syntactic or semantic error/warnings.

Language

VHDL

Description

The *STX_VH_644* rule reports a violation if VHDL syntactic or semantic error/warnings are detected in the encrypted source file. The rule suppresses the actual error message for security reasons.

Messages and Suggested Fix

This rule reports the following message:

[SYNTAX] VHDL syntactic or semantic error/warnings detected in the encrypted source file. The actual error message has been suppressed for security reasons. Please contact the vendor of this file for resolution of this problem.

Consequences of Not Fixing

If you do not fix this violation, the SpyGlass run does not proceed further.

How To Debug and Fix

Contact the vendor of the encrypted file for resolving this violation.

Example Code and/or Schematic

This rule reports a violation for the following example because of the missing `protect end_protected directive for the `protect begin_protected directive.

```
`protect begin_protected
`protect version = 1
`protect data_method = "aes128-cbc"
`protect key_keyowner = "Atrenta" , key_keyname =
```

```
 "ATR-SG-RSA-1" , key_method = "rsa"
`protect encoding = (enctype = "base64", line_length =
 64, bytes = 384), key_block
 HcWZCZv/
 LG+tKRHQ+bgrxjo4RMgLUrcGYujGD1RRhKkJJnNDHFSw3wkJufXemSyW
 ...
 ...
 ...
 ...
 ...
 i0GN8bHJRNxPUYO/IAvpB6sA/3J2yENvsMSh2pgyv88=
```

Default Severity Label

Syntax

STX_VH_641

Element of an access type is not found

Description

The *STX_VH_641* rule reports if an element of an access type is not found. It can because memory is not properly allocated or the elaboration of this access type is not done.

For example:

If `X(Y downto 0)` is an element of record `Z` and `Y` is a generic whose value is coming from entity instance defined in top architecture. Then calling evaluation on the element `X`, without elaborating it, results in this elaboration error.

Language

VHDL

Messages and Suggested Fix

This rule reports the following violation:

[Internal Fatal] `Elaboration error: The element <element-name> could not be found. Please contact Atrenta Support.`

Default Severity Label

Internal fatal

Verilog Syntax Error Rules

Rules of this category report a violation for any syntax error in the Verilog code.

Most of the rules under this category are semantic errors in nature except the *STX_VE_481*, *STX_VE_477*, and *STX_VE_479* rules.

STX_VE_264

Illegal timescale unit is specified. Expected unit must be either of these - fs, ps, ns, us, ms, or s.

Language

Verilog

Rule Description

SpyGlass reports this syntax error if you specify an invalid timescale unit. You should specify only any of the following timescale units:

fs	ps	ns	us	ms	s

For example, the *STX_VE_264* rule reports a violation in the following case because an invalid timescale unit is specified:

```
`timescale 1 s/ 10 Fs
module top;
  ...
endmodule
```

Message Details

```
Illegal timescale unit ( <timescale-unit> ) specified
```

Severity

Syntax

STX_VE_266

Specified hierarchical reference cannot be resolved.

Language

Verilog

Message Details

`Cannot resolve hierarchical reference <reference>.`

Severity

Syntax

STX_VE_269

Incompatible declaration of an input port is found.

Language

Verilog

Message Details

Incompatible declaration, (<declaration>) defined as input at line no <line-num>.

Severity

Syntax

STX_VE_270

Incompatible declaration of an inout port is found.

Language

Verilog

Message Details

```
Incompatible declaration, ( <declaration> ) defined as inout at
line no <line-num>.
```

Severity

Syntax

STX_VE_272

The number of parameter assignments in instance is more than the number of parameters defined in the parent.

Language

Verilog

Rule Description

The number of parameter assignments in an instance should be less than or equal to the number of parameters defined in the parent. For example:

```
module mod;
  parameter p1 = 1;
endmodule

module mod1;
  mod MOD1();          // Fine
  mod #(7) MOD2();     // Fine
  mod #(7,10) MOD3();  // Error
endmodule
```

Message Details

```
The number of parameter assignments in instance ( <instance> )
is <num> more than the number of parameters defined in <parent>
```

Severity

Syntax

STX_VE_INACTIVE_272

The number of parameter assignments in instance is more than the number of parameters defined in the parent.

Language

Verilog

Rule Description

The number of parameter assignments in an instance should be less than or equal to the number of parameters defined in the parent. For example:

```
module mod;
  parameter p1 = 1;
endmodule

module mod1;
  mod MOD1();          // Fine
  mod #(7) MOD2();     // Fine
  mod #(7,10) MOD3();  // Error
endmodule
```

NOTE: *The STX_VE_INACTIVE_272 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
The number of parameter assignments in instance ( <instance> )
is <num> more than the number of parameters defined in <parent>
```

Severity

Syntax

STX_VE_274

The Param assignment list must be complete.

Language

Verilog

Message Details

`Param assignment list is incomplete.`

Severity

Syntax

STX_VE_275

No expression should be given for connection to null port.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that an expression is given for connection to null port. No expression should be given for connection to null port.

For example,

```
module TOP;
   wire w1, w2;
   child inst1(w1, w2);
endmodule

module child( , in1);
   input in1;
endmodule
```

In the above example, *w1* is an invalid connection to null port.

Message Details

`Expression given for connection to null port.`

Severity

Syntax

STX_VE_INACTIVE_275

No expression should be given for connection to null port.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that an expression is given for connection to null port. No expression should be given for connection to null port.

For example,

```
module TOP;
   wire w1, w2;
   child inst1(w1, w2);
endmodule

module child( , in1);
   input in1;
endmodule
```

In the above example, *w1* is an invalid connection to null port.

NOTE: *The STX_VE_INACTIVE_275 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Expression given for connection to null port.`

Severity

Syntax

STX_VE_276

A '!' cannot be used with a sequence/property.

Language

Verilog

Rule Description

A sequence cannot be negated and a property can be negated with the SystemVerilog `not` operator, and not with the '!' operator; so using a unary '!' with a sequence/property is illegal.

Message Details

```
! used with sequence/property '<name>'
```

Severity

Syntax

STX_VE_277

Specifying the port-range for a scalar net is not allowed.

Language

Verilog

Message Details

```
Port range specified for a scalar net (<net>) defined at file
(<file-name>), line (<line-num>).
```

Severity

Syntax

STX_VE_278

The value returned from an SV array query function can only be assigned to a variable of integer type.

Language

Verilog

Rule Description

The return type of SystemVerilog array query functions is of integer type, which denotes either some index or size which is predominantly integral.

Message Details

`Value returned from SV array query function '<function>' is not assigned to variable of integer type.`

Severity

Syntax

STX_VE_279

The left-hand and right-hand sides of an unpacked array copy must have identical layouts.

Language

Verilog

Rule Description

In order to directly copy multiple elements into an unpacked array, the element size and the number of dimensions copied must be the same.

Message Details

```
The left-hand and right-hand sides of an unpacked array copy
must have identical layouts.
```

Severity

Syntax

STX_VE_280

Improper list-assignment to unpacked array.

Language

Verilog

Rule Description

The assignment requires nested sets of brackets that exactly match the dimensions of the array.

Message Details

```
Improper list-assignment to unpacked array.
```

Severity

Syntax

STX_VE_282

Port name in instance is not found in the parent module/interface definition.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the port name in instance is not found in the parent module/interface definition. Every port name, used in named port association, must be declared in the parent module/interface definition.

Message Details

```
Port name ( <port-name> ) used in instance ( <instance> ) is
not found in the parent <parent> ( <definition> )
```

```
Port name ( <port-name> ) in library cell ( <cell> ) does not
match with supplied RTL description
```

Severity

Syntax

STX_VE_INACTIVE_282

Port name in instance is not found in the parent module/interface definition.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the port name in instance is not found in the parent module/interface definition. Every port name, used in named port association, must be declared in the parent module/interface definition.

NOTE: *The STX_VE_INACTIVE_282 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Port name ( <port-name> ) used in instance ( <instance> ) is
not found in the parent <parent> ( <definition> )
```

```
Port name ( <port-name> ) in library cell ( <cell> ) does not
match with supplied RTL description
```

Severity

Syntax

STX_VE_284

The number of parameter assignments in an instance must be equal to the number of parameters in the parent.

Language

Verilog

Message Details

Too many assignments for parameter (<parameter>) in instance (<instance>).

Severity

Syntax

STX_VE_INACTIVE_284

The number of parameter assignments in an instance must be equal to the number of parameters in the parent.

Language

Verilog

Message Details

`Too many assignments for parameter ( <parameter> ) in instance ( <instance> ).`

NOTE: *The SYNTH_VE_INACTIVE_284 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Severity

Syntax

STX_VE_286

The parameter name in an instance must match with the parent.

Language

Verilog

Message Details

```
Parameter name ( <name> ) in instance ( <instance> ) does not
match with parent.
```

Severity

Syntax

STX_VE_INACTIVE_286

The parameter name in an instance must match with the parent.

Language

Verilog

Message Details

`Parameter name ( <name> ) in instance ( <instance> ) does not match with parent.`

NOTE: *The STX_VE_INACTIVE_286 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Severity

Syntax

STX_VE_287

Packed Arrays cannot be assigned to Unpacked Arrays.

Language

Verilog

Rule Description

Packed arrays cannot be assigned to unpacked arrays without bit stream casting.

Message Details

```
Packed Arrays cannot be assigned to Unpacked Arrays.
```

Severity

Syntax

STX_VE_288

Improper assignment to an unpacked array.

Language

Verilog

Rule Description

Assignment to unpacked array consists of assignment to every element of every dimension specified.

Message Details

```
Improper assignment to unpacked array.
```

Severity

Syntax

STX_VE_289

Delay expression is invalid.

Language

Verilog

Message Details

```
Delay expression is invalid.
```

Severity

Syntax

STX_VE_290

An out-of-range dimension is specified for the array query function.

Language

Verilog

Rule Description

The dimension argument specified to an array query function must be valid, that is, not more than the number of dimensions of the array or type object upon which the function is queried.

Message Details

```
Out-of-range dimension specified for array query function
'<function>'.
```

Severity

Syntax

STX_VE_INACTIVE_290

An out-of-range dimension is specified for the array query function.

Language

Verilog

Rule Description

The dimension argument specified to an array query function must be valid, that is, not more than the number of dimensions of the array or type object upon which the function is queried.

NOTE: *The STX_VE_INACTIVE_290 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Out-of-range dimension specified for array query function
'<function>'.
```

Severity

Syntax

STX_VE_291

Type mismatch is detected for scope variable.

Language

Verilog

Message Details

```
Type mismatch for scope variable ( <variable> ).
```

Severity

Syntax

STX_VE_292

Type mismatch between actual argument and formal argument in task/function.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate a type mismatch between actual and formal arguments in task/function. Every argument specified in the declaration of a task/function must match exactly with the corresponding argument actually given when that task/function is called.

Message Details

```
Type mismatch between actual argument '<argument> and its
formal argument in <task | function> '<name>'.
```

Severity

Syntax

STX_VE_INACTIVE_292

Type mismatch between actual and formal arguments in task/function.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate a type mismatch between actual and formal arguments in task/function. Every argument specified in the declaration of a task/function must match exactly with the corresponding argument actually given when that task/function is called.

NOTE: *The STX_VE_INACTIVE_292 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Type mismatch between actual argument 'argument' and its formal argument in <task | function> '<name>'.`

Severity

Syntax

STX_VE_293

Recursive instantiation of an interface is not allowed.

Language

Verilog

Rule Description

Recursive instantiation of an interface is not allowed.

Message Details

```
Recursive instantiation ( <instantiation> ) of interface (
<interface> ).
```

Severity

Syntax

STX_VE_294

In a case statement, specifying multiple default items is not allowed.

Language

Verilog

Message Details

```
Multiple default items specified.
```

Severity

Syntax

STX_VE_295

Invalid access to a member of the interface

Language

Verilog

Rule Description

Invalid access to a member of the interface. Any member of the interface must be accessed either by instantiating the interface or via a port net.

Message Details

```
Invalid access to a member of the interface
```

Severity

Syntax

STX_VE_296

Improper assignment to a mixed array

Language

Verilog

Rule Description

A mixed array has both packed and unpacked dimensions, and any object assigned to it must have compatible data type, unpacked dimensions and packed dimensions.

Message Details

```
Improper assignment to mixed array.
```

Severity

Syntax

STX_VE_297

A UDP instance can have only two delay specifications.

Language

Verilog

Rule Description

Only two delays may be specified with a UDP instance because z is not supported for UDPs.

Message Details

```
UDP instance ( <instance> ) can have only two delay
specifications.
```

Severity

Syntax

STX_VE_INACTIVE_297

A UDP instance can have only two delay specifications.

Language

Verilog

Rule Description

Only two delays may be specified with a UDP instance because `z` is not supported for UDPs.

NOTE: *The STX_VE_INACTIVE_297 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
UDP instance ( <instance> ) can have only two delay
specifications.
```

Severity

Syntax

STX_VE_298

Improper assignment to a packed array

Language

Verilog

Rule Description

Objects with unpacked dimensions cannot be assigned directly to packed arrays. For that purpose, bit-stream casting has to be done.

Message Details

```
Improper assignment to packed array.
```

Severity

Syntax

STX_VE_299

Incompatible connection to a module/interface port

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of an incompatible connection to a module/interface port.

Ports passed to module/interface instances must be assignment-compatible with the objects specified in the respective module/interface declarations.

Message Details

```
Incompatible connection to <port | parameter> '<port/
parameter-name> parameter' of <module | interface> '<module/
interface-name>'.
```

Severity

Syntax

STX_VE_INACTIVE_299

Incompatible connection to a module/interface port

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of an incompatible connection to a module/interface port.

Ports passed to module/interface instances must be assignment-compatible with the objects specified in the respective module/interface declarations.

NOTE: *The STX_VE_INACTIVE_299 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Incompatible connection to <port | parameter> '<port/
parameter-name> parameter' of <module | interface> '<module/
interface-name>'.
```

Severity

Syntax

STX_VE_300

A const variable cannot be re-assigned

Language

Verilog

Message Details

```
Illegal re-assignment to const variable '<variable>'
```

Severity

Syntax

STX_VE_301

A void function cannot be used as an expression

Language

Verilog

Rule Description

A void function is a statement and cannot be used as an expression.

Message Details

```
Illegal use of void function '<function>' as expression
```

Severity

Syntax

STX_VE_302

Gate types such as "pullup" and "pulldown" must have only one connection.

Language

Verilog

Message Details

```
Gate type ( <type> ) should have only one connection.
```

Severity

Syntax

STX_VE_303

Gate types such as TRAN, RTRAN, TRANIF0, RTRANIF0, TRANIF1, and RTRANIF1 must have only two connections.

Language

Verilog

Message Details

```
Gate type ( <type> ) should have only two connections.
```

Severity

Syntax

STX_VE_304

Gate types such as AND, NAND, OR, NOR, XOR, and XNOR must have at least two connections.

Language

Verilog

Message Details

```
Gate type ( <type>) should have at least two connection.
```

Severity

Syntax

STX_VE_305

Gate types such as NOTIF0, NOTIF1, BUFIF0, and BUFIF1 can have only three connections.

Language

Verilog

Message Details

```
Gate type ( <type> ) can have only three connections.
```

Severity

Syntax

STX_VE_306

Gate types such as NMOS, PMOS, RNMOS, and RPMOS must have three connections.

Language

Verilog

Message Details

```
Gate type ( <type> ) should have three connections.
```

Severity

Syntax

STX_VE_307

Gate types such as CMOS and RCMOS must have four connections.

Language

Verilog

Message Details

`Gate type ( <type> ) should have four connections.`

Severity

Syntax

STX_VE_308

The port size for primitive gate array port connection must be equal to the size specified in the port definition.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the port size for primitive gate array port connection is not equal to the size specified in the port definition.

For example,

```
module test;
   wire [7:0] a;
   wire [7:0] b;

   not i2[5:0] (a,b);
endmodule
```

In the above example, size of connection a and b should be 6-bit, which is equal to gate array size [5:0]).

Message Details

```
Incorrect port size ( incorrect-size> ) for primitive (
<primitive> ) gate array port connection (<connection>). The
port size should be ( <correct-size> )
```

Severity

Syntax

STX_VE_INACTIVE_308

The port size for primitive gate array port connection must be equal to the size specified in the port definition.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the port size for primitive gate array port connection is not equal to the size specified in the port definition.

For example,

```
module test;
   wire [7:0] a;
   wire [7:0] b;

   not i2[5:0] (a,b);
endmodule
```

In the above example, size of connection a and b should be 6-bit, which is equal to gate array size [5:0]).

NOTE: *The STX_VE_INACTIVE_308 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Incorrect port size ( <incorrect-size> ) for primitive (
<primitive> ) gate array port connection (<connection>). The
port size should be ( <correct-size> )
```

Severity

Syntax

STX_VE_309

A void function cannot have any return value

Language

Verilog

Message Details

`Void function '<function>' cannot return anything`

Severity

Syntax

STX_VE_310

Return expression type should match function return type

Language

Verilog

Message Details

```
Type of return expression '<expression>' is incompatible with
the return type of function '<function>'
```

Severity

Syntax

STX_VE_311

Blocking assignment within expression used in continuous assignment/event expression

Language

Verilog

Rule Description

A blocking assignment within an expression cannot be used in any continuous assignment or in any event expression.

Message Details

```
Blocking assignment within expression used in continuous
assignment/event expression
```

Severity

Syntax

STX_VE_312

The 'return' statement cannot be used outside of a task or function

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate the use of return statement outside a task or a function. The return statement, as allowed in SystemVerilog, can only be used inside a task or a function.

Message Details

```
'return' statement used outside of task or function
```

Severity

Syntax

STX_VE_313

Access to item declarations within generate-for loops should be indexed, while to the named generate block which is not in a generate-for loop should be non-indexed.

Language

Verilog

Rule Description

Only indexed access is allowed for declarations within generate-for loops (using scope variables). Whereas, for a named generate block which is not in a generate-for loop, indexed access should not be used.

Message Details

```
<Scope/Array-of-scopes> '<variable-used-incorrectly-in-
hierarchical-reference>' in '<complete-hierarchical-variable-
name>' cannot be referenced <with | without> an index
```

Severity

Syntax

STX_VE_314

Both strengths cannot be highZ.

Language

Verilog

Message Details

```
Both strengths cannot be highZ.
```

Severity

Syntax

STX_VE_315

A non-void function should return some value

Language

Verilog

Rule Description

Only void functions can be used without a return value. For non-void functions, it is necessary to return a value.

Message Details

```
return from non-void function '<function>' must have a value expression
```

Severity

Syntax

STX_VE_316

Task should not return a value expression

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that a value expression is returned by a task.

Message Details

```
The 'return' from task '<task>' must not have a value
expression
```

Severity

Syntax

STX_VE_317

A two-input gate should have both drive-strength parts specified.

Language

Verilog

Message Details

`(<gate>) gate should have both drive-strength parts specified.`

Severity

Syntax

STX_VE_318

If any parameter declaration appears in the module/interface header then all the remaining parameters declared within the module/interface cannot be overridden. These are treated as localparams.

Language

Verilog

Rule Description

If any parameter declaration appears in the module/interface header, all the remaining parameters declared within the module/interface cannot be overridden. Such parameters are treated as localparams. For example:

```
module test;
  parameter p = 4;
  leaf #(.b(p),.a(p))i(); //cannot override parameter a
  ...
endmodule
module leaf#(parameter b = 2);
  parameter a = 1;
  ...
endmodule
```

Message Details

```
Illegal parameter override for parameter ( <parameter> ) in
instance ( <instance>)
```

Severity

Syntax

STX_VE_INACTIVE_318

If any parameter declaration appears in the module/interface header then all the remaining parameters declared within the module/interface cannot be override. These are treated as localparams.

Language

Verilog

Rule Description

If any parameter declaration appears in the module/interface header, all the remaining parameters declared within the module/interface cannot be overridden. Such parameters are treated as localparams. For example:

```
module test;
  parameter p = 4;
  leaf #(.b(p),.a(p))i(); //cannot override parameter a
  ...
endmodule
module leaf#(parameter b = 2);
  parameter a = 1;
  ...
endmodule
```

NOTE: *The STX_VE_INACTIVE_318 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Illegal parameter override for parameter ( <parameter> ) in
instance ( <instance> )
```

Severity

Syntax

STX_VE_321

Cannot specify 0 strength values for a PULLUP gate.

Language

Verilog

Message Details

```
Cannot specify 0 strength values for a PULLUP gate.
```

Severity

Syntax

STX_VE_322

Cannot specify 1 strength values for PULLDOWN gate.

Language

Verilog

Message Details

```
Cannot specify 1 strength values for PULLDOWN gate.
```

Severity

Syntax

STX_VE_326

More than two delays cannot be specified for gate types such as AND, NAND, NOR, OR, XOR, XNOR, BUF, NOT, TRANIF1, RTRANIF1, TRANIF0, and RTRANIF0.

Language

Verilog

Message Details

```
More than two delays cannot be specified for gate type ( <type>
).
```

Severity

Syntax

STX_VE_327

A delay cannot be specified with PULLUP or PULLDOWN source.

Language

Verilog

Message Details

```
Delay cannot be specified with PULLUP or PULLDOWN source.
```

Severity

Syntax

STX_VE_328

A delay cannot be specified for TRAN or RTRAN gate types.

Language

Verilog

Message Details

```
Delay cannot be specified for TRAN or RTRAN gate types.
```

Severity

Syntax

STX_VE_332

Gate output specification must be valid.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of an invalid gate output specification. Connection to the gate output port should be a valid out type net.

For example:

```
module test();
   wire a;
   wire b;
   not (1'b1,b);
endmodule
```

In the above example, invalid connection *1'b1* should be out type.

Message Details

```
Gate '<gate>' has invalid output specification for
'<specification>'
```

Severity

Syntax

STX_VE_INACTIVE_332

Gate output specification must be valid.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of an invalid gate output specification. Connection to the gate output port should be a valid out type net.

For example:

```
module test();
   wire a;
   wire b;
   not (1'b1,b);
endmodule
```

In the above example, invalid connection *1'b1* should be out type.

NOTE: *The STX_VE_INACTIVE_332 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Gate '<gate>' has invalid output specification for
'<specification>'
```

Severity

Syntax

STX_VE_334

Gate inout specification must be valid.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a connection to the gate inout port that is not a valid inout type net.

Message Details

```
<gate> gate has invalid inout specification.
```

Severity

Syntax

STX_VE_INACTIVE_334

Gate inout specification must be valid.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a connection to the gate inout port that is not a valid inout type net.

NOTE: *The STX_VE_INACTIVE_334 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`<gate> gate has invalid inout specification.`

Severity

Syntax

STX_VE_338

The input width must be same as the output width for the specified gate instance.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a gate instance in which input width is not same as the output width.

Message Details

`Input width is not same as output width for ( <instance> ) gate instance.`

Severity

Syntax

STX_VE_INACTIVE_338

The input width must be same as the output width for the specified gate instance.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a gate instance in which input width is not same as the output width.

NOTE: *The STX_VE_INACTIVE_338 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Input width is not same as output width for ( <instance> ) gate instance.`

Severity

Syntax

STX_VE_339

Control or enable input must be single-bit for the specified gate instance.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a gate instance for which the control or enable input is not single-bit.

Message Details

`Control or enable input should be single-bit for ( <instance> ) gate instance.`

Severity

Syntax

STX_VE_INACTIVE_339

Control or enable input must be single-bit for the specified gate instance.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a gate instance for which the control or enable input is not single-bit.

NOTE: *The STX_VE_INACTIVE_339 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Control or enable input should be single-bit for ( <instance> ) gate instance.`

Severity

Syntax

STX_VE_340

Only 4-state data type can be used to declare a net.

Language

Verilog

Rule Description

In SystemVerilog, only 4-state data type can be used to declare a net.

For example:

```
trireg (large) logic #(0,0,0) cap1;
typedef logic [31:0] addressT;
wire addressT w1;
wire int w2;
```

Message Details

```
Only 4-state data type can be used to declare a net
```

Severity

Syntax

STX_VE_341

Packed dimension is a must with VECTORED or SCALARED keyword.

Language

Verilog

Rule Description

There should be at least one packed dimension when `vectored` or `scalared` keyword is used.

For example:

```
wire vectored reg [3:0] a;
wire scalared logic [2:0] b;
wire vectored integer w1;
```

Message Details

```
At least one packed dimension is required with VECTORED or
SCALARED keyword
```

Severity

Syntax

STX_VE_342

A Verilog net type keyword shall not be followed directly by the REG keyword.

Language

Verilog

Rule Description

The REG keyword can be used in a net or port declaration if there are lexical elements between the net type keyword and the REG keyword.

For example:

```
tri reg a;
inout wire reg p;
wire vectored reg [3:0] a;
```

Message Details

```
A Verilog net type keyword cannot be followed directly by the
REG keyword
```

Severity

Syntax

STX_VE_343

Charge strength can only be used with the TRIREG keyword.

Language

Verilog

Rule Description

Charge strength can only be used with the TRIREG keyword.

For example:

```
trireg (large) logic a;    // Fine
wire (medium) integer s;    // Error
```

Message Details

```
Charge strength can only be used with the TRIREG keyword
```

Severity

Syntax

STX_VE_346

Invalid arguments are specified in task or function for ports in inout/output direction.

Language

Verilog

Rule Description

SpyGlass reports this syntax error to indicate that the expression specified in a function or task call for inout or output direction argument is an invalid lvalue.

For example, this rule reports a violation in the following case because the variable `a` is of type `wire` and cannot be assigned a value:

```
wire a;
function int f2(output x, input y);
  f2 = 10;
endfunction
assign abc = f2(a, 0);
```

However, this rule does not report a violation in the following case because the variable `a` is of type integer and can be assigned a value:

```
integer a;
function int f2(inout x, input y);
  f2 = 10;
endfunction
assign abc = f2(a, 0);
```

Message Details

```
Invalid argument (<argument-name>) to <function | task>
(<function/task-name>) (<inout | output>).
```

Severity

Syntax

STX_VE_INACTIVE_346

Invalid arguments are specified in task or function for ports in inout/output direction.

Language

Verilog

Rule Description

SpyGlass reports this syntax error to indicate that the expression specified in a function or task call for inout or output direction argument is an invalid lvalue.

For example, this rule reports a violation in the following case because the variable `a` is of type `wire` and cannot be assigned a value:

```
wire a;
function int f2(output x, input y);
  f2 = 10;
endfunction
assign abc = f2(a, 0);
```

In the above example, this rule reports a violation because the

However, this rule does not report a violation in the following case because the variable `a` is of type integer and can be assigned a value:

```
integer a;
function int f2(inout x, input y);
  f2 = 10;
endfunction
assign abc = f2(a, 0);
```

NOTE: *The STX_VE_INACTIVE_346 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Invalid argument (<argument-name>) to <function | task>
(<function/task-name>) (<inout | output>).
```

Severity

Syntax

STX_VE_347

Invalid task is specified.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that an invalid construct is specified in the context where a task was expected.

Message Details

```
( <task> ) is not a task
```

Severity

Syntax

STX_VE_INACTIVE_347

Invalid task is specified.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that an invalid construct is specified in the context where a task was expected.

NOTE: *The STX_VE_INACTIVE_347 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
( <task> ) is not a task
```

Severity

Syntax

STX_VE_348

Invalid function is specified.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that an invalid construct is specified in the context where a function was expected.

Message Details

```
( <function> ) is not a function.
```

Severity

Syntax

STX_VE_INACTIVE_348

Invalid function is specified.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that an invalid construct is specified in the context where a function was expected.

NOTE: *The STX_VE_INACTIVE_348 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`( <function> ) is not a function.`

Severity

Syntax

STX_VE_349

Task or function name must be defined.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the specified task or function name is undefined.

Message Details

`Task or function name ( <name> ) not defined.`

Severity

Syntax

STX_VE_INACTIVE_349

Task or function name must be defined.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the specified task or function name is undefined.

NOTE: *The STX_VE_INACTIVE_349 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Task or function name ( <name> ) not defined.`

Severity

Syntax

STX_VE_350

Specified component name is not a task or a block.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the argument to the `disable` statement is not a task or named block identifier.

Message Details

`Component name ( <name> ) not a task or block.`

Severity

Syntax

STX_VE_INACTIVE_350

Specified component name is not a task or a block.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the argument to the `disable` statement is not a task or named block identifier.

NOTE: *The STX_VE_INACTIVE_350 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Component name ( <name> ) not a task or block.`

Severity

Syntax

STX_VE_351

Specified function cannot enable a task.

Language

Verilog

Message Details

`Function cannot enable a task ( <task> ).`

Severity

Syntax

STX_VE_352

Recursion in a function must be avoided.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a recursive function. Recursive functions are not permitted. That means, a function shall not call itself.

Message Details

`Recursion found in function ( <function> ).`

Severity

Syntax

STX_VE_INACTIVE_352

Recursion in a function must be avoided.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a recursive function. Recursive functions are not permitted. That means, a function shall not call itself.

NOTE: *The STX_VE_INACTIVE_352 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Recursion found in function ( <function> ).`

Severity

Syntax

STX_VE_353

Recursion stack overflow must be avoided.

Language

Verilog

Message Details

```
Recursion stack overflow found in function evaluator.
```

Severity

Syntax

STX_VE_354

Invalid expression in type reference.

Language

Verilog

Rule Description

An expression that is used as an argument in a type reference should not contain any hierarchical references or references to elements of dynamic objects.

For example:

```
type(a) x1;
type(int) x2;
type(b.c) x3;
```

Message Details

`Invalid expression ( <expression> ) in type reference`

Severity

Syntax

STX_VE_355

A function cannot include delay (#) or event control (@, wait) statements.

Language

Verilog

Message Details

```
Function cannot include delay (#) or event control (@, wait)
statements.
```

Severity

Syntax

STX_VE_358

System Function must be called with correct number of arguments.

Language

Verilog

Rule Description

This error is issued if a system function is called with incorrect number of arguments. For example, $rose, $rose(), $rose(x,y,z), and so on

Message Details

```
Incorrect number of arguments passed to System Function (
<function> ).
```

Severity

Syntax

STX_VE_359

A system task/function must be called with proper arguments.

Language

Verilog

Message Details

`Invalid argument ( <argument> ) of system task/function.`

Severity

Syntax

STX_VE_360

The second argument to $sdf_annotate must be a module instance.

Language

Verilog

Message Details

```
Second argument to $sdf_annotate ( <name> ) must be module
instance.
```

Severity

Syntax

STX_VE_361

A procedural assignment statement cannot drive a net other than reg type.

Language

Verilog

Message Details

```
Procedural assignment statement cannot drive a net : <net>.
```

Severity

Syntax

STX_VE_INACTIVE_361

A procedural assignment statement cannot drive a net other than reg type.

Language

Verilog

Message Details

`Procedural assignment statement cannot drive a net : <net>.`

NOTE: *The STX_VE_INACTIVE_361 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Severity

Syntax

STX_VE_362

The left-hand side of a continuous assignment cannot be reg type.

Language

Verilog

Message Details

```
The left-hand-side of continuous assignment cannot be reg type.
```

Severity

Syntax

STX_VE_363

The left-hand-side of a continuous assignment must be valid.

Language

Verilog

Message Details

`The left-hand-side of the continuous assignment is invalid.`

Severity

Syntax

STX_VE_364

The left-hand side of an assignment must be valid.

Language

Verilog

Message Details

`Invalid left-hand-side of an assignment.`

Severity

Syntax

STX_VE_365

The left-hand side of an ASSIGN statement cannot be a net other than reg type.

Language

Verilog

Message Details

```
Invalid left-hand-side in ASSIGN statement.
```

Severity

Syntax

STX_VE_INACTIVE_365

The left-hand side of an ASSIGN statement cannot be a net other than reg type.

Language

Verilog

Message Details

```
Invalid left-hand-side in ASSIGN statement.
```

NOTE: *The STX_VE_INACTIVE_365 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Severity

Syntax

STX_VE_366

The data-type in a DEASSIGN statement cannot be a net other than reg type.

Language

Verilog

Message Details

```
Invalid data-type in DEASSIGN statement.
```

Severity

Syntax

STX_VE_INACTIVE_366

The data-type in a DEASSIGN statement cannot be a net other than reg type.

Language

Verilog

Message Details

`Invalid data-type in DEASSIGN statement.`

NOTE: *The STX_VE_INACTIVE_366 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Severity

Syntax

STX_VE_367

The left-hand side for a FORCE statement must be valid.

Language

Verilog

Message Details

```
Invalid left-hand-side for FORCE statement.
```

Severity

Syntax

STX_VE_INACTIVE_367

The left-hand side for a FORCE statement must be valid.

Language

Verilog

Message Details

`Invalid left-hand-side for FORCE statement.`

`Invalid data-type in DEASSIGN statement.`

NOTE: *The STX_VE_INACTIVE_367 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Severity

Syntax

STX_VE_368

The data type for a RELEASE statement must be valid.

Language

Verilog,

Message Details

`Invalid data-type for RELEASE statement.`

Severity

Syntax

STX_VE_INACTIVE_368

The data type for a RELEASE statement must be valid.

Language

Verilog,

Message Details

`Invalid data-type for RELEASE statement.`

NOTE: *The STX_VE_INACTIVE_368 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Severity

Syntax

STX_VE_369

A register must be specified to hold a loop counter.

Language

Verilog

Message Details

A register must be specified to hold loop counter (<counter>).

Severity

Syntax

STX_VE_371

Specparam is expected as identifier.

Language

Verilog

Message Details

```
Specparam expected as identifier ( <identifier> ).
```

Severity

Syntax

STX_VE_375

Bit-select on a scalar net is invalid.

Language

Verilog

Message Details

```
Bit-select on a scalar net is invalid.
```

Severity

Syntax

STX_VE_376

Part-select on a scalar net is invalid.

Language

Verilog

Message Details

```
Part-select on scalar net is invalid.
```

Severity

Syntax

STX_VE_378

Mismatch in number of items in an assignment pattern and its target.

Language

Verilog

Rule Description

This rule reports a violation if the number of items in an assignment pattern are not same as the lvalue for which it is used.

The following example shows some violating and non-violating cases of this rule:

```
module top;
  int a[2], b[2], c[2];
  assign a = '{1,1,1}; // Too many items in pattern
  assign b = '{1};     // Too few items in pattern
  assign c = '{1,1};
endmodule
```

Message Details

```
Too <number-of-items> assignment pattern items for given
assignment
```

Severity

Syntax

STX_VE_INACTIVE_378

Mismatch in number of items in an assignment pattern and its target.

Language

Verilog

Rule Description

This rule reports a violation if the number of items in an assignment pattern are not same as the lvalue for which it is used.

The following example shows some violating and non-violating cases of this rule:

```
module top;
  int a[2], b[2], c[2];
  assign a = '{1,1,1}; // Too many items in pattern
  assign b = '{1};     // Too few items in pattern
  assign c = '{1,1};
endmodule
```

NOTE: *The STX_VE_INACTIVE_378 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Too <number-of-items> assignment pattern items for given assignment
```

Severity

Syntax

STX_VE_379

Assignment pattern does not cover all items of the lvalue.

Language

Verilog

Rule Description

This rule reports a violation if the assignment pattern used for an array or a structure does not contain an assignment value for all indices of the array or all members of the structure.

For example, this rule reports a violation in the following cases:

```
int a[3] = '{0:1,1:2}; // Incomplete array literal
struct {int a; int b;} s1 = '{a:1}; // incomplete
                                    // structure literal
```

However, this rule does not report any violation in the following case:

```
int b[3] = '{0:1,1:2,2:3};
```

Message Details

```
Incomplete array/structure literal
```

Severity

Syntax

STX_VE_INACTIVE_379

Assignment pattern does not cover all items of the lvalue.

Language

Verilog

Rule Description

This rule reports a violation if the assignment pattern used for an array or a structure does not contain an assignment value for all indices of the array or all members of the structure.

For example, this rule reports a violation in the following cases:

```
int a[3] = '{0:1,1:2}; // Incomplete array literal
struct {int a; int b;} s1 = '{a:1}; // incomplete
                                    // structure literal
```

However, this rule does not report any violation in the following case:

```
int b[3] = '{0:1,1:2,2:3};
```

NOTE: *The STX_VE_INACTIVE_379 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Incomplete array/structure literal`

Severity

Syntax

STX_VE_381

Mismatch in number of items in unpacked array concatenation and its target.

Description

The *STX_VE_381* rule reports a violation if the number of items in an unpacked array concatenation is different from the lvalue for which it is used.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

```
Too <many | few> items in unpacked array concatenation for
given assignment"
```

How to Debug and Fix

Match the number of items in an unpacked array concatenation with the lvalue for which it is used.

Example Code and/or Schematic

The following example shows the violating and non violating cases of this rule:

```
module top;
int a[2], b[2], c[2];
assign a = {1,1,1};   // Too many items in concatenation
assign b = {1};       // Too few items in concatenation
assign c = {1,1};   // Fine
endmodule
```

Default Severity Label

Syntax

STX_VE_INACTIVE_381

Mismatch in number of items in unpacked array concatenation and its target.

Description

The *STX_VE_INACTIVE_381* rule reports a violation if the number of items in an unpacked array concatenation is different from the lvalue for which it is used.

NOTE: *The STX_VE_INACTIVE_381 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

```
Too <many | few> items in unpacked array concatenation for given assignment"
```

How to Debug and Fix

Match the number of items in an unpacked array concatenation with the lvalue for which it is used.

Example Code and/or Schematic

The following example shows the violating and non violating cases of this rule:

```
module top;
int a[2], b[2], c[2];
assign a = {1,1,1};   // Too many items in concatenation
assign b = {1};       // Too few items in concatenation
assign c = {1,1};   // Fine
endmodule
```

Default Severity Label

Syntax

STX_VE_382

Part-select expressions must have a constant index.

Language

Verilog

Message Details

```
Part-select expression (<expression>) should have a constant
index.
```

Severity

Syntax

STX_VE_384

Concatenations must not have un-sized based numbers.

Language

Verilog

Message Details

Concatenation must not have un-sized based number (<number>).

Severity

Syntax

STX_VE_386

The use of real type in a concatenation is invalid.

Language

Verilog

Message Details

```
Invalid use of (<type>) (real type) in concatenation.
```

Severity

Syntax

STX_VE_389

The specified expression uses values of type real in an invalid context.

Language

Verilog

Message Details

```
Expression uses real (<value>) in an invalid context.
```

Severity

Syntax

STX_VE_394

Invalid reference to memory is detected.

Language

Verilog

Message Details

```
Invalid reference to memory ( <memory> ).
```

Severity

Syntax

STX_VE_395

Invalid reference to array.

Description

The following example shows the violating and non violating case of this rule:

```
wire [3:0] a[3:0];
wire [3:0] b;
assign b = a;        // Violation
assign b = a[1];   // No violation
```

Language

Verilog

Message Details

```
Invalid reference to array ( <array> )
```

Severity

Syntax

STX_VE_INACTIVE_395

Invalid reference to array.

Description

The following example shows the violating and non violating case of this rule:

```
wire [3:0] a[3:0];
wire [3:0] b;
assign b = a;        // Violation
assign b = a[1];   // No violation
```

NOTE: *The STX_VE_INACTIVE_395 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Language

Verilog

Message Details

`Invalid reference to array ( <array> )`

Severity

Syntax

STX_VE_396

Invalid reference to an event is detected.

Language

Verilog

Message Details

```
Invalid reference to event ( <event> ).
```

Severity

Syntax

STX_VE_401

The right-hand side expression is not valid.

Language

Verilog

Message Details

```
Right-hand-side expression is not valid.
```

Severity

Syntax

STX_VE_402

This rule has been deprecated.

The violation of this rule indicated that a conditioning signal can only be a scalar net of single bit.

However, as per LRM, a conditioning signal can also be a vector net in which case the least significant bit of the vector net is used as a signal.

Therefore, this rule has been deprecated.

STX_VE_403

The specified module is not the top module.

Language

Verilog

Message Details

```
( <module> ) is not the top module.
```

Severity

Syntax

STX_VE_404

Invalid digit/symbol is present in the based number.

Language

Verilog

Message Details

```
Invalid digit in based number ( <digit/symbol> ).
```

Severity

Syntax

STX_VE_409

An edge-descriptor value other than '0X', '1X', '01', or '10' is invalid.

Language

Verilog

Message Details

```
Invalid edge-descriptor value.
```

Severity

Syntax

STX_VE_412

Only an event can be triggered.

Language

Verilog

Message Details

`Only events may be triggered, <name> is not an event.`

Severity

Syntax

STX_VE_413

Library dump is not compatible with the current platform.

Language

Verilog

Message Details

```
Library dump (<32 | 64> bit) of (<design-unit>) is not
compatible with current platform (<32 | 64> bit).
```

Severity

Syntax

STX_VE_414

Library dump is not compatible with the current version of SpyGlass.

Language

Verilog

Rule Description

SpyGlass reports this error while restoring a design unit that was dumped by using an earlier version of SpyGlass.

Restoring a design unit that was dumped by using an earlier version of SpyGlass may cause problems because of the changes in the Verilog analyzer object model between different SpyGlass releases.

Message Details

```
Library dump of ( <design-unit-name> ) lying at (
<library-path> ) is compiled with an earlier version of
SpyGlass and is not compatible with the current SpyGlass
version. Please recompile with the current SpyGlass version";
```

Severity

Syntax

STX_VE_415

Library dump of a design unit is out-of-date

Rule Description

SpyGlass flags this error while restoring a design unit (say A) which has a reference to another design unit (say B), and the design unit, B is recompiled after the design unit, A.

Language

Verilog

Message Details

```
Library dump of <design-unit> is out-of-date. Kindly refer to
WRN_1044 during the compilation step
```

Severity

Syntax

STX_VE_416

Only valid input-path must be specified.

Language

Verilog

Message Details

```
( <path> ) is not a valid input-path.
```

Severity

Syntax

STX_VE_417

Only valid output-path must be specified

Language

Verilog

Message Details

```
( <path> ) is not a valid output-path.
```

Severity

Syntax

STX_VE_418

A path that is not driven by a gate output is invalid.

Language

Verilog

Message Details

`Path ( <path> ) is not valid, because it is not driven by a gate output.`

Severity

Syntax

STX_VE_422

A multi-input-path list with vector is not allowed for parallel connect (=>).

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a multi-input-path list with vector for parallel connect (=>).

For example,

```
input[1:0] a, b;
output[1:0] x;
specify
   (a, b => x) = (30, 20);
endspecify
```

Message Details

```
Multi-input-path list with vector is not allowed for parallel
connect ( => )
```

Severity

Syntax

STX_VE_INACTIVE_422

A multi-input-path list with vector is not allowed for parallel connect (=>).

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a multi-input-path list with vector for parallel connect (=>).

For example,

```
input[1:0] a, b;
output[1:0] x;
specify
   (a, b => x) = (30, 20);
endspecify
```

NOTE: *The STX_VE_INACTIVE_422 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Multi-input-path list with vector is not allowed for parallel connect ( => )`

Severity

Syntax

STX_VE_423

More than one output in the output-path list is not allowed for parallel connect.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of more than one output in the output path list for parallel connect.

For example,

```
input[1:0] a, b;
output[1:0] x,y;
specify
   (a => x,y) = (30, 20);
endspecify
```

Message Details

```
More than one output in output-path list for parallel connect.
```

Severity

Syntax

STX_VE_INACTIVE_423

More than one output in the output-path list is not allowed for parallel connect.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of more than one output in the output path list for parallel connect.

For example,

```
input[1:0] a, b;
output[1:0] x,y;
specify
   (a => x,y) = (30, 20);
endspecify
```

NOTE: *The STX_VE_INACTIVE_423 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`More than one output in output-path list for parallel connect.`

Severity

Syntax

STX_VE_425

Identifier is not in a downward path.

Language

Verilog

Message Details

```
( <path> ) is not in a downward path.
```

Severity

Syntax

STX_VE_428

Register in specify block path.

Language

Verilog

Message Details

`Register ( <register> ) in specify block path.`

Severity

Syntax

STX_VE_430

Bit-select is not allowed for path delay value expression; specparam is expected.

Language

Verilog

Message Details

```
Bit-select not allowed for path delay value expression,
specparam expected.
```

Severity

Syntax

STX_VE_431

Part-select not allowed for path delay value expression; specparam is expected.

Language

Verilog

Message Details

```
Part-select not allowed for path delay value expression,
specparam expected.
```

Severity

Syntax

STX_VE_437

Input- and output-port size must match for parallel connect.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the input and output port sizes for parallel connect do not match.

Message Details

```
Size mismatch between input and output port for parallel
connect.
```

Severity

Syntax

STX_VE_INACTIVE_437

Input- and output-port size must match for parallel connect.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the input and output port sizes for parallel connect do not match.

NOTE: *The STX_VE_INACTIVE_437 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Size mismatch between input and output port for parallel connect.`

Severity

Syntax

STX_VE_444

High-impedance as the initial value for UDP is invalid.

Language

Verilog

Message Details

`Invalid initial value for UDP.`

Severity

Syntax

STX_VE_445

Any statement other than assign or block-assignment as initial statement for UDP is invalid.

Language

Verilog

Message Details

```
Invalid initial statement for UDP.
```

Severity

Syntax

STX_VE_446

An unassigned enumerated name should not follow an enum name with x or z assignment.

Language

Verilog

Rule Description

For enum, the values assigned to the elements of the enumerated list, are by default started from 0 to whatever the number is. If the list has any of the elements initialized to a specific value, the subsequent (un-initialized) elements are assigned values incremented by 1.

For example, in the following case a=0,b=7,c=8,d=10.

```
enum {a,b=7,c,d=10}values;
```

If an element is assigned x or z, the unassigned element that follows it would not have a specifically defined value to take as x and z are not numbers that can be incremented.

So the following statement is incorrect as the values to be taken by c and e are indeterminate:

```
enum {a=0,b=x,c,d=z,e}value; //INCORRECT
```

Message Details

```
An unassigned enumerated name ( <name> ) should not follow an
enum name with x or z assignment.
```

Severity

Syntax

STX_VE_447

An enumerated name cannot have a value that is assigned to a previous member of the enumeration.

Language

Verilog

Rule Description

For enum, the values assigned to the elements of the enumerated list, are by default started from 0 to whatever the number is. We can have elements initialized to a specific value too. But, in that case the values should be unique.

For example, in the following case, a=0,b=2,c=3 and d=3 which is incorrect:

```
enum {a,b=2,c,d=3}values; //INCORRECT
```

Message Details

```
Enumerated name ( <name> ) cannot have a value (<value>) which
is assigned to a previous member of the enumeration.
```

Severity

Syntax

STX_VE_448

An x or z assignment should be used only for 4-state enumerations.

Language

Verilog

Rule Description

It is an error to assign an element of the enumerated list a value that has a different size than that declared for the enumerated type. By default, the type is taken to be int (32-bit, 2-state data type).Thus, the data type should be changed to a 4-state data type to be able to assign x or z values to it. Therefore, the following example is incorrect:

```
enum {a=0,b=1,c=1'bz}value;
```

One of the correct ways would be:

```
enum logic {a=0,b=1,c=1'bz}value;
```

Message Details

```
An x or z assignment for <name> should be used only for 4-state
enumerations.
```

Severity

Syntax

STX_VE_449

The size of the enumeration constant is more than the range specified for the enumeration.

Language

Verilog

Rule Description

It is an error to assign an element of the enumerated list a value that has a different size than that declared for the enumerated type.

In the following example, the size of the enum is taken to be 2 (by default) but the size of the constants is 3 which is incorrect:

```
enum {a=3'b001,b=3'b010,c=3'b100}values;
```

the correct way of doing the same would be:

```
enum bit[2:0] {a=3'b001,b=3'b010,c=3'b100}values;
```

This rule is specifically for cases where base numbers are used, for example, `3'b101`.

Message Details

```
Size of the enumeration constant ( <constant> ) for enum label
<label> is more than the range specified for the enumeration.
```

Severity

Syntax

STX_VE_450

An unpacked member is not allowed in a packed structure/union.

Language

Verilog

Rule Description

A packed structure/union cannot contain unpacked structures, unpacked unions, or unpacked arrays.

This is essentially because the purpose of a packed structure/union is that the memory allocated is contiguous; an unpacked member defeats this purpose.

For example, consider the following structure:

```
struct packed {
   logic x[3:0];
   } a;
```

The above structure contains a member that has unpacked dimensions. Therefore, the *STX_VE_450* rule reports a violation.

Message Details

```
Unpacked member ( <member> ) found in packed <structure |
union> (<name>).
```

Severity

Syntax

STX_VE_INACTIVE_450

An unpacked member is not allowed in a packed structure/union.

Language

Verilog

Rule Description

A packed structure/union cannot contain unpacked structures, unpacked unions, or unpacked arrays.

This is essentially because the purpose of a packed structure/union is that the memory allocated is contiguous; an unpacked member defeats this purpose.

For example, consider the following structure:

```
struct packed {
   logic x[3:0];
   } a;
```

The above structure contains a member that has unpacked dimensions. Therefore, the *STX_VE_INACTIVE_450* rule reports a violation.

NOTE: *The STX_VE_INACTIVE_450 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Unpacked member ( <member> ) found in packed <structure | union> (<name>).`

Severity

Syntax

STX_VE_451

Packed union members should have the same size.

Language

Verilog

Rule Description

In a packed union, the number of bits of each union member must be the same. This ensures that a packed union will represent its storage with the same number of bits, regardless of member in which a value is stored.

Consider the following example:

```
union packed {
  logic x;
  integer b;
  } a;
```

In the above example, two members of the union are not of the same size. The integer has the width of 32 bits while the logic is the 1 bit data member. Therefore, the *STX_VE_451* rule reports a violation.

Message Details

```
Packed union members should have same size. Member ( <member1>
) differs in size from member ( <member2> ).
```

Severity

Syntax

STX_VE_INACTIVE_451

Packed union members should have the same size.

Language

Verilog

Rule Description

In a packed union, the number of bits of each union member must be the same. This ensures that a packed union will represent its storage with the same number of bits, regardless of member in which a value is stored.

Consider the following example:

```
union packed {
  logic x;
  integer b;
  } a;
```

In the above example, two members of the union are not of the same size. The integer has the width of 32 bits while the logic is the 1 bit data member. Therefore, the *STX_VE_INACTIVE_451* rule reports a violation.

NOTE: *The STX_VE_INACTIVE_451 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Packed union members should have same size. Member ( <member1> ) differs in size from member ( <member2> ).`

Severity

Syntax

STX_VE_452

A void member cannot be declared here.

Language

Verilog

Rule Description

It is invalid to use members of the type void in structures/unions.

Consider the following example:

```
struct packed {
void a;
}b;
```

For the above example, the *STX_VE_452* rule reports a violation because void members are not allowed in a packed structure.

Message Details

`void member cannot be declared here.`

Severity

Syntax

STX_VE_INACTIVE_452

A void member cannot be declared here.

Language

Verilog

Rule Description

It is invalid to use members of the type void in structures/unions.

Consider the following example:

```
struct packed {
void a;
}b;
```

For the above example, the *STX_VE_INACTIVE_452* rule reports a violation because void members are not allowed in a packed structure.

NOTE: *The STX_VE_INACTIVE_452 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`void member cannot be declared here.`

Severity

Syntax

STX_VE_453

A size mismatch is found between enumeration value and enumeration range.

Language

Verilog

Rule Description

It is an error to assign an element of the enumerated list a value that has a different size than that of the enumeration range. By default, the type is taken to be int (32-bit, 2-state data type).Thus, the data range should be changed according to the value desired.

Therefore, the following example is INCORRECT:

```
enum {a=0,b=1,c=x,d=z}value;
```

One of the correct ways would be:

```
enum bit [3:0] {a=0,b=1,c=x,d=z}value;
```

Message Details

```
Size mismatch in enumeration value ( <value> ) for the
enumerated literal <literal>.
```

Severity

Syntax

STX_VE_454

An enumeration label cannot have negative values.

Language

Verilog

Rule Description

It is invalid to have negative values for enumerations. For example, the following is INCORRECT:

```
enum {a=2,b=-3}values; //INCORRECT
```

Message Details

```
Enumeration label <label> cannot have negative values.
```

Severity

Syntax

STX_VE_455

Packed dimensions are not allowed on this type. Packed dimensions are only allowed on types resolving to single-bit types, packed arrays, packed structures, and packed unions.

Language

Verilog

Rule Description

Packed dimension can only be used on types resolving to single-bit types (bit, logic, reg), net_type data types, packed arrays, packed structures, and packed unions. Using packed dimensions in any other context is invalid. For example:

```
int [3:0] a;
typedef bit BIT;
BIT [2:0] b;
typedef struct
{
int a;
} str;
str [3:0] c;
```

Message Details

```
Packed dimensions not allowed on this type. Packed dimensions
are only allowed on types resolving to single-bit types, packed
arrays, packed structures, and packed unions.
```

Severity

Syntax

STX_VE_456

Expression used for sized casting must be constant

Language

Verilog

Rule Description

The STX_VE_456 rule reports a violation when expressions used for sized casting are not constant.

For example:

```
 module top;
 parameter p = 3;
 int a,b;
 struct packed { int a;} s;
assign s = p'(a);    // Fine
 assign s = b'(a);      // Error
 endmodule
```

Message Details

`Expression (<expression>) used for sized casting is non-constant`

Severity

Syntax

STX_VE_INACTIVE_456

Expression used for sized casting must be constant

Language

Verilog

Rule Description

The STX_VE_INACTIVE_456 rule reports a violation when expressions used for sized casting are not constant.

For example:

```
module top;
parameter p = 3;
int a,b;
struct packed { int a;} s;
assign s = p'(a);    // Fine
assign s = b'(a);      // Error
endmodule
```

Message Details

`Expression (<expression>) used for sized casting is non-constant`

Severity

Syntax

STX_VE_459

Undefined forward typedef; The typedef is not defined in the scope where it is declared.

Language

Verilog

Rule Description

The scope of the typedef is restricted to the block in which it has been defined. Here, the typedef is being used outside the scope of its definition.

Message Details

```
Undefined forward typedef. The typedef (<typedef>) is not
defined in the scope where it is declared.
```

Severity

Syntax

STX_VE_460

Macro names may not be the same as compiler directive keywords.

Language

Verilog

Rule Description

Macro names may not be the same as compiler directive keywords.

For example, in the following statement, `include` is an illegal macro name:

```
`define include 5
```

Message Details

```
Compiler directive '<directive>' cannot be used as a macro name
```

Severity

Syntax

STX_VE_461

Lifetime specification (automatic/static) is not allowed for variable declarations in the module scope.

Language

Verilog

Rule Description

The variables declared at module level cannot be explicitly declared as static or automatic as at the module level, all variables are static.

Message Details

```
Lifetime specification (automatic/static) is not allowed for
variable declarations in module scope.
```

Severity

Syntax

STX_VE_462

Expecting assignment pattern in place of concatenation.

Language

Verilog

Rule Description

Concatenation cannot be used for the assignment of arrays or structures.

An assignment pattern/assignment pattern expression is expected in place of concatenation.

SpyGlass allows concatenation to be used for the assignment of arrays or structures only if both the following conditions are true:

- The width of concatenation is same as that of its target.
- The `allow_non_lrm` project-file command is set to `yes`.

Specify the `set_option enableSV09 yes` option to use the concentration as unpacked array concatenation, as per the 1800-2009 standard.

The following example shows the violating and non violating cases of this rule:

```
module top;
int a[4] = {1,2,3,4}; // Violation (As per standard)
                      // No violation if
                     // set_option allow_non_lrm yes is given
int b[4] = '{1,2,3,4};  // No violation
int c[4] = {1,2,3,4};    //No violation as per 1800-2009 standard

endmodule
```

Message Details

```
Illegal assignment, expecting assignment pattern. Please use
'set_option enableSV09 yes' to use unpacked array concatenation
```

Severity

Syntax

STX_VE_INACTIVE_462

Expecting assignment pattern in place of concatenation.

Language

Verilog

Rule Description

Concatenation cannot be used for the assignment of arrays or structures.

An assignment pattern/assignment pattern expression is expected in place of concatenation.

SpyGlass allows concatenation to be used for the assignment of arrays or structures only if both the following conditions are true:

- The width of concatenation is same as that of its target.
- The `allow_non_lrm` project-file command is set to `yes`.

Specify the `set_option enableSV09 yes` option to use the concentration as unpacked array concatenation, as per the 1800-2009 standard.

The following example shows the violating and non violating cases of this rule:

```
module top;
int a[4] = {1,2,3,4}; // Violation (As per standard)
                      // No violation if
                     // set_option allow_non_lrm yes is given
int b[4] = '{1,2,3,4};  // No violation
int c[4] = {1,2,3,4};    //No violation as per 1800-2009 standard

endmodule
```

NOTE: *The STX_VE_INACTIVE_462 rule is flagged when the set_option enable_inactive_rtl_checks yes command is used in the project file.*

Message Details

```
Illegal assignment, expecting assignment pattern. Please use
'set_option enableSV09 yes' to use unpacked array concatenation
```

Severity

Syntax

STX_VE_463

Only positional notation is allowed when an assignment pattern is used on the left hand side of an assignment.

Language

Verilog

Rule Description

Only positional notation is allowed when an assignment pattern/ assignment pattern expression is used on the left hand side of assignment-like expression.

Message Details

```
Only positional notation is allowed when an assignment pattern
is used on the left hand side of an assignment.
```

Severity

Syntax

STX_VE_464

An assignment pattern in assignment-like expression requires other side to be self-determined data type.

Language

Verilog

Rule Description

An assignment pattern has no self-determined data type. It can be used in an assignment-like expression only if the other side has self-determined data type.

Message Details

```
Invalid use of assignment pattern.
```

Severity

Syntax

STX_VE_465

An unpacked variable cannot be used as a whole with arithmetic and logical operators.

Language

Verilog

Rule Description

Arithmetic and logical operators can be applied only on packed variables as packed variables are stored as a vector, and operations on the complete variable is treated as a vector operation.

However, unpacked structures cannot be used as a whole with arithmetic and logical operators.

Message Details

```
Unpacked <variable> cannot be used as a whole with arithmetic
and logical operators.
```

Severity

Syntax

STX_VE_466

real, shortreal, or realtime members are not allowed in packed structure/union.

Language

Verilog

Rule Description

A structure/union cannot be packed if any of its members cannot be represented as a vector. So, real/shortreal/realtime members are not allowed in packed structure/union.

Consider the following example:

```
struct packed {
real a;
}b;
```

For the above example, the *STX_VE_466* violation appears because the members of type real, shortreal, or realtime are not allowed in a packed structure.

Message Details

```
Real/shortreal/realtime member ( <member> ) found in packed
<structure | union> (<name>).
```

Severity

Syntax

STX_VE_INACTIVE_466

real, shortreal, or realtime members are not allowed in packed structure/union.

Language

Verilog

Rule Description

A structure/union cannot be packed if any of its members cannot be represented as a vector. So, real/shortreal/realtime members are not allowed in packed structure/union.

Consider the following example:

```
struct packed {
real a;
}b;
```

For the above example, the *STX_VE_466* violation appears because the members of type real, shortreal, or realtime are not allowed in a packed structure.

NOTE: *The STX_VE_INACTIVE_466 rule is flagged when the set_option enable_inactive_rtl_checks yes command is used in the project file.*

Message Details

```
Real/shortreal/realtime member ( <member> ) found in packed
<structure | union> (<name>).
```

Severity

Syntax

STX_VE_467

Non-equivalent data types are found in assignment operation.

Language

Verilog

Rule Description

The data types on either side of the assignment operator do not match.

Message Details

```
Non-equivalent data types in assignment operation.
```

Severity

Syntax

STX_VE_468

String concatenation is not allowed on the left-hand side of an assignment.

Language

Verilog

Rule Description

String concatenation cannot be the lvalue of an expression.

Message Details

```
String concatenation is not allowed on the left hand side of an
assignment, only as an expression.
```

Severity

Syntax

STX_VE_469

Non-equivalent data types cannot be casted.

Language

Verilog

Rule Description

The data types on either side of the static cast operator do not match.

Message Details

```
Non-equivalent types cannot be casted.
```

Severity

Syntax

STX_VE_471

End of pragma is expected after tokens like 'synthesis_on' and 'translate_on'.

Language

Verilog

Message Details

`Syntax error after token ( <token> ), end of pragma expected.`

Severity

Syntax

STX_VE_472

Escaped names are not supported in the OVL file.

Language

Verilog

Rule Description

It is illegal to used escaped names (starting with "/" and ending in /n or /t or <space> etc) in OVL files.

Message Details

```
Escaped names are not supported in OVL file.
```

Severity

Syntax

STX_VE_473

The expression uses chandle / event variables in an invalid context.

Language

Verilog

Rule Description

chandle/event variables can only be used with Equality (==),Inequality (!=),Case equality (===), Case inequality (!==) operators. It is invalid to use it with any other operators.

Message Details

```
Expression uses chandle/event (<variable>) in an invalid
context.
```

Severity

Syntax

STX_VE_474

Using chandle as ports is not allowed

Language

Verilog

Rule Description

It is illegal to use chandle as ports.

Message Details

```
Use of chandle as ports not allowed
```

Severity

Syntax

STX_VE_475

Non-equivalent data types cannot be compared.

Language

Verilog

Rule Description

The data types on either side of the comparison operator do not match.

Message Details

`Non-equivalent data types cannot be compared.`

Severity

Syntax

STX_VE_476

Multiple bits of an unpacked array cannot be used with arithmetic or logical operators.

Language

Verilog

Rule Description

Arithmetic and logical operators can be applied only on packed variables because packed variables are stored as a vector, so multiple bits of an unpacked array cannot be used with arithmetic or logical operators.

Message Details

```
Multiple bits of an unpacked array ( <array> ) cannot be used
with arithmetic or logical operators
```

Severity

Syntax

STX_VE_477

A Verilog 2001 construct cannot be used with "set_option disablev2k yes".

Language

Verilog

Rule Description

Please run without specifying the following project file command to support Verilog 2001 constructs:

```
set_option disablev2k yes
```

Message Details

```
Syntax error near ( <identifier> ). Please run without
'set_option disablev2k yes' to support the Verilog 2001 IEEE
1364-2001 construct ( <construct> )
```

Severity

Syntax

STX_VE_478

Vector reference is not allowed on a scalar variable.

Language

Verilog

Rule Description

Bit select and part select are only allowed on vector variables/parameters/nets.

The following example shows the violating and non violating cases of this rule:

```
module test;
int i;
bit b;
assign b[1] = 0;    // Violation
assign i[1] = 0;  // No violation
endmodule
```

Message Details

`Invalid vector reference to scalar <variable>.`

Severity

Syntax

STX_VE_INACTIVE_478

Vector reference is not allowed on a scalar variable.

Language

Verilog

Rule Description

Bit select and part select are only allowed on vector variables/parameters/nets.

The following example shows the violating and non violating cases of this rule:

```
module test;
int i;
bit b;
assign b[1] = 0;    // Violation
assign i[1] = 0;   // No violation
endmodule
```

NOTE: *The STX_VE_INACTIVE_478 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Invalid vector reference to scalar <variable>.`

Severity

Syntax

STX_VE_479

A SystemVerilog construct cannot be used without specifying 'set_option enableSV yes'

Language

Verilog

Rule Description

A SystemVerilog construct cannot be used without specifying the `set_option enableSV yes` command in the project file.

Message Details

```
Syntax error near ( <identifier> ). Please use 'set_option
enableSV yes' to support SystemVerilog construct ( <construct>
)
```

Severity

Syntax

STX_VE_480

Port list must not contain both ordered and named association simultaneously.

Language

Verilog

Message Details

```
Port list contains both ordered and named association.
```

Severity

Syntax

STX_VE_481

A syntax error has been detected near the mentioned token.

Language

Verilog

Message Details

```
Syntax error near ( <token> ).
```

Severity

Syntax

STX_VE_482

A syntax error is detected before the mentioned token.

Language

Verilog

Message Details

Syntax error before token (<token>).

Severity

Syntax

STX_VE_483

Index bound for vectors must be integers.

Language

Verilog

Message Details

```
Index bound for vectors must be integers.
```

Severity

Syntax

STX_VE_484

Parameter list must not contain both ordered and named association simultaneously.

Language

Verilog

Rule Description

Parameter list should not contain a mix of ordered and named associations.

For example:

```
module top;
mod #(10, 12, 14) MOD1();           // Fine
mod #(.par1(7), .par2(8)) MOD2();   // Fine
mod #(10, .par1(7)) MOD3();         // Error
endmodule
```

Message Details

```
Parameter list contains both ordered and named association
```

Severity

Syntax

STX_VE_485

The specified include-file could not be found or opened in read mode.

Language

Verilog

Message Details

```
Include file ( <file> ) could not be found or opened in read
mode from current working directory ( <directory> ) or other
include directory paths (if any).
```

Severity

Syntax

STX_VE_486

Recursive include must be avoided.

Language

Verilog

Message Details

`Recursive include detected for file ( <file> ). Trace: (<trace>)`

Here, *<trace>* provides a back reference information for the recursive include files.

Severity

Syntax

STX_VE_487

Digits must follow decimal point in real.

Language

Verilog

Rule description

Digits must follow decimal point in real. For example, SpyGlass reports the STX_VE_487 error in the following case:

```
parameter p = 9.;
```

However, no violation is flagged for the following case:

```
parameter p = 9.0;
```

Message Details

```
Expecting a digit after a decimal point
```

Severity

Syntax

STX_VE_488

A " must be specified in the include directive.

Language

Verilog

Message Details

```
Missing " in include directive.
```

Severity

Syntax

STX_VE_489

Symbol used in delay or parameter assignment must be a constant or a parameter.

Language

Verilog

Rule Description

This warning was generated by SpyGlass because SpyGlass could not find all statements within the force on-off directives preceded by the comment marker '--'.

Message Details

`Symbol ( <symbol> ) must be a constant or a parameter.`

Severity

Syntax

STX_VE_490

Deep nesting of the include file must be avoided.

Language

Verilog

Rule Description

Message Details

```
Include files nested too deeply.
```

Severity

Syntax

STX_VE_491

Part-select direction does not match declared direction of the variable.

Language

Verilog

Rule Description

If a variable is declared with the left bound greater than the right bound, the part-select must also use a left bound greater than or equal to the right bound, and vice versa.

For example:

```
module test(x, y);
  input [2:0] x;
  output [1:0] y;
  assign y = x[1:2];
endmodule
```

In the above example, part-select, `x[1:2]`, does not match the declared direction of `x`.

Message Details

```
Bounds of part-select ( <part-select> ) are reversed
```

Severity

Syntax

STX_VE_492

Mentioned construct is not as per IEEE standards

Language

Verilog

Rule Description

The STX_VE_492 rule reports a violation when a non LRM construct is used in code.

To use non LRM construct, specify the following project-file command:

```
set_option allow_non_lrm yes
```

You should set the above command for the following non LRM features:

- The `final` construct, which is not an IEEE 1800-2009 construct.

  The following example shows the usage of this construct:

  ```
  a1: assert final (out1 == ~in1)
  $fatal (2,"System Task FATAL executed");
  ```

- An assignment pattern without an apostrophe is not a part of the LRM standard.

  The following example shows such pattern:

  ```
  int a[4] = {default:0};
  ```

Message Details

```
Syntax error near ( <identifier> ). Please use 'set_option
allow_non_lrm yes' to allow ( <feature> ) NON-LRM feature
```

Severity

Syntax

STX_VE_493

Mentioned construct cannot be processed through built-in method

Language

Verilog

Rule Description

The *STX_VE_493* rule reports a violation when the mentioned construct cannot be processed through the built-in method.

For Example:

```
typedef enum {one =1, two =2} num;
    module top ();
     parameter num [1:0][1:0]a = '{'{one,two},'{one,two}};
     wire b = a[1].num(); //Error case since a[1] is an
array and built-in method is not allowed on this.
    endmodule
```

Message Details

[SYNTAX] Incorrect reference of built-in method '<method>' for arrayed enum

Severity

Syntax

STX_VE_INACTIVE_493

Mentioned construct cannot be processed through built-in method

Language

Verilog

Rule Description

The *STX_VE_INACTIVE_493* rule reports a violation when the mentioned construct cannot be processed through the built-in method.

For Example:

```
typedef enum {one =1, two =2} num;
    module top ();
     parameter num [1:0][1:0]a = '{'{one,two},'{one,two}};
     wire b = a[1].num(); //Error case since a[1] is an array and built-in method is not allowed on this.
    endmodule
```

NOTE: *The STX_VE_INACTIVE_493 rule is flagged when the set_option enable_inactive_rtl_checks yes command is used in the project file.*

Message Details

[SYNTAX] Incorrect reference of built-in method '<method>' for arrayed enums

Severity

Syntax

STX_VE_494

Number of connections to the mentioned built-in method call is incorrect

Language

Verilog

Rule Description

The STX_VE_494 rule reports a violation when the number of connections to the mentioned built-in method call is incorrect.

The built-in method calls to `.next()` and `.prev()` allow single connection to it. All remaining built-in methods, `.num()`, `.name()`, `.last()` and `.first()`, should not have any connections.

For Example:

```
typedef enum {one =1, two =2} num;
    module top ();
     parameter num a = one;
     wire b = a.num(1); //Error case since num() does not accept any connections.
    endmodule
```

Message Details

[SYNTAX] Invalid number of arguments to built-in method call '<method>'

Severity

Syntax

STX_VE_INACTIVE_494

Number of connections to the mentioned built-in method call is incorrect

Language

Verilog

Rule Description

The STX_VE_INACTIVE_494 rule reports a violation when the number of connections to the mentioned built-in method call is incorrect.

The built-in method calls to `.next()` and `.prev()` allow single connection to it. All remaining built-in methods, `.num()`, `.name()`, `.last()` and `.first()`, should not have any connections.

For Example:

```
typedef enum {one =1, two =2} num;
    module top ();
     parameter num a = one;
     wire b = a.num(1); //Error case since num() does not accept any connections.
    endmodule
```

NOTE: *The STX_VE_INACTIVE_494 rule is flagged when the set_option enable_inactive_rtl_checks yes command is used in the project file.*

Message Details

[SYNTAX] Invalid number of arguments to built-in method call '<method>'

Severity

Syntax

STX_VE_495

Each `ifdef and `ifndef compile directive must match an `endif directive before the end of file is encountered.

Language

Verilog

Rule Description

This violation appears if there is no matching `endif directive for the `ifdef and `ifndef directive.

However, this violation might also appear if the HDL text containing directives is skipped completely due to the following reasons:

- HDL text is inside the translate_off/translate_on region.
- HDL text is inside the boundary of the ignored design unit (set_option ignoredu *<du-name>*).

For example, this rule reports a violation in the following case:

```
`ifdef POWER_PIN
    module libcell
    `else
    module libcell
    `endif   // This will not be read resulting in
               // reporting of violation
    ...
    endmodule
```

How to Fix this Violation if a Skipping HDL text Occurs

To fix this violation, correct the HDL so that no directive is present inside the above-mentioned regions.

Message Details

```
End of source encountered before closing all `ifdef/`ifndef in
region starting in file ( <file-name> ) at line ( <line-num> )
```

Severity

Syntax

STX_VE_496

End of file should not be encountered in between paired compiler directives, e.g., an `ifdef must match an `endif before end of file

Language

Verilog

Rule Description

End of file should not be encountered in between paired compiler directives, e.g., an `ifdef must match an `endif before end of file

Message Details

```
End of file detected inside <compiler-directive> directive.
```

Severity

Syntax

STX_VE_498

Named connections to built-in methods are not allowed

Language

Verilog

Rule Description

The *STX_VE_498* rule reports a violation when named connections to any built-in method are found.

For Example:

```
 typedef enum {one =1, two =2} num;
    module top ();
     parameter num a = one;
     parameter num b = a.next(.N(1)); //Error case since
no named connections are allowed.
    endmodule
```

Message Details

[SYNTAX] Named connections not allowed for built-in method '<method>'

Severity

Syntax

STX_VE_INACTIVE_498

Named connections to built-in methods are not allowed

Language

Verilog

Rule Description

The *STX_VE_INACTIVE_498* rule reports a violation when named connections to any built-in method are found.

For Example:

```
typedef enum {one =1, two =2} num;
   module top ();
    parameter num a = one;
    parameter num b = a.next(.N(1)); //Error case since
no named connections are allowed.
   endmodule
```

NOTE: *The STX_VE_INACTIVE_498 rule is flagged when the set_option enable_inactive_rtl_checks yes command is used in the project file.*

Message Details

[SYNTAX] Named connections not allowed for built-in method '<method>'

Severity

Syntax

STX_VE_499

Argument to mentioned built-in method (next or previous) should be integer compatible

Language

Verilog

Rule Description

The *STX_VE_499* rule reports a violation when argument to mentioned built-in method (next or previous) is not integer compatible.

For Example:

```
 typedef enum {one =1, two =2} num;
    typedef struct {int b ;} myVar;|

    module top ();
    parameter num a = two;
    myVar c;
    wire b;
     always @*
      begin
       assign b = a.next(c);  // Error case since 'c' is
not integer compatible.
      end
    endmodule
```

Message Details

[SYNTAX] Incompatible argument to built-in method '<method>'

Severity

Syntax

STX_VE_INACTIVE_499

Argument to mentioned built-in method (next or previous) should be integer compatible

Language

Verilog

Rule Description

The *STX_VE_INACTIVE_499* rule reports a violation when argument to mentioned built-in method (next or previous) is not integer compatible.

For Example:

```
 typedef enum {one =1, two =2} num;
    typedef struct {int b ;} myVar;|

    module top ();
    parameter num a = two;
    myVar c;
    wire b;
     always @*
      begin
       assign b = a.next(c);  // Error case since 'c' is
not integer compatible.
      end
    endmodule
```

NOTE: *The STX_VE_INACTIVE_499 rule is flagged when the set_option enable_inactive_rtl_checks yes command is used in the project file.*

Message Details

[SYNTAX] Incompatible argument to built-in method '<method>'

Severity

Syntax

STX_VE_502

An `else/`elsif/`endif directive must match an `ifdef/`ifndef directive before the end of file.

Language

Verilog

Message Details

```
Mismatching `else or `elsif or `endif compiler directive.
```

Severity

Syntax

STX_VE_503

Count of compiler directives `begin_keywords and `end_keywords must match before the end of a design.

Language

Verilog

Rule Description

SpyGlass reports this syntax error to indicate a mismatch between the number of `begin_keywords and `end_keywords directives before the end of a design.

The following examples explain the violating and non-violating cases for this syntax error.

Example 1

In the following example, the `begin_keywords directive is not accompanied by the `end_keywords directive. Therefore, the *STX_VE_503* rule reports a violation in this case.

```
// Source file test1.v
`begin_keywords "1364-1995"
module test
wire w;
  ...
endmodule

// End of design
```

Example 2

In the following example, the number of the `begin_keywords directive in the test1.v file matches with the number of the `end_keywords directive in the test2.v file. Therefore, the *STX_VE_503* rule does not report any violation in this case.

```
// Source file test1.v
```

```
`begin_keywords "1364-1995"
module test
wire w;
  ...
endmodule

// Source file test2.v
module test2
wire z;
  ...
endmodule
`end_keywords

// End of Design
```

Message Details

```
Missing compiler directive ( <`begin_keywords | `end_keywords>
) for ( <`end_keywords | `begin_keywords> )
```

Severity

Syntax

STX_VE_504

Compiler Directives, `begin_keywords and `end_keywords are allowed in SystemVerilog.

Language

Verilog

Rule Description

SpyGlass reports this syntax error if you specify the `begin_keywords and `end_keywords compiler directive in a language mode other than SystemVerilog.

For example, if you specify these directives in the Verilog mode, the *STX_VE_504* rule reports a violation.

Message Details

```
Compiler Directive ( <directive> ) is only supported for
SystemVerilog
```

Severity

Syntax

STX_VE_505

Compiler Directives, `begin_keywords and `end_keywords are allowed only outside a design element.

Language

Verilog

Rule Description

SpyGlass reports this syntax error if you specify the `begin_keywords and `end_keywords compiler directives within the scope of a design element.

The following examples explain the violating and non-violating cases for this syntax error.

Example 1

In the following example, the *STX_VE_505* rule reports a violation because the `begin_keywords directive is specified within the scope of the design element.

```
// Source file test1.v
module test
wire w;
`begin_keywords "1364-1995"
  ...
endmodule
`end_keywords

// End of Design
```

Example 2

In the following example, the *STX_VE_505* rule does not report a violation because the `begin_keywords directive is specified outside the scope of the design element.

```
// Source file test2.v
`begin_keywords "1364-1995"
```

```
module test2
wire z;
  ...
endmodule
`end_keywords

// End of Design
```

Message Details

```
Compiler Directive ( <directive> ) can only be specified
outside a design element
```

Severity

Syntax

STX_VE_506

Version specified with compiler directive `begin_keywords is invalid.

Language

Verilog

Rule Description

SpyGlass reports this syntax error if you specify a version other than the following versions with the `begin_keywords compiler directive:

1364-1995	1364-2001-noconfig	1364-2001	1364-2005
1800-2005	1800-2009		

The following examples explain the violating and non-violating cases for this syntax error.

Example 1

In the following example, the *STX_VE_506* rule reports a violation because a non-supported version is specified with the `begin_keywords compiler directive.

```
// Source file test1.v
`begin_keywords "1364-2009"
module test
wire w;
  ...
endmodule
`end_keywords

// End of Design
```

Example 2

In the following example, the *STX_VE_506* rule does not report a violation because a supported version is specified with the `begin_keywords

compiler directive.

```
// Source file test2.v
`begin_keywords "1364-1995"
module test2
wire z;
  ...
endmodule
`end_keywords

// End of Design
```

Message Details

```
Incorrect version ( <version> ) specified with `begin_keywords
compiler directive
```

Severity

Syntax

STX_VE_509

Compiler directives, such as 'accelerate', 'expand', 'noexpand', 'remove', and so on are not allowed inside the module definition boundary.

Language

Verilog

Message Details

```
Compiler directive ( <compiler-directive> ) not allowed inside
module definition boundary.
```

Severity

Syntax

STX_VE_511

Strength types other than pull0 and pull1 for `unconnected_drive directive are invalid.

Language

Verilog

Message Details

```
Invalid strength type for `unconnected_drive directive.
```

Severity

Syntax

STX_VE_520

Un-terminated quoted strings are not allowed.

Language

Verilog

Message Details

```
Un-terminated quoted string
```

Severity

Syntax

STX_VE_521

Un-terminated comments are not allowed.

Language

Verilog

Message Details

```
Un-terminated comment.
```

Severity

Syntax

STX_VE_522

The dc_script_begin pragma should have corresponding dc_script_end in the same file

Description

The *STX_VE_522* rule reports a violation when the `dc_script_begin` pragma does not have a corresponding `dc_script_end` in the same file.

For example:

```
always @(sel or inp)
begin
//synopsys dc_script_begin
case(sel)
  2'b00 : outp = inp[0];
  2'b01 : outp = inp[1];
  2'b10 : outp = inp[2];
  2'b11 : outp = inp[3];
endcase
end
```

Language

Verilog

Message Details

```
End of file reached before dc_script_end
```

Severity

Syntax

STX_VE_527

A macro definition must be syntactically correct for macro expansion.

Language

Verilog

Message Details

```
Syntax error in the definition of macro ( <macro> ).
```

Severity

Syntax

STX_VE_528

Macro usage must be syntactically correct for macro expansion.

Language

Verilog

Message Details

```
Syntax error : cannot expand macro ( <macro> ).
```

Severity

Syntax

STX_VE_533

A macro must be defined before it is used.

Language

Verilog

Message Details

```
Used macro (<macro>) has not been defined. Trace: (<macro
trace>)
```

Severity

Syntax

STX_VE_536

Macros must be used with the proper number of arguments as per the macro definition.

Language

Verilog

Rule Description

The number of arguments specified in a macro use must be equal to the number of arguments specified in its definition.

Message Details

```
Macro ( <macro> ) defined in file ( <file> ), at line (
<line-num> ), requires ( <num1> ) arguments, whereas ( <num2> )
arguments are specified here
```

Severity

Syntax

STX_VE_537

Recursions in macro must be avoided.

Language

Verilog

Message Details

`Recursion found in macro ( <macro> ).`

Severity

Syntax

STX_VE_541

The edge symbol must have only two-level symbols concatenated.

Language

Verilog

Message Details

```
Edge symbol must have only two level symbols concatenated.
```

Severity

Syntax

STX_VE_551

The name of a module must be specified.

Language

Verilog

Message Details

Name of module missing.

Severity

Syntax

STX_VE_552

The name of a UDP must be specified.

Language

Verilog

Message Details

`Name of UDP missing.`

Severity

Syntax

STX_VE_554

The name of a function must be specified.

Language

Verilog

Message Details

```
Name of function missing.
```

Severity

Syntax

STX_VE_561

The keyword 'endmodule ' must be specified to end a module.

Language

Verilog

Message Details

```
Keyword 'endmodule' missing for module <module>
```

Severity

Syntax

STX_VE_562

The keyword 'endprimitive ' must be specified to end a primitive.

Language

Verilog

Message Details

`Keyword 'endprimitive' missing.`

Severity

Syntax

STX_VE_563

The keyword 'endattribute' must be specified to end an attribute.

Language

Verilog

Message Details

```
Keyword 'endattribute' missing.
```

Severity

Syntax

STX_VE_564

The keyword 'endfunction' must be specified to end a function.

Language

Verilog

Message Details

```
Keyword 'endfunction' missing for function '<function-name>'
defined in file '<file-name>' at line '<line-num>'
```

Severity

Syntax

STX_VE_565

The keyword 'endtask' must be specified to end a task.

Language

Verilog

Message Details

```
Keyword 'endtask' missing.
```

Severity

Syntax

STX_VE_566

The keyword 'endtable' must be specified to end a table.

Language

Verilog

Message Details

```
Keyword 'endtable' missing
```

Severity

Syntax

STX_VE_569

The keyword 'end' must be specified to end a block.

Language

Verilog

Message Details

`Keyword 'end' missing.`

Severity

Syntax

STX_VE_570

The keyword 'join' must be specified to end a fork-join block.

Language

Verilog

Message Details

```
Keyword 'join' missing
```

Severity

Syntax

STX_VE_573

Semicolon should be present.

Language

Verilog

Message Details

`semicolon missing`

Severity

Syntax

STX_VE_576

***)' is missing.**

Language

Verilog

Message Details

```
'*)' missing
```

Severity

Syntax

STX_VE_585

The instance-specific definition must be found in the `uselib path.

Language

Verilog

Message Details

```
Instance specific definition not found in `uselib path
```

Severity

Syntax

STX_VE_589

The re-declaration of a module with the same name is not allowed.

Language

Verilog

Rule Description

The re-declaration of a module with the same name is not allowed.

Use the `set_option allow_module_override yes` command in the project file to enable SpyGlass to read multiple module definitions with the same name. Then, the last analyzed module definition overrides the previous design unit definitions, which can be either modules or UDPs. This message is flagged if either a module or a UDP definition is given before the current module of the same name.

Message Details

```
Module name ( <module-name> ) previously declared in file (
<file> ), at line no <line-num>. Specify 'set_option
allow_module_override yes' to support multiple module
definitions with the same name
```

Severity

Syntax

STX_VE_590

The re-declaration of a UDP with the same name is not allowed.

Language

Verilog

Rule Description

The re-declaration of a UDP with the same name is not allowed. Use the `set_option allow_module_override yes` command in the project file for SpyGlass to read multiple UDP definitions with the same name. Then, the last analyzed UDP definition overrides the previous UDP definition(s). This message is flagged if either a module or a UDP definition is given before the current UDP of the same name.

Message Details

```
UDP name ( <name> ) previously declared in file ( <file> ), at
line no <line-num>. Specify 'set_option allow_module_override
yes' to support multiple udp definitions with the same name"
```

Severity

Syntax

STX_VE_591

The re-declaration of a gate with the same name is not allowed.

Language

Verilog

Message Details

```
Gate (<gate>) previously declared in file (<file>), at line
(<line-num>)
```

Severity

Syntax

STX_VE_597

Module/UDP Instance cannot be used in a port connection.

Language

Verilog

Rule Description

In a port connection, you cannot use a module or UDP instance. For example, SpyGlass flags the STX_VE_597 message in the following case:

```
module mid;
endmodule

module top;
  mid m1();
  low l1 (m1);
endmodule
```

Message Details

```
Module/UDP Instance ( <instance> ) cannot be used in a port
connection
```

Severity

Syntax

STX_VE_INACTIVE_597

Module / UDP Instance cannot be used in a port connection.

Language

Verilog

Rule Description

In a port connection, you cannot use a module or UDP instance. For example, SpyGlass flags the STX_VE_597 message in the following case:

```
module mid;
endmodule

module top;
  mid m1();
  low l1 (m1);
endmodule
```

NOTE: *The STX_VE_INACTIVE_597 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Module/UDP Instance ( <instance> ) cannot be used in a port connection`

Severity

Syntax

STX_VE_598

The declaration of a module with an existing name is not allowed.

Language

Verilog

Message Details

```
Module name ( <name> ) previously declared in file (<file>), at
line (<line-num>)
```

Severity

Syntax

STX_VE_599

The re-declaration of a Net with the same name is not allowed.

Language

Verilog

Message Details

```
Net name ( <name> ) previously declared in file ( <file> ) at
line ( <line-num> )
```

Severity

Syntax

STX_VE_600

The re-declaration of a variable with the same name is not allowed. Re-declaration error can be flagged due to reparsing of global variables in library files.

Language

Verilog

Description

This rule reports a violation for redeclaration of a variable with the same name.

Redeclaration errors can be reported due to reparsing of global variables in library files.

Message Details

```
Variable name ( <name> ) previously declared in file (<file>),
at line (<line-num>)
```

Severity

Syntax

STX_VE_601

Net type must be explicitly declared

Language

Verilog

Rule Description

The STX_VE_601 rule flags implicit net declarations. Such type of declaration is not allowed as `'default_nettype` is declared as none.

Message Details

```
Net type of ( <type> ) must be explicitly declared, since
`default_nettype is NONE
```

Severity

Syntax

STX_VE_602

The redefinition of an identifier with the same name is not allowed. Redefinition error can be flagged due to reparsing of global variables in library files.

Language

Verilog

Description

This rule reports a violation for redefinition of an identifier with the same name.

Redefinition error can be reported due to reparsing of global variables in library files.

Message Details

```
Redefinition of name ( <name> ). Previously defined in file (
<file> ), at line ( <line-num> )
```

Severity

Syntax

STX_VE_603

Invalid user data type is specified.

Language

Verilog

Message Details

`<data-type> is not a user-defined data type`

Severity

Syntax

STX_VE_604

Re-declaration of extern declaration of a module/interface in the same file

Language

Verilog

Message Details

```
Re-declaration of extern declaration of <module | interface> (
<name> ) in the file ( <file> ) at line ( <line-num> )
```

Severity

Syntax

STX_VE_605

Invalid use of specified identifier.

Language

Verilog

Message Details

```
Invalid use of identifier ( <identifier> )
```

Severity

Syntax

STX_VE_606

Only those identifiers can be used that are declared in the current scope.

Language

Verilog

Message Details

```
Identifier ( <identifier> ) not declared in current scope
```

Severity

Syntax

STX_VE_607

Identifiers of size greater than 10240 are not allowed.

Language

Verilog

Message Details

```
Identifiers of size greater than 10240 are not allowed
```

Severity

Syntax

STX_VE_608

The text specified with the compiler directive is not a valid identifier.

Language

Verilog

Message Details

```
( <identifier> ) is not a valid identifier
```

Severity

Syntax

STX_VE_609

Specified expression is illegal to be used.

Language

Verilog

Rule Description

SpyGlass reports this syntax error to indicate that the specified expression is illegal.

An expression is considered as legal in the following cases:

- If the condition expression is a singular expression inside a case construct

  Consider the following example:

```
int a;
struct { int s;} str;
case (a) inside // No violation
...
endcase

case (str) inside //Violation as str is not singular
...
endcase
```

- If expressions defining a range are singular in a value range construct

  Consider the following example:

```
int a,b,c;
int d [4];
case (a) inside
  [a : b ] : begin  // No violation
    ...
end

[c : d ] : begin  // Violation as d is not singular
    ...
end
endcase
```

- If only single expression is used in a value range construct, then that expression is either singular or an array of singular type

  Consider the following example:

```
int a,b;
int d[4];
struct { int s;} str;
case (a) inside
 b  : begin // No Violation
  ...
end
 d  : begin // No Violation
  ...
end
  str : begin // Violation as str is not singular
  ...
end
endcase
```

Message Details

`Illegal expression ( <expression-string> ), <message>`

Where, *<message>* can be any of the following text:

- `Expression should be a singular type`
- `Expression should either be a singular type or an array of singular type`

Severity

Syntax

STX_VE_INACTIVE_609

Specified expression is illegal to be used.

Language

Verilog

Rule Description

SpyGlass reports this syntax error to indicate that the specified expression is illegal.

An expression is considered as legal in the following cases:

- If the condition expression is a singular expression inside a case construct

  Consider the following example:

```
int a;
struct { int s;} str;
case (a) inside //No violation
...
endcase

case (str) inside //Violation as str is not singular
...
endcase
```

- If expressions defining a range are singular in a value range construct

  Consider the following example:

```
int a,b,c;
int d [4];
case (a) inside
  [a : b ] : begin  // No violation
    ...
end

[c : d ] : begin  // Violation as d is not singular
    ...
end
endcase
```

- If only single expression is used in a value range construct, then that expression is either singular or an array of singular type

  Consider the following example:

```
int a,b;
int d[4];
struct  { int s;} str;
case (a) inside
 b  : begin // No Violation
  ...
end
 d  : begin // No Violation
  ...
end
  str : begin // Violation as str is not singular
  ...
end
endcase
```

NOTE: *The STX_VE_INACTIVE_609 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Illegal expression ( <expression-string> ), <message>`

Severity

Syntax

STX_VE_610

Specified expression is illegally used in the mentioned context

Language

Verilog

Description

The STX_VE_610 rule reports a violation if specified expression is illegally used in the mentioned context.

Messages and Suggested Fix

This rule reports the following message:

[SYNTAX] `Illegal use of <expression-type> (<invalid-expression>) in <context-in-which-invalid-exp-is-used> context`

For example, consider the following RTL expression containing an invalid expression:

```
assign a[1] = b[mem[i +: 1]];
```

For the above RTL, the following *STX_VE_610* violation appears:

```
Illegal use of unpacked construct (mem[i +:1] ) in select index context
```

Consequence of Not Fixing

SpyGlass run does not proceed further.

How to Debug and Fix

Remove the reported invalid expression from the code.

Example Code and/or Schematic

Example 1

Consider the following example:

```
module top;
```

```
  wire b;
  int a[2] = '{0:1,1:1};
  real i;
  assign b = a[i]; // Index i is of the type real.
 endmodule
```

For the above example, the *STX_VE_610* rule reports a violation because the select index is of the type real.

Example 2

Consider the following example:

```
 module top;
  wire b;
  real i;
  int a[2] = '{0:1,1:1};
  assign b = a[i:0]; // Error since bound i is of type
real.
endmodule
```

For the above example, the *STX_VE_610* rule reports a violation because the array bound is of the type real.

Example 3

Consider the following example:

```
 enum {red =1}a;
  module top;
  always
  begin
  for (a.num = 0; s<1 ; s++ ); // Error since built-in
method cannot be used as LHS of assignment
  begin
  end
  end
  endmodule
```

For the above example, the *STX_VE_610* rule reports a violation because the expression is a built-in method on the LHS of the assignment.

Default Severity Label

Syntax

STX_VE_INACTIVE_610

Specified expression is illegally used in the mentioned context

Language

Verilog

Description

The STX_VE_INACTIVE_610 rule reports a violation if specified expression is illegally used in the mentioned context.

NOTE: *The STX_VE_INACTIVE_610 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Messages and Suggested Fix

This rule reports the following message:

[SYNTAX] `Illegal use of <expression-type> (<invalid-expression>) in <context-in-which-invalid-exp-is-used> context`

For example, consider the following RTL expression containing an invalid expression:

```
assign a[1] = b[mem[i +: 1]];
```

For the above RTL, the following *STX_VE_610* violation appears:

```
Illegal use of unpacked construct (mem[i +:1] ) in select index context
```

Consequence of Not Fixing

SpyGlass run does not proceed further.

How to Debug and Fix

Remove the reported invalid expression from the code.

Example Code and/or Schematic

Example 1

Consider the following example:

```
 module top;
  wire b;
  int a[2] = '{0:1,1:1};
  real i;
  assign b = a[i]; // Index i is of the type real.
 endmodule
```

For the above example, the *STX_VE_INACTIVE_610* rule reports a violation because the select index is of the type real.

Example 2

Consider the following example:

```
 module top;
  wire b;
  real i;
  int a[2] = '{0:1,1:1};
  assign b = a[i:0]; // Error since bound i is of type real.
endmodule
```

For the above example, the *STX_VE_INACTIVE_610* rule reports a violation because the array bound is of the type real.

Example 3

Consider the following example:

```
 enum {red =1}a;
  module top;
  always
  begin
  for (a.num = 0; s<1 ; s++ ); // Error since built-in
method cannot be used as LHS of assignment
  begin
  end
  end
```

```
endmodule
```

For the above example, the *STX_VE_INACTIVE_610* rule reports a violation because the expression is a built-in method on the LHS of the assignment.

Default Severity Label

Syntax

STX_VE_627

Call to function/task (sequence/property) must be done with the proper number of arguments as per the function/task (sequence/property) definition.

Language

Verilog

Rule Description

SystemVerilog supports function/task (sequence/property) call having proper number of arguments as per the function/task (sequence/property) definition. For example:

```
property p(a,b);
    a or b;
endproperty
property p1;
    p(a);
endproperty
```

Here in property p1, p is called with 1 argument which is illegal.

Message Details

```
call to <function | task> ( <name> ) defined at file ( <file> ),
line ( <line-num> ) has too <many | few> arguments ( ( <num> )
<more | less> )
```

Severity

Syntax

STX_VE_629

Calls to task/function must be made with the proper number of arguments.

Language

Verilog

Message Details

```
Too few arguments in call to <task | function> ( <name> )
```

Severity

Syntax

STX_VE_643

A port must be defined as either input, output, or inout.

Language

Verilog

Message Details

`Port ( <port> ) not defined as input, output or inout`

Severity

Syntax

STX_VE_647

An identifier cannot be declared as input if it is not present in the module header.

Language

Verilog

Message Details

```
( <identifier> ) is declared as input though not in module
header
```

Severity

Syntax

STX_VE_648

An identifier cannot be declared as output if it is not present in the module header.

Language

Verilog

Message Details

```
( <identifier> ) is declared as output though not in module
header
```

Severity

Syntax

STX_VE_649

An identifier cannot be declared as inout if it is not present in the module header.

Language

Verilog

Message Details

```
( <identifier> ) is declared as inout though not in module
header
```

Severity

Syntax

STX_VE_650

An identifier cannot be declared as input if it is not present in the interface header.

Language

Verilog

Message Details

```
( <identifier> ) is declared as input though not in <interface>
header
```

Severity

Syntax

STX_VE_651

An identifier cannot be declared as output if it is not present in the interface header.

Language

Verilog

Message Details

```
( <identifier> ) is declared as inout though not in <interface>
header
```

Severity

Syntax

STX_VE_652

An identifier cannot be declared as inout if it is not present in the interface header.

Language

Verilog

Message Details

```
( <identifier> ) is declared as inout though not in <interface>
header
```

Severity

Syntax

STX_VE_667

Unary and binary type expressions are not allowed on output ports.

Language

Verilog

Message Details

```
Expression ( <expression> ) on output port in illegal
```

Severity

Syntax

STX_VE_668

Non-static bit-select/part-select expression is not allowed for module output port connection.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of non-static bit-select/part-select expression for module output port connection.

For example,

```
module top;
   wire [7:0] a;
   wire [1:0] b;
   wire c;
   chile inst1(a[b], c);
endmodule

module child(y,in1);
   output y;
   input in1;
   ...
endmodule
```

In the above example, *a[b]* is an invalid non-static bit-selection for output port *(y)*.

Message Details

```
Non-static bit-select/part-select expression ( <expression> )
not allowed for module output port ( <port> ) connection
```

Severity

Syntax

STX_VE_INACTIVE_668

Non-static bit-select/part-select expression is not allowed for module output port connection.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of non-static bit-select/part-select expression for module output port connection.

For example,

```
module top;
   wire [7:0] a;
   wire [1:0] b;
   wire c;
   chile inst1(a[b], c);
endmodule

module child(y,in1);
   output y;
   input in1;
   ...
endmodule
```

In the above example, *a[b]* is an invalid non-static bit-selection for output port *(y)*.

NOTE: *The STX_VE_INACTIVE_668 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Non-static bit-select/part-select expression ( <expression> )
not allowed for module output port ( <port> ) connection
```

Severity

Syntax

STX_VE_672

Registers are not allowed in input or inout port.

Language

Verilog

Message Details

```
Register not allowed in input or inout port
```

Severity

Syntax

STX_VE_674

Re-declaration of a port is not allowed.

Language

Verilog

Rule Description

Two ports of the same name cannot be declared.

For example, the following code snippet would report STX_VE_674:

```
module (a,a);
    input a;
    output a;
end module
```

Message Details

```
Port name ( <name> ) previously declared in file (<file>), at
line (<line-num>)
```

Severity

Syntax

STX_VE_676

Direction cannot be specified for an object, which is neither a net nor a port net.

Language

Verilog

Message Details

```
Direction cannot be specified for this object ( <object> )
```

Severity

Syntax

STX_VE_681

Port connections must be given for a UDP.

Language

Verilog

Message Details

`Port connections are not given for ( <name> ) UDP`

Severity

Syntax

STX_VE_689

The specified port occurs more than once in the specified instance.

Language

Verilog

Rule Description

In the named association port list, the port has been used more than once.

Message Details

```
Multiple connections to port ( <port> ) in instance (
<instance> )
```

Severity

Syntax

STX_VE_690

A module or UDP instance must have the same number of connection as defined in the parent module or UDP.

Language

Verilog

Rule Description

A module or UDP instance must have the same number of connections as defined in the parent module or UDP. To get the list of all extra connections, view the text container.

Message Details

Normal SpyGlass Run:

```
Instance (<inst-name>) of <module | udp> (<name>) defined at
file (<file-name>), at line (<line-num>) has (<num>) extra
connections (<extra-connections-list>).
```

SpyGlass Library Compiler Run:

```
Ports of cell <cell-name> <module | udp> (<name>) defined at
file (<file-name>), line (<line-num>) has <num> less
connections (<connections-list>)
```

Severity

Syntax

STX_VE_INACTIVE_690

A module or UDP instance must have the same number of connection as defined in the parent module or UDP.

Language

Verilog

Rule Description

A module or UDP instance must have the same number of connections as defined in the parent module or UDP. To get the list of all extra connections, view the text container.

NOTE: *The STX_VE_INACTIVE_690 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

Normal SpyGlass Run:

```
Instance (<inst-name>) of <module | udp> (<name>) defined at
file (<file-name>), at line (<line-num>) has (<num>) extra
connections (<extra-connections-list>).
```

SpyGlass Library Compiler Run:

```
Ports of cell <cell-name> <module | udp> (<name>) defined at
file (<file-name>), line (<line-num>) has <num> less
connections (<connections-list>)
```

Severity

Syntax

STX_VE_699

For a vectored net, bit-select or part-select on that net in the output of any instance is not allowed.

Language

Verilog

Message Details

```
Conflict in vectored net expansion for <net>
```

Severity

Syntax

STX_VE_711

Only reg type net declarations are allowed inside task/function declaration.

Language

Verilog

Message Details

```
Only reg type net declarations are allowed inside task/function
declaration
```

Severity

Syntax

STX_VE_712

Only reg type net declarations are allowed inside block statements.

Language

Verilog

Message Details

```
Only reg type net declarations are allowed inside block
statements
```

Severity

Syntax

STX_VE_714

Event declaration inside a scope other than a module is not supported.

Language

Verilog

Message Details

```
Event declaration inside a scope other than a module is not
supported
```

Severity

Syntax

STX_VE_718

Re-declaration of a port with the same name in a UDP is not allowed.

Language

Verilog

Message Details

```
In UDP the port ( <port> ) is already declared
```

Severity

Syntax

STX_VE_719

Vectored terminals are not allowed in primitive definitions.

Language

Verilog

Message Details

```
Vectored terminals are not allowed in primitive definitions
```

Severity

Syntax

STX_VE_722

In a UDP table, the number of columns must match with the number of ports.

Language

Verilog

Message Details

```
In UDP table, number of columns does not match with number of
ports
```

Severity

Syntax

STX_VE_725

The first terminal must be declared as output in the UDP header.

Language

Verilog

Message Details

```
The first terminal ( <terminal> ) must be declared as output in
UDP header
```

Severity

Syntax

STX_VE_726

In a UDP, only the first port in the port list can be declared as output.

Language

Verilog

Message Details

In UDP only the first port (<port1>) in the port list can be declared as output. But (<port2>) is declared as output

Severity

Syntax

STX_VE_727

Only one output terminal is allowed in a UDP definition.

Language

Verilog

Message Details

```
Only one output terminal is allowed in a primitive definition
```

Severity

Syntax

STX_VE_731

An identifier cannot be declared as output if it is not specified in the UDP header.

Language

Verilog

Message Details

```
( <identifier> ) is declared as output though not in UDP header
```

Severity

Syntax

STX_VE_732

An identifier cannot be declared as input if it is not specified in the UDP header.

Language

Verilog

Message Details

```
( <identifier> ) is declared as input though not in UDP header
```

Severity

Syntax

STX_VE_735

A port cannot be declared as reg type if it is not specified as an output port.

Language

Verilog

Message Details

```
( <port> ) is declared as reg type though it is not an output
port
```

Severity

Syntax

STX_VE_736

A net cannot be declared as reg type if it is not specified in the UDP header.

Language

Verilog

Message Details

```
( <net> ) is declared as reg type though not in UDP header
```

Severity

Syntax

STX_VE_741

A connection must be specified for a UDP output port.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a connection to the output port that is open or is null.

Message Details

`No connection to UDP output port ( <port> )`

Severity

Syntax

STX_VE_INACTIVE_741

A connection must be specified for a UDP output port.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of a connection to the output port that is open or is null.

NOTE: *The STX_VE_INACTIVE_741 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
No connection to UDP output port ( <port> )
```

Severity

Syntax

STX_VE_743

Named port association is not allowed in UDP instances.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of named port association in the UDP instances. Only positional port association is allowed.

Message Details

```
Instance ( <instance> ) defined incorrectly. Named port
association is not allowed in UDP instances
```

Severity

Syntax

STX_VE_INACTIVE_743

Named port association is not allowed in UDP instances.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate presence of named port association in the UDP instances. Only positional port association is allowed.

NOTE: *The STX_VE_INACTIVE_743 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Instance ( <instance> ) defined incorrectly. Named port
association is not allowed in UDP instances
```

Severity

Syntax

STX_VE_748

In a UDP, the initial statement can only be attached to a net that is already declared.

Language

Verilog

Message Details

```
In the UDP, initial statement attached to a net ( <net> ) that
is not already declared
```

Severity

Syntax

STX_VE_749

In a UDP, the initial statement can only be attached to a net that is already declared as output.

Language

Verilog

Message Details

```
In UDP, initial statement attached to a net ( <net> ) which is
not declared as output
```

Severity

Syntax

STX_VE_754

Present state can have only one level symbol.

Language

Verilog

Message Details

```
Present state can have only one level symbol
```

Severity

Syntax

STX_VE_755

In sequential UDP, one table entry cannot have more than one edge.

Language

Verilog

Message Details

```
In sequential UDP, one table entry cannot have more than one
edge
```

Severity

Syntax

STX_VE_756

A UDP table that has entries with the same input but different output values is not allowed.

Language

Verilog

Message Details

```
UDP table entry conflicts with entry defined at line <line-num>
```

Severity

Syntax

STX_VE_757

Specifying two symbols that are same for edge specification in a primitive table is not allowed.

Language

Verilog

Message Details

```
Primitive table has invalid edge specifications: (
<specifications> )
```

Severity

Syntax

STX_VE_760

Combinational UDP cannot have Initial Statement.

Language

Verilog

Message Details

```
Combinational UDP cannot have initial statement
```

Severity

Syntax

STX_VE_762

Combinational UDPs cannot have an output declared as REG.

Language

Verilog

Message Details

Combinational UDP cannot have output declared as REG. But port (<port>) is declared as REG

Severity

Syntax

STX_VE_767

Sequential UDPs must have an output declared as REG.

Language

Verilog

Message Details

```
Sequential UDP must have output declared as REG. But port (
<port> ) is not declared as REG
```

Severity

Syntax

STX_VE_774

Extern task/function and modport declarations can come inside generate blocks only within an interface scope.

Language

Verilog

Rule Description

Extern task/function and modport declarations can come inside generate blocks only within an interface scope. For example:

```
interface intf;
  generate
  if (c == 1)
   m1(input a);
  endgenerate
endinterface
```

Message Details

```
<task | function> inside generate blocks can only come inside
interface scope
```

Severity

Syntax

STX_VE_775

Initial statement is not allowed in a scope other than a module.

Language

Verilog

Message Details

```
Initial statement not allowed in this scope
```

Severity

Syntax

STX_VE_776

Always statement is not allowed in a scope other than a module.

Language

Verilog

Message Details

```
Always statement not allowed in this scope
```

Severity

Syntax

STX_VE_781

A variable is valid only within the scope of a module, task, function, and sequential or parallel block.

Language

Verilog

Message Details

```
Invalid scope for variable ( <variable> )
```

Severity

Syntax

STX_VE_782

Invalid scope found for the specified scope variable.

Language

Verilog

Message Details

```
Invalid scope found for scope variable ( <variable> )
```

Severity

Syntax

STX_VE_799

The width of port connection for an array of module/interface instance must be the same or 'X' times the width of port (where X is a positive integer and size of array).

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the width of a port connection for an array is not valid. The width of a port connection for an array of module/interface instance must be the same or 'X' times the width of port (where X is a positive integer and size of array).

For example,

```
module test;
   wire [19:0] a;
   wire [1:0] b;
   child inst1 [3:0] (.y(a),.in1(b));
endmodule

module child(y, in1);
   output [4:0] y;
   input [4:0] in1;
   ...
endmodule
```

In the above example, size of *a* should be either 5-bit (which is equal to size [4:0]y) or 20-bit (which is equal to 5 (that is, size of port y) * 4 (that is, size of instance array inst1)).

Message Details

```
Incorrect width of port connection ( <connection> ) for array
of instance ( <instance> ) of ( <module/interface> ). Must be
same or ( <num> ) times the width ( <width> ) of port ( <port> )
```

Severity

Syntax

STX_VE_INACTIVE_799

The width of port connection for an array of module/interface instance must be the same or 'X' times the width of port (where X is a positive integer and size of array).

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the width of a port connection for an array is not valid. The width of a port connection for an array of module/interface instance must be the same or 'X' times the width of port (where X is a positive integer and size of array).

For example,

```
module test;
   wire [19:0] a;
   wire [1:0] b;
   child inst1 [3:0] (.y(a),.in1(b));
endmodule

module child(y, in1);
   output [4:0] y;
   input [4:0] in1;
   ...
endmodule
```

In the above example, size of *a* should be either 5-bit (which is equal to size [4:0]y) or 20-bit (which is equal to 5 (that is, size of port y) * 4 (that is, size of instance array inst1)).

Message Details

```
Incorrect width of port connection ( <connection> ) for array
of instance ( <instance> ) of ( <module/interface> ). Must be
same or ( <num> ) times the width ( <width> ) of port ( <port> )
```

Severity

Syntax

STX_VE_800

Value of the expression specified in the array bound should be within integer range

Language

Verilog

Rule Description

SpyGlass generates this syntax error if an array bound specified in both bit-select and part-select of an expression exceeds an integer range.

For example, this rule reports a violation in the following case because `p` in the part-select of `q` exceeds the integer range:

```
module top;
  parameter p = 36'hfffffffff;
  wire [p:0] q;
endmodule
```

Message Details

`Array bound exceeds integer range in expression <expr>`

Severity

Syntax

STX_VE_INACTIVE_800

Value of the expression specified in the array bound should be within integer range

Language

Verilog

Rule Description

SpyGlass generates this syntax error if an array bound specified in both bit-select and part-select of an expression exceeds an integer range.

For example, this rule reports a violation in the following case because `p` in the part-select of `q` exceeds the integer range:

```
module top;
  parameter p = 36'hfffffffff;
  wire [p:0] q;
endmodule
```

NOTE: *The STX_VE_INACTIVE_800 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Array bound exceeds integer range in expression <expr>`

Severity

Syntax

STX_VE_807

An output port when redeclared as a reg, then range must be same in both declarations.

Language

Verilog

Message Details

```
Incompatible declaration, ( <port> ) defined as a vector at
line number <line-num>
```

Severity

Syntax

STX_VE_809

Hierarchical reference is not allowed in a constant expression.

Language

Verilog

Message Details

```
Illegal hierarchical reference (<reference>) in a constant
expression
```

Severity

Syntax

STX_VE_810

Non-constant expressions is not permissible where only constant expressions are allowed

Language

Verilog

Message Details

```
Non-constant expression (<expression>) specified where only
constant expressions are allowed
```

Severity

Syntax

STX_VE_811

Object is invalid to be used in the expression.

Language

Verilog

Message Details

```
Object type is invalid to be used in the expression
```

Severity

Syntax

STX_VE_821

Unknown operator type is given for evaluation.

Language

Verilog

Rule Description

NOTE: *The STX_VE_821 rule is an internal rule.*

Message Details

```
Unknown operator type given for evaluation. Please contact
Atrenta Support. Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_823

Unknown operator type is given for expression width calculation.

Language

Verilog

Rule Description

NOTE: *The STX_VE_823 rule is an internal rule.*

Message Details

```
Unknown operator type given for expression width calculation.
Please contact Atrenta Support. Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_841

Specified file could not be opened for reading.

Language

Verilog

Message Details

```
Could not open file "<file>" for reading
```

Severity

Syntax

STX_VE_842

Specified file does not exist.

Language

Verilog

Message Details

```
File "<file>" does not exist
```

Severity

Syntax

STX_VE_850

Premature end of source encountered.

When to Use

This rule runs by default

Language

Verilog

Description

SpyGlass reports this violation if the specified source file is syntactically incomplete or some valid RTL is inadvertently marked as not to be analyzed.

For example, SpyGlass reports a violation in the following cases:

- Mismatch in using the translate_off and translate_on pragmas resulting in some valid RTL getting ignored during SpyGlass analysis.
- Use of `protected - `endprotected directives that results in some valid RTL snippet getting ignored during SpyGlass analysis.
- Presence of unmatched synthesis directive translate_off that is flagged by the WRN_73 rule.
- Design unit not ending in same file where it started, flagged by the *WRN_1463* rule.

Parameter(s)

None

Constraint(s)

None

Messages and Suggested Fix

Message 1

The following message appears if the source design is syntactically incomplete or some valid RTL is inadvertently marked as not to be analyzed.

```
[FATAL] Premature end of source encountered
```

Potential Issues

This violation appears if there are syntactical errors in the source design. The following describes some possible syntactical errors:

- Mismatch in using the translate_off and translate_on pragmas resulting in some valid RTL getting ignored during SpyGlass analysis.
- Use of `protected - `endprotected directives that results in some valid RTL snippet getting ignored during SpyGlass analysis.

Consequences of Not Fixing

SpyGlass cannot proceed with the analysis until the reported errors are resolved.

How to Debug and Fix

Ensure that the specified source file is syntactically complete and any valid RTL is not skipped inadvertently.

Message 2

The following message appears if there is a premature end of source design:

```
[FATAL] Premature end of source encountered, check WRN_73 to
resolve issues related to unmatched 'translate_off' directive,
which may be one of the reasons for premature end of source.
```

Potential Issues

Presence of unmatched synthesis directive translate_off, resulting in some valid RTL code inadvertently getting ignored during SpyGlass analysis.

Consequences of Not Fixing

SpyGlass cannot proceed with the analysis until the reported errors are resolved.

How to Debug and Fix

Resolve all WRN_73 violations to ensure that the specified source file is syntactically complete and any valid RTL is not ignored inadvertently.

Message 3

The following message appears if there is a premature end of source design due to design unit not ending in the same file:

```
[FATAL] Premature end of source, check WRN_1463 to resolve
issue related to missing endmodule which may be one of the
reasons for premature end of source.
```

Potential Issues

Design unit not ending in same file where it started, flagged by the *WRN_1463* rule.

Consequences of Not Fixing

SpyGlass cannot proceed with the analysis until the reported errors are resolved.

How to Debug and Fix

Resolve all WRN_1463 violations to ensure that all design units end within the same file where they started.

Example Code and/or Schematic

Message 1: Premature end of source encountered

```
/// File: test.v
Module test;
Wire w;
Assign w = 1;
//endmodule
```

In the above example, the endmodule keyword is commented and therefore the above file is not syntactically complete.

Message 2: Premature end of source, check WRN_73 to resolve issue related to unmatched 'translate_off' directive which may be one of the reasons for premature end of source.

```
/// File: test.v
Module test;
Wire w;
Assign w = 1;
//synopsys translate_off
endmodule
/// End of file: test.v
```

In the above example, the `translate_off` directive causes the `endmodule` keyword to be ignored. Therefore, the above file is not syntactically complete.

Message 3: Premature end of source, check WRN_1463 to resolve issue related to missing endmodule which may be one of the reasons for premature end of source

```
File: test.v
module A(a,b);  // Violation
      input a,b;
File test1.v
      reg c;
endmodule
```

In the above example, the *WRN_1463* rule reports a violation because the module `A` is begun in file test.v but is not ending in the test.v file.

Default Severity Label

SYNTAX

Related Reports

None

STX_VE_902

veAnalyze() must be called before calling this routine.

Language

Verilog

Severity

INTERNAL_FATAL

STX_VE_904

Out of range expression is not allowed.

Language

Verilog

Message Details

```
Out-of-range error for the expression in line <line-num> in
file <file> in instance <instance>. Please contact Atrenta
Support
```

Severity

INTERNAL_FATAL

STX_VE_905

Out of range expression is not allowed.

Language

Verilog

Message Details

```
Out-of-range error for the expression in line <line-num> in
file <file>. Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_906

The specified message number does not exist.

Language

Verilog

NOTE: *The STX_VE_906 rule is an internal rule.*

Message Details

```
Message <message> does not exist. Please contact Atrenta
Support
```

Severity

INTERNAL_FATAL

STX_VE_907

veInit() must be called before calling this routine.

Language

Verilog

Message Details

```
veInit() must be called before calling this routine. Please
contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_908

The severity of messages with message number greater than 475 cannot be changed.

Language

Verilog

Message Details

```
Severity of message <message> cannot be changed. Please contact
Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_909

A severity not within the range of [1:4] is unknown.

Language

Verilog

Message Details

```
Unknown severity <severity>. Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_910

The messages with message number greater than 475 cannot be suppressed.

Language

Verilog

Message Details

```
The message <message> cannot be suppressed. Please contact
Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_911

Null Object must not be passed to a function.

Language

Verilog

Message Details

```
Null Object Passed to the function <function>. Please contact
Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_912

The file name passed to the function was not found.

Language

Verilog

Rule Description

SpyGlass generates this error if a file passed to a function is not found in the current work library or other specified libraries. For example, consider the following test case (test.v):

```
‘include “abc.v”
   module test;
end module
```

In this case, if the following API is called, STX_VE_912 will be flagged:

```
veFileGetEndLineNo(“cab.v”)
```

Message Details

```
File '<file>' passed to the function '<function>' not found.
Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_913

ASSERT: Unsupported expression type in connection expression.

Language

Verilog

Message Details

```
ASSERT : Unsupported expression type in connection expression.
Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_914

ASSERT: In connection expressions, objects other than net are not supported.

Language

Verilog

Message Details

```
ASSERT : Object other than net in connection expression, is not
supported yet. Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_915

The Stack passed to function must not be empty.

Language

Verilog

Message Details

```
The Stack passed to function <function> is empty. Please
contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_916

Specified argument is NULL.

Language

Verilog

Message Details

```
Argument is NULL. Please contact Atrenta Support
```

Severity

INTERNAL_FATAL

STX_VE_918

Incorrect expression string is specified.

Language

Verilog

Message Details

```
Incorrect expression string ( <string> )
```

Severity

Syntax

STX_VE_919

The expression size is zero.

Language

Verilog

Message Details

```
Expression size is zero
```

Severity

Syntax

STX_VE_1181

Genvar should be assigned a non-negative value.

Language

Verilog

Message Details

```
Genvar should be assigned a non-negative value
```

Severity

Syntax

STX_VE_1182

Genvar is expected as identifier.

Language

Verilog

Message Details

```
Genvar expected as identifier( <identifier> )
```

Severity

Syntax

STX_VE_1183

The two genvar assignments in a generate loop should refer to the same genvar.

Language

Verilog

Message Details

```
The two genvar assignments in a generate loop should refer to
the same genvar
```

Severity

Syntax

STX_VE_1184

Too many select indices

Description

Number of select indices cannot be greater than the number of dimensions specified in a declaration.

The following example shows the violating and the non violating case of this rule:

```
bit a [4][4];
assign a[1][1][1] = 1;  // Violation
assign a[1][1] = 1;   // Fine
```

Language

Verilog

Message Details

```
Too many select indices for (<index> )
```

Severity

Syntax

STX_VE_INACTIVE_1184

Too many select indices

Description

Number of select indices cannot be greater than the number of dimensions specified in a declaration.

The following example shows the violating and the non violating case of this rule:

```
bit a [4][4];
assign a[1][1][1] = 1;  // Violation
assign a[1][1] = 1;   // Fine
```

NOTE: *The STX_VE_INACTIVE_1184 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Language

Verilog

Message Details

`Too many select indices for (<index> )`

Severity

Syntax

STX_VE_1185

More than one word cannot be selected for multi-dimensional arrays.

Description

More than one word cannot be selected for multi-dimensional arrays.

The following example shows the violating and non violating case of this rule:

```
reg [63:0] array [0:4095][0:127];
assign temp=array[100];       // Violation
assign temp=array[100][1];  // No violation
```

Language

Verilog

Message Details

`More than one word cannot be selected for multi-dimensional arrays`

Severity

Syntax

STX_VE_INACTIVE_1185

More than one word cannot be selected for multi-dimensional arrays.

Description

More than one word cannot be selected for multi-dimensional arrays.

The following example shows the violating and non violating case of this rule:

```
reg [63:0] array [0:4095][0:127];
assign temp=array[100];       // Violation
assign temp=array[100][1];   // No violation
```

NOTE: *The STX_VE_INACTIVE_1185 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Language

Verilog

Message Details

`More than one word cannot be selected for multi-dimensional arrays`

Severity

Syntax

STX_VE_1186

Two generate loops using the same genvar as an index variable shall not be nested.

Language

Verilog

Rule Description

Two generate loops using the same genvar as an index variable shall not be nested. For example:

```
genvar i;
for ( i =0 ;i <5;i++) begin :A
  for ( i =0 ;i <5;i++) begin :B
    ...
  end
end
```

Message Details

`Two nested generate loops using the same genvar '<variable>'`

Severity

Syntax

STX_VE_1187

Mixed-style port declaration is not allowed.

Language

Verilog

Message Details

```
Mixed-style port declaration is not allowed
```

Severity

Syntax

STX_VE_1188

Invalid context for usage of genvar.

Language

Verilog

Rule Description

Usage of genvar is illegal in the following cases:

- A genvar cannot be used outside generate loop. It can only be used inside the generate loop that it indexes. For example:

```
module top;
  genvar a,b;
  int c;
  generate
    for ( a = 0; a < 2; a = a + 1)
    begin
      assign c = a;
      assign c = b;
    end
  endgenerate
endmodule
```

- The first genvar assignment in for-loop shall not reference the loop index variable on the right hand side. For example:

```
module top;
genvar a;
generate
  for ( a = a + 1; a < 2; a = a + 1)
    ...
endgenerate
endmodule
```

Message Details

`Invalid context for genvar '<genvar-name>'. <message>.`

Where, *<message>* can be any of the following:

- A genvar can be used only inside of the generate loop that it indexes
- The first genvar assignment in for-loop shall not reference the loop index variable on the right hand side

Severity

Syntax

STX_VE_1189

Invalid width specified for indexed part-select.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the width expression specified with an indexed part-select is not a positive integer expression.

Message Details

```
Invalid width specified for part-select of '<variable-name>'
```

Severity

Syntax

STX_VE_INACTIVE_1189

Invalid width specified for indexed part-select.

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the width expression specified with an indexed part-select is not a positive integer expression.

NOTE: *The STX_VE_INACTIVE_1189 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Invalid width specified for part-select of '<variable-name>'
```

Severity

Syntax

STX_VE_1190

The total number of dimensions selected does not match with actual number of dimensions of the parent of the selected object.

Language

Verilog

Rule Description

In hierarchical reference to an element, the number of dimensions of hierarchical identifier should be correct. For example,

```
typedef struct { bit [3:0] b;} [2:0] bb [3:0];
bb a;
assign o=a[2][2][0].b[1]; //Number of dimensions
                          //more than actual

assign a[2].b[1]=1; //Number of dimension less
                    //than actual
```

Message Details

```
Invalid dimension selection in hierarchical identifier
'<identifier>'
```

Severity

Syntax

STX_VE_1191

C-style array dimension declaration size must be greater than zero

Language

Verilog

Rule Description

C-style array dimension declaration size must be greater than zero. For example,

```
bit a[0];//violation as size of the dimension is zero
```

Message Details

```
C-style array dimension declaration size must be greater than zero
```

Severity

Syntax

STX_VE_1192

if break/continue statement does not have any enclosing loop statement, then it should be an error

Language

Verilog

Rule Description

Syntactically, break/continue statement can appear within a block. However, if there is no enclosing loop then it should be an error. For example, SpyGlass flags an error in the following case:

```
always @*
  begin
    i = a + b;
    break;
  end
```

Message Details

```
<break | continue> statement not within any loop statement
```

Severity

Syntax

STX_VE_1193

Break/Continue statement within fork...join cannot be used to control the loop outside the fork...join statement

Language

Verilog

Rule Description

This error is generated when the break/continue statement within fork-join block is used to control the loop outside the fork-join block. For example, the STX_VE_1193 flags an error in the following case:

```
for(i =0; i< 4; i++)
  begin
    fork
      break;
    join
  end
```

Message Details

```
<break | continue> statement within fork...join cannot be used
to control the loop outside the fork...join statement
```

Severity

Syntax

STX_VE_1194

Design statement with no top cell is not allowed.

Description

The *STX_VE_1194* rule reports a violation if no top-level cell name is specified in the design statement of a configuration block.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SYNTAX] Empty design statement is not allowed

Consequences of Not Fixing

SpyGlass run does not proceed further.

How to Debug and Fix

Modify the design statement of the configuration to specify a top-level cell name.

Example Code and/or Schematic

The following example shows the violating and non-violating scenarios of this rule:

```
config cfg;
  design ;     // Violation
endconfig
config cfg1;
 design top; // No violation
endconfig
```

Default Severity Label

Syntax

STX_VE_1195

Multiple default clauses in a configuration block are not allowed.

Description

The *STX_VE_1195* rule reports a violation if multiple default clauses are present in a configuration block.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

```
[SYNTAX] Multiple default clause in configuration block are not
allowed. Previous default clause found in file ( <file-name> )
at line ( <line-num> )
```

Consequences of Not Fixing

SpyGlass run does not proceed further.

How to Debug and Fix

Retain only one default clause in the configuration.

Example Code and/or Schematic

The following example shows the violating scenario of this rule:

```
config cfg1;
  design top;
  default liblist L1 L2;
  default liblist L1 L2 L3;   // Violation
endconfig
```

Default Severity Label

Syntax

STX_VE_1196

Configuration cell clause contains both a cell with a library name and a liblist expansion clause.

Description

The *STX_VE_1196* rule reports a violation if a configuration cell clause contains both the cell with a library name and a `liblist` expansion clause.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SYNTAX] `Illegal use of liblist clause in a cell clause for cell with library name`

Consequences of Not Fixing

SpyGlass run does not proceed further.

How to Debug and Fix

Replace the `liblist` clause with the `use` clause.

Example Code and/or Schematic

The following example shows the violating and non-violating scenarios of this rule:

```
config cfg;
  design top;
 cell mid liblist L2;      // No violation
  cell L1.mid liblist L2;      // Violation
endconfig
```

Default Severity Label

Syntax

STX_VE_1197

Assignments not allowed in property expression, when used directly in assertion.

Language

Verilog

Rule Description

Only local variables can be used in LHS of assignment expression when used in sequence match item of a property/sequence expression. Since, no local variables can be defined inside an assertion, so assignment is not allowed when used directly in a property expression inside an assertion.

For example,

```
assert property ((p1, q=1));
```

will give error.

Message Details

```
Assignments not allowed in property expression, when used
directly in assertion
```

Severity

Syntax

STX_VE_1198

Illegal operator used inside a property/sequence expression.

Language

Verilog

Rule Description

SystemVerilog does not allow several operators to be given inside a sequence/property. Some of these include C assignment operators such as += and the C increment and decrement operators, ++ and --. This restriction has been put to prevent side effects.

Message Details

```
Operator (<operator>) not allowed inside a property/sequence
expression
```

Severity

Syntax

STX_VE_1199

The identifier on the LHS of assignment operator is not defined as local variable.

Language

Verilog

Rule Description

This error is generated when variable in LHS of assignment operator is not defined as local variable in Sequence/Property. Only local variables are allowed in LHS of assignment operator.

For example, the following case will give an error.

```
int k,j;
property P(clk);
  (j,k=j);
endproperty : P1
```

Message Details

```
Identifier ( <identifier> ) is not defined as local variable in
sequence/property ( <name> )
```

Severity

Syntax

STX_VE_1200

Construct name does not match with label name specified with end of construct.

Language

Verilog

Rule Description

This error is generated when the label with end of construct, does not match with the construct name.

For example, the following case will give an error.

```
property P(clk);
  clk;
endproperty : P1
```

Message Details

`<type> name (<design-unit>) does not match with end <type> label name (end-label-name>)`

Where, *<type>* can be `interface`, `module`, `package`, `class`, or `coverage`.

Severity

Syntax

STX_VE_1201

Begin label/block name does not match will end label name.

Language

Verilog

Rule Description

This error is generated when the name after the end construct does not match the label given at the beginning of a begin-end block. It will also be flagged for the cases where the label for a block is not specified, and there is a label after the end construct. For example,

```
module name1(input i, output reg o);
  always begin
    o = i;
    end : name2
//the name of the sequential block is not specified.
endmodule : name1
```

An error message would be flagged for name2.

Message Details

```
Begin <label | block> name ( <name> ) does not match with end
label name ( <end-label-name> )
```

Severity

Syntax

STX_VE_1206

Fork label name does not match with (join/join_none/join_any) label name.

Language

Verilog

Rule Description

This error is generated when the label after the join/join_none/join_any construct does not match the label following fork.

For example, the following case will give an error.

```
fork:F1
  #20 r1 = in1 - in2;
  #30 out1 = r1;
  #50 r1 = in1 * in2;
join_any:F2
```

Message Details

```
Fork label name ( <fork-label-name> ) does not match with
<join/join_none/join_any> label name (<name>)
```

Severity

Syntax

STX_VE_1207

An event control statement in an always_comb or always_latch is not allowed.

Language

Verilog

Rule Description

For always_comb and always_latch blocks, the sensitivity list is automatically inferred, because the software tools know that the intent is to represent combinational / latched logic. Explicitly specifying an event control make it a case of multiple event control, an error is thus flagged.

For example, this error would be issued in the following case:

```
always_latch@(in2) //event control is not allowed here
  begin
    if(in2==1)
      out1<=in1;
end
```

Message Details

```
An event control statement in an always_comb or always_latch
block is not allowed
```

Severity

Syntax

STX_VE_1208

Simultaneous usage of the unique and priority qualifier in if_else_if statement is not allowed without sequential block usage

Language

Verilog

Rule Description

The unique qualifier indicates that the order of decision is not important. This allows the decision series to be executed in parallel. The priority qualifier indicates that the order of decision is important. In this case, the software tools must maintain the order of decision sequence. Since the implementations of unique and priority contradict each other, it is an error for a SystemVerilog design to contains a chain of if-else-if statements with conflicting unique or priority keywords.

To nest another unique/priority if statement within such a series of if-else-if conditions, a begin...end block should be used.

Message Details

```
Mixing of unique and priority qualifier in if_else_if statement
is not allowed without sequential block usage
```

Severity

Syntax

STX_VE_1209

Assignment/comparison to non-null/non-event/chandle expression is violated.

Language

Verilog

Rule Description

It is illegal to assign or compare chandle/event variable with variable of any other data type. They can only be assigned/compared with elements of same type or null.

For example,

```
chandle l;
event ev;
int in2;
always l=in2;
//chandle assigned to non-chandle/non-null
always ev=in2;
//event assigned to non-chandle/non-null
```

Message Details

```
Assignment/comparison to non-null/non-event/chandle expression
is violated
```

Severity

Syntax

STX_VE_1210

Use of chandle as an untagged union member type is not allowed.

Language

Verilog

Rule Description

It is illegal to use chandle data type in unions. For example, error would be flagged for chandle c1 and c2 being used in union in the following case.

```
union {bit l;chandle c1;
  struct { chandle c2;bit b;} st1;} un;
```

Message Details

```
Invalid use of chandle as untagged union member type
```

Severity

Syntax

STX_VE_1211

Non-constant Repetition multiplier in concatenation can only be used with lvalue as string type

Language

Verilog

Rule Description

String is a variable size, dynamically allocated array of bytes. When using the string data type instead of an integral variable, strings can be of arbitrary length and no truncation occurs. Therefore, for replication operator (also called a multiple concatenation), a non-constant repetition multiplier is allowed only if the left hand side of assignment contains variable of type string.

Error would be issued for any other data type in left hand side. For example,

```
integer i;
bit [1:0] by1;
bit [3:0] bytes;

assign bytes={i{by1}};
// i used as repetition multiplier is non-constant
```

Message Details

```
Non-constant repetition multiplier (<multiplier>) in
concatenation can only be used with lvalue as string type
```

Severity

Syntax

STX_VE_1212

Non-string variable(s) or non-string literal(s) cannot be used in concatenation when lvalue is string type

Language

Verilog

Rule Description

When left hand side of an assignment is of type string, only a string literal or variable can be used for concatenation assignment. For example,

```
string st1,st2;
int in1;
assign st1={in1,st2};
```

Error would be issued for non-string variable in1 used in concatenation assignment to string st1.

Message Details

```
Non-string variable(s) or Non-string literal(s) (<name>) cannot be used in concatenation when lvalue is of string type
```

Severity

Syntax

STX_VE_1216

Implicit port connection (.*) can appear only once in the port list.

Language

Verilog

Rule Description

A .* implicit instantiation, such as:

```
Mod inst(.*,.*)
```

where multiple .* specified is not permitted.

Message Details

```
Implicit port connection ( .* ) can appear at most once in the
port list
```

Severity

Syntax

STX_VE_1217

There is a mismatch in port declaration of module/interface with it's declaration in prototype.

Language

Verilog

Rule Description

Mismatch in port declaration of module/interface with its declaration in prototype.

Message Details

```
Mismatch in port declaration of <module | interface> with its
declaration in prototype
```

Severity

Syntax

STX_VE_1218

'.*' style port specified where no matching module/interface prototype was found.

Language

Verilog

Rule Description

'.*' style port specified where no matching module/interface prototype found.

Message Details

```
'.*' style port specified where no matching <module |
interface> prototype found
```

Severity

Syntax

STX_VE_1219

Port Connection between two incompatible data types is not allowed.

Language

Verilog

Rule Description

Port connection between two incompatible data types is not allowed.

For example:

```
module X(input userDef1 in )
...
endmodule

module Y ( input userDef2 in)
...
X  inst( .in(in);
endmodule
```

In this example, in the instantiation of module X, port connection is between two incompatible data types (userDef1 and userDef2), which is not valid.

Message Details

```
Port connection ( <connection> ) between two incompatible data
types ( <type1> and <type2>  )
```

Severity

Syntax

STX_VE_1220

Multiple Declaration of TimeUnit/TimePrecision in same scope is not permissible.

Language

Verilog

Rule Description

Multiple declarations of TimeUnit/TimePrecision in same scope are not permissible.

Message Details

```
Multiple declaration of TimeUnit/TimePrecision in same scope is
not permissible
```

Severity

Syntax

STX_VE_1221

Implicit port-connect declaration for .name port connection is not allowed.

Language

Verilog

Rule Description

The identifier referred by .port_identifier shall not create an implicit wire declaration.

Message Details

```
Identifier ( <identifier> ) of implicit .name port connection
not declared
```

Severity

Syntax

STX_VE_1223

Mixing .* and positional port connection not allowed.

Language

Verilog

Rule Description

Mixing .* and positional port connection is not allowed as in the following example:

```
Mod X ( .*,a,b);
```

Message Details

```
mixing .* and positional port connection not allowed
```

Severity

Syntax

STX_VE_1224

A variable data type is not permitted on either side of an inout port.

Language

Verilog

Rule Description

A variable data type is not permitted on either side of inout port.

Message Details

```
A variable data type ( <data-type> ) is not permitted on either
side of inout port
```

Severity

Syntax

STX_VE_1225

Non-net variable cannot be an inout port.

Language

Verilog

Rule Description

Non-net variable cannot be an inout port.

Message Details

```
Non-net variable '<variable>' cannot be an inout port
```

Severity

Syntax

STX_VE_1226

Variable/net specified in modport port list not declared inside interface.

Language

Verilog

Rule Description

Variable/net specified in modport port list must be declared inside the same interface where modport is defined.

Message Details

```
Variable/net ( <name> ) specified in modport port list not declared inside interface
```

Severity

Syntax

STX_VE_1227

Re-declaration of symbol in modport is not allowed.

Language

Verilog

Rule Description

A task/function prototype once specified in a modport cannot be re-specified in the same modport inside an interface.

Message Details

```
Re-declaration of symbol <symbol> as <declaration> in modport
<modport>
```

Severity

Syntax

STX_VE_1228

Type incompatibility for an interface port

Language

Verilog

Rule Description

An interface port declared in module or interface port list can only be connected with interface instance or interface port during its instantiation.

Message Details

```
Type incompatibility for interface port
```

Severity

Syntax

STX_VE_1230

Modport specified in the instance is different from the modport in the master.

Language

Verilog

Rule Description

SpyGlass reports this syntax error if a modport specified in an instance is different from the modport in the master.

For example, the *STX_VE_1230* rule reports a violation in the following case:

```
interface i2;
reg a, b, c, d;
modport master (output a, b, input c, d);
modport slave (output a, b, input c, d);
endinterface

module m (i2.master i);
endmodule


module top;
  i2 i();
  m u1(.i(i.slave));
endmodule
```

Message Details

```
Modport name ( <modport-name1> ) in the instance ( <instance1>
) does not match the modport name ( <modport-name2> ) in the
<instance2> ( <master> )
```

Severity

Syntax

STX_VE_1231

Module or UDP instantiation inside interface declaration is not allowed

Language

Verilog

Rule Description

Module or UDP instantiation inside interface declaration is not allowed. The interface port connection should be specified.

Message Details

```
Module or UDP instantiation (<instantiation>) inside interface
declaration is not allowed
```

Severity

Syntax

STX_VE_1232

The declaration statement is invalid

Language

Verilog

Rule Description

The *STX_VE_1232* rule reports a violation when the declaration statement is invalid. This is explained in the following examples:

Example 1

The following example shows the use of unknown type, that is, missing `typedef` declaration:

```
module top;
  TYPE1 a;   //Violation: TYPE1 is not declared before use
  typedef int TYPE2;
  TYPE2 a;   //No violation: TYPE2 is declared before use
endmodule
```

Example 2

```
interface intf;
 ...
endinterface
module top1();
 intf p1;     // Violation: p1 not mentioned in port list
endmodule
module top(p1);
 intf p1;     // No violation
endmodule
```

Example 3

In the following example, parenthesis is missing in the declaration is actually an instance missing parenthesis

```
interface intf;
  ...
```

```
endinterface
module top1();
 intf p1;         // Violation
 endmodule
module top();
 intf p1();    // No violation
endmodule
```

Message Details

Invalid declaration (<element-name>). Either (<type>) is unknown type or (<element-name>) is not mentioned in module header

Severity

Syntax

STX_VE_1234

This rule is deprecated.

The functionality of this rule is covered by the *STX_VE_799* rule.

STX_VE_1237

There is a mismatch in port declaration of task/function with that of its prototype declaration.

Language

Verilog

Rule Description

Task/Function prototype declaration must match with the actual definition.

Message Details

```
Mismatch in port declaration of task/function (<name>) with
that of its prototype declaration
```

Severity

Syntax

STX_VE_1238

When importing a function/task from another module a full prototype should be used.

Language

Verilog

Rule Description

When importing a function/task from another module through interface, a full function/task prototype should be used.

Message Details

```
When importing a function/task (<name>) from another module, a
full prototype should be used
```

Severity

Syntax

STX_VE_1240

Interface/Program does not have any definition

Language

Verilog

Rule Description

If an extern declaration for any interface/program and its definition is missing, then using such interfaces/programs either in the module header or instantiating them is not supported.

Message Details

```
Interface/Program (<name>) without any definition is not supported
```

Severity

Syntax

STX_VE_1241

Virtual interface used as port not allowed.

Language

Verilog

Rule Description

Virtual used as port not allowed. However, it can be used as task/function arguments.

Message Details

```
<virtual-interface> used as port not allowed
```

Severity

Syntax

STX_VE_1242

Virtual interface used as Interface/union items not allowed.

Language

Verilog

Rule Description

Virtual interface used as Interface items or Union members not allowed.

Message Details

```
virtual interface used as <interface | union> not allowed
```

Severity

Syntax

STX_VE_1243

Expression uses virtual interface in an illegal context.

Language

Verilog

Rule Description

Virtual interface can only be used with == and != in an expression.

Message Details

```
Expression uses virtual interface (<interface>) in an illegal
context
```

Severity

Syntax

STX_VE_1244

Unconnected interface port for an instance.

Language

Verilog

Rule Description

Any interface/module cannot have an unconnected interface port at time of its instantiation. For example, SpyGlass flags the STX_VE_1244 message in the following case:

```
module mid(intf i1);
endmodule

module top;
  mid m1();
endmodule
```

Message Details

`Unconnected interface port (<port>) for instance (<instance>)`

Severity

Syntax

STX_VE_INACTIVE_1244

Unconnected interface port for an instance.

Language

Verilog

Rule Description

Any interface/module cannot have an unconnected interface port at time of its instantiation. For example, SpyGlass flags the STX_VE_1244 message in the following case:

```
module mid(intf i1);
endmodule

module top;
  mid m1();
endmodule
```

NOTE: *The STX_VE_INACTIVE_1244 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Unconnected interface port (<port>) for instance (<instance>)`

Severity

Syntax

STX_VE_1245

Illegal access to interface signal. Signal not visible via modport.

Language

Verilog

Rule Description

Access to any of the interface signal that is not part of a modport is not visible via that modport.

Message Details

```
Invalid access to interface signal ( <signal> ). Signal not
visible via modport
```

Severity

Syntax

STX_VE_1246

The definition is missing for the imported task/func.

Language

Verilog

Rule Description

Any task/function imported within a modport and which is not exported via another module must have its definition within the interface itself.

Message Details

`Definition missing for the imported task/func ( <name> )`

Severity

Syntax

STX_VE_1247

Modport is not defined.

Language

Verilog

Rule Description

Modport is not found declared for (interface.modport_name) port interface declaration.

Message Details

```
Modport (<modport>) is not defined
```

Severity

Syntax

STX_VE_1248

Exported function / task not defined in module.

Language

Verilog

Rule Description

The exported task/function is not defined in the module that it is being exported to.

Message Details

```
Exported function/task ( <name> ) not defined in module (
<module> ) for modport ( <modport> ) of interface ( <interface>
)
```

Severity

Syntax

STX_VE_INACTIVE_1248

Exported function / task not defined in module.

Language

Verilog

Rule Description

The exported task/function is not defined in the module that it is being exported to.

NOTE: *The STX_VE_INACTIVE_1248 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Exported function/task ( <name> ) not defined in module ( <module> ) for modport ( <modport> ) of interface ( <interface> )
```

Severity

Syntax

STX_VE_1250

Only net types can be aliased.

Language

Verilog

Rule Description

Only net types, that is, net of the types wire, tri, tri1, supply0, wand, triand, tri0, supply1, wor, and trior can be aliased.

Message Details

```
Only net types can be aliased
```

Severity

Syntax

STX_VE_1251

Only net of the same type can be aliased.

Language

Verilog

Rule Description

Only net of the same type can be aliased for example a wire net cannot be aliased with a wor net.

Message Details

```
Only nets of the same type can be aliased
```

Severity

Syntax

STX_VE_1252

Size mismatch between the two aliased nets.

Language

Verilog

Rule Description

There is a size mismatch between two of the nets, in the group of aliased nets.

Message Details

```
Size mismatch in aliased nets '<net1>' and '<net2>'
```

Severity

Syntax

STX_VE_1260

A parameter declared as type / value and overridden with a value / type is not permitted.

Language

Verilog

Rule Description

Declaring a parameter as type and then overriding it as a value (or vice-versa) is illegal.

For example, the following code will give an error.

```
parameter type x =int
top  #(.x(1)) inst()
```

Message Details

```
Parameter ( <parameter> ) declared as <type> cannot be
overridden with <value>
```

Severity

Syntax

STX_VE_INACTIVE_1260

A parameter declared as type / value and overridden with a value / type is not permitted.

Language

Verilog

Rule Description

Declaring a parameter as type and then overriding it as a value (or vice-versa) is illegal.

For example, the following code will give an error.

```
parameter type x =int
top #(.x(1)) inst()
```

NOTE: *The STX_VE_INACTIVE_1260 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Parameter ( <parameter> ) declared as <type> cannot be
overridden with <value>
```

Severity

Syntax

STX_VE_1262

Item referenced from a package not found in that package.

Language

Verilog

Rule Description

When an item is referenced from a package, the item should be present within that package.

Message Details

`Item ( <item> ) not found in package ( <package> )`

Severity

Syntax

STX_VE_1263

Built in package "std" cannot be replaced.

Language

Verilog

Rule Description

Package std cannot be redefined.

Message Details

```
Builtin package 'std' cannot be redefined
```

Severity

Syntax

STX_VE_1264

If an item is found through multiple wildcard imports, then the usage of the item makes the case invalid.

Language

Verilog

Rule Description

If an item is wildcard imported into a scope from two different packages, an error results, if the item is used.

Message Details

```
Multiple definitions of ( <item> ) found through wildcard
import from the packages ( <package1> ) and ( <package2> )
```

Severity

Syntax

STX_VE_1265

An item can only be imported if not previously defined in that scope.

Language

Verilog

Rule Description

An explicit import is illegal if the imported item is defined in the same scope or explicitly imported from another package. Importing an item from the same package multiple times is allowed.

Message Details

```
Imported item ( <item> ) already defined in file ( <file> ), at
line ( <line-num> )
```

Severity

Syntax

STX_VE_1266

Wire declarations with implicit continuous assignments are not allowed within package.

Language

Verilog

Rule Description

Implicit continuous assignment is not allowed within package. For example:

```
package pkg_name;
wire w = a; // Error
endpackage
```

Message Details

```
Implicit continuous assignment is not allowed within package
```

Severity

Syntax

STX_VE_1267

Items within packages cannot have hierarchical references.

Language

Verilog

Rule Description

Within a package, any hierarchical reference to an item outside the package scope is not allowed.

Message Details

```
Items within packages cannot have hierarchical references
```

Severity

Syntax

STX_VE_INACTIVE_1267

Items within packages cannot have hierarchical references.

Language

Verilog

Rule Description

Within a package, any hierarchical reference to an item outside the package scope is not allowed.

NOTE: *The STX_VE_INACTIVE_1267 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Items within packages cannot have hierarchical references
```

Severity

Syntax

STX_VE_1268

Package used in hierarchical reference

Description

The *STX_VE_1268* rule reports a violation if a package is used in a hierarchical reference using the dot (`.`) operator.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SYNTAX] `Illegal use of package ( <package> ) in hierarchical reference`

Consequences of Not Fixing

SpyGlass analysis does not proceed further.

How to Debug and Fix

Use the package-scope resolution operator `::` to refer to the items defined inside the package.

Example Code and/or Schematic

Consider the following example:

```
package pkg;
parameter int pa = 4;
endpackage
module top;
...
assign x = pkg.pa;      // Error
assign x = pkg::pa;   // Correct
...
endmodule
```

For the above example, the *STX_VE_1268* rule reports a violation for the incorrect usage of the `pa` package in the hierarchical reference using the dot operator.

However, the correct usage is shown in the next line in which the package is used with the `::` operator.

Default Severity Label

Syntax

STX_VE_INACTIVE_1268

Package used in hierarchical reference

Description

The *STX_VE_INACTIVE_1268* rule reports a violation if a package is used in a hierarchical reference using the dot (`.`) operator.

NOTE: *The STX_VE_INACTIVE_1268 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SYNTAX] `Illegal use of package ( <package> ) in hierarchical reference`

Consequences of Not Fixing

SpyGlass analysis does not proceed further.

How to Debug and Fix

Use the package-scope resolution operator `::` to refer to the items defined inside the package.

Example Code and/or Schematic

Consider the following example:

```
package pkg;
parameter int pa = 4;
endpackage
module top;
...
assign x = pkg.pa;      // Error
```

```
assign x = pkg::pa;     // Correct
...
endmodule
```

For the above example, the *STX_VE_INACTIVE_1268* rule reports a violation for the incorrect usage of the `pa` package in the hierarchical reference using the dot operator.

However, the correct usage is shown in the next line in which the package is used with the `::` operator.

Default Severity Label

Syntax

STX_VE_1269

Item specified using $unit:: need to be declared in the compilation unit scope.

Language

Verilog

Rule Description

Using an item through $unit:: which is not present in the compilation unit scope is erroneous.

Message Details

```
Item ( <item> ) not declared in $unit scope
```

Severity

Syntax

STX_VE_1270

For a block, block name and label cannot co-exist.

Language

Verilog

Rule Description

It is illegal to have both a label before begin or fork and a block name after the begin or fork.

Message Details

```
Named block cannot have label
```

Severity

Syntax

STX_VE_1274

Loop index variable and array variable should not have identical name.

Language

Verilog

Rule Description

SpyGlass reports this syntax error to indicate that a loop index variable and an array variable have identical names.

For example, this rule reports a violation in the following case because the array variable A and foreach loop index have identical names:

```
bit A [1:0][2:0];
foreach ( A [ A, B ] )
```

Message Details

```
Identical names for array variable and foreach-loop index
```

Severity

Syntax

STX_VE_1275

It is illegal to assign values to foreach-loop index variables.

Language

Verilog

Rule Description

SpyGlass reports this syntax error to indicate that foreach loop index variables are being assigned values.

Such variables are read-only variables and cannot be assigned values.

For example, this rule reports a violation in the following case because the foreach loop index variable `i` is being assigned a value:

```
bit A [1:0][2:0]
foreach ( A [ i, j ] )
begin
i = i | j;
```

Message Details

```
Loop index variable ( <var-name> ) cannot be assigned a value.
In foreach-loop, index variables are read only
```

Severity

Syntax

STX_VE_1276

Number of foreach-loop index variables should not exceed the number of dimensions of array variable.

Language

Verilog

Rule Description

SpyGlass reports this syntax error to indicate that the number of foreach loop index variables is not less than or equal to the dimensions of an array variable.

For example, this rule reports a violation in the following case because four forloop index variables are specified for the array a of three dimensions:

```
bit [1:0][2:0][3:0] a;
foreach ( a [ i, j, k, m] )
```

However, this rule does not report a violation in the following case because two forloop index variables are specified for the array a of three dimensions:

```
bit [1:0][2:0][3:0] b;
foreach ( b [ i, j ] )
```

Message Details

```
Number of foreach-loop index variables ( <num> ) is more than
the dimensions specified ( <dimension> ) for array ( <array> )
```

Severity

Syntax

STX_VE_INACTIVE_1276

Number of foreach-loop index variables should not exceed the number of dimensions of array variable.

Language

Verilog

Rule Description

SpyGlass reports this syntax error to indicate that the number of foreach loop index variables is not less than or equal to the dimensions of an array variable.

For example, this rule reports a violation in the following case because four forloop index variables are specified for the array a of three dimensions:

```
bit [1:0][2:0][3:0] a;
foreach ( a [ i, j, k, m] )
```

However, this rule does not report a violation in the following case because two forloop index variables are specified for the array a of three dimensions:

```
bit [1:0][2:0][3:0] b;
foreach ( b [ i, j ] )
```

NOTE: *The STX_VE_INACTIVE_1276 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Number of foreach-loop index variables ( <num> ) is more than
the dimensions specified ( <dimension> ) for array ( <array> )
```

Severity

Syntax

STX_VE_1330

Only those variables declared at module level can have an initial value assignment.

Language

Verilog

Message Details

```
Only variables declared at module level can have initial value
assignment
```

Severity

Syntax

STX_VE_1350

In $random function, the seed variable can only be of integral type

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the seed argument in $random function is not valid. The seed argument should only be of integral type.

For example,

```
reg a;
int b[1:0];
int c;
assign w = $random(c);    // Correct
assign w = $random(a);    // Correct
assign w = $random(b);    // Error
```

Message Details

`In \$random function, seed variable can only be of integral type`

Severity

Syntax

STX_VE_INACTIVE_1350

In $random function, the seed variable can only be of the integral type

Language

Verilog

Rule Description

SpyGlass generates this syntax error to indicate that the seed argument in $random function is not valid. The seed argument should only be of the integral type.

For example,

```
reg a;
int b[1:0];
int c;
assign w = $random(c);    // Correct
assign w = $random(a);    // Correct
assign w = $random(b);    // Error
```

NOTE: *The STX_VE_INACTIVE_1350 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`In \$random function, seed variable can only be of integral type`

Severity

Syntax

STX_VE_1351

Un-terminated library file list is not allowed.

Language

Verilog

Message Details

```
Un-terminated library file list not allowed
```

Severity

Syntax

STX_VE_1360

Invalid base used in based number

Language

Verilog

Rule Description

SpyGlass reports this error if an invalid base is used in a based number. Legal base specifications are d, D, h, H, o, O, b, or B for the bases decimal, hexadecimal, octal, and binary, respectively.

Message Details

```
Invalid base (<base>) used in based number
```

Severity

Syntax

STX_VE_1366

Scalar register must be specified for timing check notifier.

Language

Verilog

Message Details

```
Illegal timing check notifier, scalar register expected (<net/
variable-name>)
```

Severity

Syntax

STX_VE_1367

Invalid argument is provided. line_number is expected.

Language

Verilog

Message Details

`Invalid argument, line_number expected (<lex-token-text>)`

Severity

Syntax

STX_VE_1368

Invalid argument is provided. level_code is expected.

Language

Verilog

Message Details

`Invalid argument, level_code expected (<lex-token-text>)`

Severity

Syntax

STX_VE_1383

Named argument in a sequence / property / function call should be associated once.

Language

Verilog

Rule Description

SystemVerilog supports named argument used in a sequence/property/function call to be associated once. For example:

```
property p(a,b,c);
p(.a(3),.b(2),.a(1));
// not allowed since a is already associated.
endproperty
```

Message Details

```
Named argument ( <argument> ) in <sequence/property/function>
call ( <name> ) is already associated
```

Severity

Syntax

STX_VE_1384

Call to sequence/property/function with invalid named argument is not allowed.

Language

Verilog

Rule Description

SystemVerilog supports call to sequence/property/function only with named argument matching the arguments of sequence/function/property declaration. For example:

```
property p(a,b,c);
p(.a(3),.b(2),.x(1));
// not allowed since x is not a formal argument
// for property p.
endproperty
```

Message Details

```
Call to <sequence/property/function> ( <name> ) defined at file
( <file> ), line ( <line-num> ) with invalid named argument (
<argument> )
```

Severity

Syntax

STX_VE_1385

All formal arguments should be associated by named argument, if default values are not specified.

Language

Verilog

Rule Description

SystemVerilog supports all formal argument of property/sequence/function should be associated with an argument in a sequence/property/function call, if default value of arguments is not specified. For example:

```
property p(a,b,c);
  p(.a(3),.b(2));
endproperty
```

The above is not allowed since c is not associated with any named argument.

Message Details

```
Unassociated formal argument ( <argument> ) in <property/
sequence/function> call ( <name> )
```

Severity

Syntax

STX_VE_1386

LHS of constant range should be less than or equal to the RHS.

Language

Verilog

Rule Description

SystemVerilog supports ranges with concatenation and repetition operators connecting sequence expressions. These ranges should be constant and LHS should be less than or equal to the RHS.

Message Details

```
LHS ( <LHS-expression> ) of constant range should be less than
or equal to the RHS ( <RHS-expression> )
```

Severity

Syntax

STX_VE_1387

The expression must result in an integer value greater than or equal to 0.

Language

Verilog

Rule Description

SystemVerilog supports constant expressions with concatenation and repetition operators connecting sequence expressions. Any such constant expression must result in an integer value greater than or equal to 0.

Message Details

```
The expression ( <expression> ) must result in an integer value
greater than or equal to 0
```

Severity

Syntax

STX_VE_1388

Property (instance/operator) not allowed inside a sequence expression.

Language

Verilog

Rule Description

SystemVerilog supports properties and sequences, both of which have a few operators in common. However, there are certain property operators (such as |->, |=>) which are not allowed in a sequence and some operators (like within, intersect) can be used.

Message Details

```
Property <instance | operator> ( <name> ) not allowed inside a
sequence expression
```

Severity

Syntax

STX_VE_1389

Property having disable iff clause cannot be used with a property operator.

Language

Verilog

Rule Description

SystemVerilog does not allow nesting of 'disable iff' clauses, explicitly or through property instantiations because the disable iff is not allowed in a property expression.

Message Details

`Property ( <property> ) having 'disable iff' clause used with a property operator`

Severity

Syntax

STX_VE_1390

Nesting of 'disable iff' clause through property instantiation is not allowed.

Language

Verilog

Rule Description

SystemVerilog does not allow nesting of 'disable iff' clauses, explicitly or through property instantiations.

Message Details

```
Nesting of \'disable iff\' clause through property
instantiation ( <instantiation> ) is not allowed
```

Severity

Syntax

STX_VE_1391

Identifier of illegal data type used in a sequence/property expression.

Language

Verilog

Rule Description

SystemVerilog does not allow several data types to be used in property and sequence expressions. These include non-integer types (shortreal, real and realtime), string, event, chandle, class, associative arrays and dynamic arrays.

Message Details

```
Identifier ( <identifier> ) of type '<type>' not allowed in a
sequence/property expression
```

Severity

Syntax

STX_VE_1392

Use of undeclared clocking block is not allowed.

Language

Verilog

Rule Description

SystemVerilog supports specifying default clocking block in a module, interface or program with a requirement that a clocking block with that name should be visible in that scope.

Message Details

```
No clocking block with name ( <name> ) declared in current
scope
```

Severity

Syntax

STX_VE_1393

Default clocking is already specified in the current scope.

Language

Verilog

Rule Description

SystemVerilog supports making a clocking block as default for a module, interface or program. But only one default clocking statement can appear in one scope.

Message Details

`Default clocking already specified in file (<file>), at line (<line-num>)`

Severity

Syntax

STX_VE_1394

Default input/output clocking skew already specified inside the clocking block.

Language

Verilog

Rule Description

SystemVerilog supports setting a skew as input/output default for a clocking block. However, only one default input skew and one output skew can be defined for a clocking block.

Message Details

```
Default <input | output> clocking skew already specified at
file (<file>), line (<line-num>)
```

Severity

Syntax

STX_VE_1395

Cyclic dependency of sequences is not allowed.

Language

Verilog

Rule Description

SystemVerilog does not allow cyclic dependency of sequences. For example, dependency of sequence S1 on S2 and that of S2 on S1 is not allowed simultaneously as it creates a cyclic dependency.

Message Details

```
Cyclic dependency of sequence ( <sequence> ) is not allowed
```

Severity

Syntax

STX_VE_1396

Use of negation operator 'not' with a recursive property is not allowed.

Language

Verilog

Rule Description

Negation operator 'not' can't be applied to any property that instantiates a recursive property, that is, the negation of a recursive property can't be asserted or used in defining another property. For example:

```
1. property p1; not (p1); endproperty

2. assert property (not p); //where p is a recursive property.
```

Message Details

```
Property expression containing instance of recursive property used with 'not' operator
```

Severity

Syntax

STX_VE_1397

Invalid method is used with sequence instance.

Language

Verilog

Rule Description

SystemVerilog allows methods (ended/matched/triggered) to be used with sequence instances using the '.' operator. Also, local variables inside sequence are not accessible outside the sequence by using the '.' operator. Therefore, trying to access any local variable outside the sequence will generate this error.

Message Details

```
Invalid method ( <method>1 ) used with sequence instance
```

Severity

Syntax

STX_VE_1398

Illegal object used where a sequence instance was expected.

Language

Verilog

Rule Description

SystemVerilog allows methods (ended/matched/triggered) to be used with sequence instances using the '.' operator. So, something like p1(c).ended results in this error, if p1 is a Property.

Message Details

```
( <object> ) is not a sequence instance
```

Severity

Syntax

Syntax Warning Rules

Rules of this category report a warning message for any syntax error in the code.

WRN_23

Avoid default values for signal parameters as they are ignored.

Language

VHDL

Rule Description

SpyGlass generates this language warning to indicate that it found default values assigned to SIGNAL parameters.

Default values cannot be assigned to SIGNAL parameters and they will never be used.

For example:

```
entity ent is
  port(in1,in2: bit_vector ( 0 to 1);
    output : out bit_vector ( 0 to 1));
end;

architecture arch of ent is
  procedure proc
    (signal in1: bit_vector;
     signal in2 : bit_vector := "11
-- WARNING:Cannot assign default value to
-- SIGNAL parameters as it will never be used.
    signal output : out bit_vector) is
    begin
      output <= in1 and in2;
    end;
begin
end;
```

Message Details

```
Cannot assign default value to SIGNAL parameters - will never
be used
```

Severity

LangWarning

Suggested Fix

Avoid default values for signal parameters as these values are ignored.

WRN_26

The macro has been redefined with a different value. If this is not really intended, it may produce unexpected synthesis results.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate a redefinition of the specified macro. Here, one of the definitions is incorrect. In such cases, SpyGlass proceeds with the last definition encountered. Some tools may even give a fatal error for such cases until the specified macro is defined only once.

Message Details

```
Redefinition of macro ( <macro> ). Previously macro was defined
at line ( <line-num> ) in file ( <file> ) with a value ( <value>
)
```

Severity

Warning

WRN_27

Bit-select should not be out-of-range.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate that it found bit-select index out of declared bounds.

Consider the following lines of code:

```
wire [0:10] w;
w[5]
w[15]
```

In the above example, reference to the bit 15 of wire *w* is out of range.

Message Details

`Bit-select (<bit-select>) is out-of-range`

Severity

Warning

WRN_VE_INACTIVE_27

Bit-select should not be out-of-range.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate that it found bit-select index out of declared bounds.

Consider the following lines of code:

```
wire [0:10] w;
w[5]
w[15]
```

In the above example, reference to the bit 15 of wire *w* is out of range.

NOTE: *The WRN_VE_INACTIVE_27 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Bit-select (<bit-select>) is out-of-range
```

Severity

Warning

WRN_28

Invalid use of part-select as left-hand-side of FORCE statement.

Language

Verilog

Rule Description

Part-select is not allowed as the left-hand-side of a force statement. In this case, the part-select on left-hand-side structurally matches the corresponding reg declaration. Hence this warning is flagged; else *STX_VE_367* is flagged.

Message Details

```
Invalid use of part-select as left-hand-side of FORCE statement
```

Severity

Warning

WRN_VE_INACTIVE_28

Invalid use of part-select as left-hand-side of FORCE statement.

Language

Verilog

Rule Description

Part-select is not allowed as the left-hand-side of a force statement. In this case, the part-select on left-hand-side structurally matches the corresponding reg declaration. Hence this warning is flagged; else *STX_VE_367* is flagged.

NOTE: *The WRN_VE_INACTIVE_28 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Invalid use of part-select as left-hand-side of FORCE statement
```

Severity

Warning

WRN_29

Module specified in a stopmodule list should be defined.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate that you have specified a module in the stopmodule list, but that module is not present in the HDL files provided in the SpyGlass run.

This warning may get generated in one of the following cases:

- The module specified in the stopmodule list has a typo
- The HDL files given to SpyGlass for analysis are incomplete, and hence some intended module has been missed.

Message Details

`Module ( <module> ) given in stopmodule list is not defined`

Severity

Warning

WRN_30

Timescale precision should be at least as precise as the time unit.

Language

Verilog

Message Details

```
Timescale precision ( <timescale-precision> ) should be at
least as precise as the unit ( <unit> )
```

Severity

Warning

WRN_31

Compiler directive can be ignored for further processing if a specific flag is set by the designer.

Language

Verilog

Message Details

```
Compiler directive ( <directive> ) will be ignored for further
processing
```

Severity

Warning

WRN_32

Only white space or a comment may appear on the same line as the `include compiler directive.

Language

Verilog

Rule description

Only white space or a comment may appear on the same line as the `include compiler directive. For example, SpyGlass does not report a warning in the following case:

```
`include "file1.v" // comment
```

However, SpyGlass reports a warning in the following case due to the occurrence of a semicolon on the same line as the `include compiler directive:

```
`include "file2.v" ;
```

Message Details

```
`include compiler directive not isolated on its own line
```

Severity

Warning

WRN_33

Module instance name not specified.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate that a module instance name is not specified in module instantiation.

For example:

```
module_name ();
```

Message Details

```
Module instance name not specified
```

Severity

Warning

WRN_VE_INACTIVE_33

Module instance name not specified.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate that a module instance name is not specified in module instantiation.

For example:

```
module_name ();
```

NOTE: *The WRN_VE_INACTIVE_33 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Module instance name not specified
```

Severity

Warning

WRN_35

Invalid library name specified in configuration block.

Description

The *WRN_35* rule reports a violation if the library specified within a configuration block has no logical to physical mapping specified in a project file.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

```
[WARNING] Invalid library name ( <lib-logical-name> )
```

Consequences of Not Fixing

The corresponding rule in the configuration, which has reference to that library name is ignored from SpyGlass analysis.

How to Debug and Fix

Either remove the library name usage or provide a logical name to the physical path library mapping for that library.

Default Severity Label

Syntax

WRN_VE_INACTIVE_35

Invalid library name specified in a configuration block.

Description

The *WRN_35* rule reports a violation if the library specified by a logical name does not exist in a configuration block.

NOTE: *The WRN_VE_INACTIVE_35 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[WARNING] `Invalid library name ( <lib-logical-name> )`

Consequences of Not Fixing

The logical name is not mapped to a library because of which SpyGlass results may vary.

How to Debug and Fix

Either remove the logical library name or provide a correct logical name that maps to an existing library name.

Default Severity Label

Syntax

WRN_36

Unable to decrypt data block in decryption envelope

When to Use

Use this rule to check the correctness of a Decryption Envelope.

Description

The *WRN_36* rule reports a violation if the data block in a Decryption Envelope cannot be decrypted due to some missing information in that envelope.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

```
[WARNING] Data block not decryptable, ignoring decryption protected envelope
```

Consequences of Not Fixing

The reported decryption envelope is ignored from SpyGlass analysis.

How To Debug and Fix

Check if all the required information is provided in the decryption envelope.

Default Severity Label

Warning

WRN_37

Integer/Time/Real type may not have a bit range specified.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate the use of a range of bits for a variable of type integer, range, real, and so on. For such variables, only the complete variable is meaningful. Bit/part selects of this variable do not have any significance. Hence, the bit/part select of a variable is ignored, and the complete variable is considered.

Message Details

```
Integer/Time/Real type may not have a bit-range specified.
Range specification ignored
```

Severity

Warning

WRN_38

Connection in instantiation should not be reg type because simulators do not support this.

Language

Verilog

Rule Description

Connection in instantiation should not be of type, `reg`. Such cases are not supported by simulators and IEEE 1364-1995/2001/2005 Verilog standards.

Message Details

```
Output connection ( <connection-name> ) in instantiation should
not be reg type as simulators do not support this. The IEEE
1364-1995/2001/2005 Verilog standards do not support it either
```

Severity

Warning

WRN_39

Combining ports of different IO types in concatenation is not permissible.

Language

Verilog

Rule Description

A port has been connected to several internal wires. However, those wires have different directions. Usually, it is very unlikely that you connect wires of different directions to the same port. Hence, you should check this connection, if the actual connection specified is really what was intended.

For example:

```
module top (.p1({a,b,c}), ..... )
input a,b ;
inout c ;
```

Here, `a`, `b` (inputs) are concatenated with `c` (inout), and all these three wires are together connected to port `p1`.

Message Details

```
Combining ports of different IO types in concatenation (
<concatenation> ) is not allowed.
```

Severity

Warning

WRN_40

This rule has been deprecated.

WRN_42

Declaration in $unit scope encountered during scan of library file. This can cause re-declaration errors on re-parsing of library files.

Language

Verilog

Description

This rule reports a violation if the declaration in the `$unit` scope is encountered during scan of a library file.

Such cases can cause re declaration errors on re parsing of library files. This is explained in the following example:

```
int global_var; // Warning for Lib Files but valid for
                // source files.
    module test;
    ...
endmodule
```

Message Details

```
Declaration in \$unit scope ( <scope> ) encountered during scan
of library file. This can cause re-declaration errors on
reparsing of library files.
```

Severity

Warning

WRN_43

Connection should be specified for UDP port.

Language

Verilog

Message Details

No connection is specified to UDP port (<port>)

Severity

Warning

WRN_44

Non-blocking assignment statements in function might not be supported by some simulators.

Language

Verilog

Rule Description

Non-blocking assignment statements in function might not be supported by some simulators. The IEEE 1364-2001/2005 Verilog standards do not support it either.

Message Details

```
Non-blocking assignment statements in function might not be
supported by some simulators. The IEEE 1364-2001/2005 Verilog
standards do not support it either.
```

Severity

Warning

WRN_45

Width should match between the terminal connection for gate instance and actual terminal connection.

Language

Verilog

Rule Description

An arrayed instantiation has been made. However, the bits connected to the terminals are not sufficiently wide to cater to all the instances of the array. For example, consider an instantiation given below:

```
tran A1[4]
```

The above instantiation means that there are 4 instances of `tran`. Hence, each terminal of `tran` should be 4 bit wide, each bit going into an individual `tran`.

If the terminals connected are lesser than 4 bits, then, some terminals of a few instances are considered as not connected.

Message Details

```
Width mismatch in terminal connection for ( <gate-type> ) gate instance. Expected width of connection ( <connected-expression> ): ( <num1> ) bits, Actual width: ( <num2> ) bits
```

Severity

Warning

WRN_46

Invalid port connection for inout port.

Rule Description

SpyGlass reports this warning message to indicate an invalid port connection for an inout port. In this case, the value used in port connection specification is not compatible. Consider the following example:

```
module top;
  S x1(.B(1'b1));
endmodule

module S(B);
  inout B;
endmodule
```

In the above example, the `B` port has a port connection with a constant value in the `top` module, which is invalid.

Message Details

`Invalid port connection for inout port ( <port> )`

Severity

Warning

WRN_47

Zero or negative repetition multiplier used in multiple concatenation.

Language

Verilog

Rule Description

Repetition multiplier used in multiple concatenation should have a positive value. For example:

```
a = {{0{2'b11}}};
```

Message Details

```
Improper repetition multiplier ( <multiplier> ) with evaluated
value ( <value> ) in concatenation
```

Severity

Warning

WRN_VE_INACTIVE_47

Zero or negative repetition multiplier used in multiple concatenation.

Language

Verilog

Rule Description

Repetition multiplier used in multiple concatenation should have a positive value. For example:

```
a = {{0{2'b11}}};
```

NOTE: *The WRN_VE_INACTIVE_47 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Improper repetition multiplier ( <multiplier> ) with evaluated value ( <value> ) in concatenation`

Severity

Warning

WRN_48

Re-declaration of a port with a name already used for some other port has been done.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate that the list of port declaration has the same name specified twice. SpyGlass considers this as one port and proceeds ahead. However, this is a strong indication of some typo. Either two distinct ports have been named similarly due to a typo, or the same port has been mistakenly specified twice.

You should correct the typo because of the following reasons:

- Some tools might not proceed without this being fixed
- If it is the first kind of mistake (two distinct ports having been named similarly) then the design intent will be captured incorrectly
- If it is the second kind of mistake then there could be an error if anyone tries to instantiate the module by specifying connections by position.

Message Details

```
Port with name ( <name> ) already exists
```

Severity

Warning

WRN_49

SystemVerilog assertion could not be translated

Language

Verilog

Rule Description

SpyGlass reports this warning if a SystemVerilog assertion was not translated due to any of the following reasons:

- The assertion has local variables, as shown in the following example:

```
module test;
wire a,b,c,d,e;
 sequence seq ;
    int z;
    a ##2 (b==z) [->2] ##0 c;
 endsequence
 assert property (seq);
endmodule
```

- The assertion has recursive property, as shown in the following example:

```
module test;
wire a,b,c;
  property prop ;
   @(posedge c)   a |=> prop;
  endproperty
  assert property (prop);
endmodule
```

- The assertion has illegal sequence instance usage, as shown in the following example:

```
module test;
wire a,b,c,d,e;
 sequence s2;
   a [*2] ##2 b [=3] ##2 s1(a,b).triggered [->4:10] ##1 c;
```

```
 endsequence
 assert property (s2);
endmodule
```

- The assertion has illegal dollar expression usage, as shown in the following example:

```
module test;
wire a,b,c,d,e;
  sequence seq ;
    @(posedge c)  $ ##2 $;
  endsequence
  assert property (seq);
endmodule
```

Message Details

```
SV assertion could not be translated (Reason: <reason>)
```

Severity

Warning

WRN_VE_INACTIVE_49

SystemVerilog assertion could not be translated

Language

Verilog

Rule Description

SpyGlass reports this warning if a SystemVerilog assertion was not translated due to any of the following reasons:

- The assertion has local variables.
- The assertion has recursive property.
- The assertion has illegal sequence instance usage.
- The assertion has illegal dollar expression usage.

NOTE: *The WRN_VE_INACTIVE_49 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`SV assertion could not be translated (Reason: <reason>)`

Severity

Warning

WRN_50

Net declaration must appear after port declaration

Language

Verilog

Rule Description

SpyGlass reports this warning if you declare a net without first declaring its direction.

Consider the following example:

```
module TOP(in1,in2,out);
 wire in1, in2;
 input  in1, in2;
 output out;
endmodule
```

In the above example, the WRN_50 rule reports a violation because the in1 and in2 nets are declared before declaring their direction.

Message Details

```
Port (<port-name>) is already declared in this scope as a net.
The nettype of a port must be declared after declaring its
direction
```

Severity

Warning

WRN_51

Some simulator might not support concatenation with un-sized number.

Language

Verilog

Message Details

Concatenation with un-sized number (<number>). Some simulators might not support this

Severity

Warning

WRN_52

Incorrect Uselib specification will be ignored.

Language

Verilog

Message Details

```
Uselib specification is incorrect, it will be ignored
```

Severity

Warning

WRN_53

Timing check limit should not be negative.

Language

Verilog

Message Details

```
Timing check limit cannot be negative
```

Severity

Warning

WRN_54

Overriding undefined parameter with null expression

Language

Verilog

Rule Description

Consider the following example:

```
module top;
  bot #(.p()) b1();  // Parameter 'p' is not defined
                     // in module bot
endmodule

module bot;
  parameter q = 1;
endmodule
```

In the above example, SpyGlass overrides the undefined parameter `p` with a null expression.

Message Details

```
Overriding undefined parameter <param-name> with null
expression
```

Severity

Warning

WRN_55

Unknown option is ignored.

Language

Verilog

NOTE: *This is an internal rule.*

Severity

Warning

WRN_56

Identifiers of size greater than 1024 might not be supported by other tools

Language

Verilog

Rule Description

Identifiers of size greater than 1024 might not be supported by other tools.

Severity

Warning

WRN_57

Vector reference in port list may not be supported by some simulators.

Language

Verilog

Message Details

```
Vector reference ( <reference> ) in port list may not be
supported by some simulators
```

Severity

Warning

WRN_58

Decimal constant too large. Numeric value exceeds 32-bit capacity

Language

Verilog

Rule description

Simple decimal numbers without the size and the base format shall be treated as signed integers, which have 32-bit capacity.

Message Details

`Numeric value ( <value> ) exceeds 32-bit capacity`

Severity

Warning

WRN_59

A system function cannot be specified when a system task is expected in the context.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate the usage of a system function when a system task was expected in this context. You should check if the code is in line with your expectations. The system function is effectively ignored since the output of the function is not assigned to anything. For example:

```
$urandom( 254 );
```

Here, if `$urandom` was intended to be used as a function then its value should have been assigned to something, as given in the following example:

```
I = $urandom( 254 ).
```

Message Details

```
System function ( <function> ) specified when a system task was expected in this context
```

Severity

Warning

WRN_60

A system task cannot be specified when a system function is expected in the context.

Language

Verilog

Message Details

```
System task ( <task> ) specified when a system function was
expected in this context
```

Severity

Warning

WRN_61

Parentheses should be used with reduction AND operator following bit-wise AND operator [a&(&b)].

Language

Verilog

Rule Description

A reduction AND operator follows a bit-wise AND operator in the Verilog source. This is often a typographical error, when a space is inserted between the two characters in '&&' (logical AND), resulting in '& &' (bit-wise AND followed by reduction AND). Hence brackets must be used to distinguish this scenario with reduction AND following a bit-wise AND.

Message Details

```
Use parentheses with reduction AND following bit-wise AND
```

Severity

Warning

WRN_62

Parentheses should be used with reduction OR operator following bit-wise OR operator [a|(|b)].

Language

Verilog

Rule Description

A reduction OR operator follows a bit-wise OR operator in the Verilog source. This is often a typographical error, when a space is inserted between the two characters in '||' (logical OR), resulting in '| |' (bit-wise OR followed by reduction OR). Hence brackets must be used to distinguish this scenario with reduction OR following a bit-wise OR.

Message Details

```
Use parentheses with reduction OR following bit-wise OR
```

Severity

Warning

WRN_63

Division by zero should be avoided because it may cause problem in simulation.

Language

Verilog

Rule Description

When a number is divided by zero, its simulation may result in `x` (unknown). In this situation, its synthesis result will depend upon the support provided by the tool and it may result in an undefined behavior.

Message Details

```
Division by zero in an expression ( <expression> ) may cause
undefined behavior
```

Severity

Warning

WRN_64

Part-select should not be out of range.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate that it found part-select index out of declared bounds.

Consider the following lines of code:

```
wire [0:10] w;
w[2:5]
w[5:15]
w[11:15]
```

In the above example, reference to the parts 5:15 and 11:15 of wire *w* are out of range.

Message Details

`Part-select (<part-select>) is out-of-range`

Severity

Warning

WRN_VE_INACTIVE_64

Part-select should not be out of range.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate that it found part-select index out of declared bounds.

Consider the following lines of code:

```
wire [0:10] w;
w[2:5]
w[5:15]
w[11:15]
```

In the above example, reference to the parts 5:15 and 11:15 of wire *w* are out of range.

NOTE: *The WRN_VE_INACTIVE_64 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Part-select (<part-select>) is out-of-range`

Severity

Warning

WRN_65

Compiler directive timescale is not allowed inside module definition boundary. Some simulators might issue an error for this.

Language

Verilog

Rule Description

Compiler directive, `timescale, is expected to be outside the module-endmodule region. Since SpyGlass need not interpret the delay values within the HDL code, therefore, SpyGlass ignores this directive and proceeds further. However, you should put this directive at the right place, else, some simulators might consider this as an error situation.

Message Details

```
Compiler directive `timescale not allowed inside module
definition boundary. Some simulators might issue an error for
this
```

Severity

Warning

WRN_66

Zero width specification of a based number is ignored.

Language

Verilog

Message Details

```
Zero width specification of based number ( <number> ) is
ignored, width is assumed to be <num> bits
```

Severity

Warning

WRN_68

Port declared with ANSI-style port declaration cannot be re-declared in the body.

Language

Verilog

Rule Description

Ports can be declared with ANSI-style port declaration or can be declared in the body. However, mixing the two types of port declarations is not allowed.

Consider the following lines of code:

```
module test (input aa, output bb);
  reg bb; // Warning
endmodule
```

Message Details

```
Multiple declarations for port '<port>' not allowed in <body>
with ANSI list of port declarations.
```

Severity

Warning

WRN_69

If a timescale directive is specified, it should be available to all the modules in the design.

Language

Verilog

Rule Description

The `timescale compiler directive is optional. Designs that do not contain this require no modification. However, if the design includes a `timescale compiler directive for any module definition, then all other modules definitions require this as well. Otherwise, simulators flag this as an error.

The `timescale can be declared in any file.

The smallest precision of all the `timescale directives determines the time unit of the simulation. For example:

```
`timescale 1 ns / 10 ps
module1 ();
...
endmodule

`timescale 100 ns / 1 ns
module2 ();
...
endmodule

`timescale 1 ps / 100 fs
module3 ();
...
endmodule
```

The first `timescale indicates that the time units for module1 are in multiples of 1 ns and it is precise to 10 ps. Thus, the smallest time step for the simulator is 10 ps.

The second `timescale is 100 ns / 1 ns. Since 1 ns is greater than 10 ps, the smallest time step remains 10 ps.

The third `timescale is 1 ps / 100 fs. Since 100 fs is smaller than 10 ps, the smallest simulator time step now becomes 100 fs.

Message Details

```
Compiler directive `timescale found for module '<module>' (in
the file <file> at line <line-num>) while previous modules do
not
```

Severity

Warning

WRN_70

IEEE has deprecated 'Standalone Generate Block'

Language

Verilog

Rule Description

SpyGlass reports this warning in SystemVerilog flow for each occurrence of a standalone `generate` block when you specify the following command in a project file:

```
set_option enableSV true
```

You must fix this violation as some tools may catch this as a fatal error.

The following example shows the violating and non-violating cases of this rule:

```
module test;
  parameter p = 1;
  generate
  if (p == 1)
   begin: BLK1    // Block is within generate-if
     ...
  end
  begin : BLK2    // Standalone Generate Block
  end
  endgenerate
endmodule
```

Message Details

```
Obsolete Verilog-2001 Construct 'Standalone Generate Block' is used
```

Severity

Warning

WRN_71

The repetition multiplier in a concatenation must be an integer.

Language

Verilog

Message Details

Repetition multiplier (<multiplier>) in the concatenation (<concatenation>) must be an integer

Severity

Warning

WRN_72

synthesis_off should match synthesis_on before the end of the file is encountered.

Language

Verilog

Message Details

```
End of file reached: unmatched synthesis_off
```

Severity

Warning

WRN_73

translate_off should match translate_on before the end of the file is encountered.

Language

Verilog

Message Details

```
End of file reached: unmatched translate_off
```

Severity

Warning

WRN_74

translate_on should not be specified without associated translate_off.

Language

Verilog

Message Details

```
translate_on specified without associated translate_off
```

Severity

Warning

WRN_75

Unrecognized pragma is treated as translate_off.

Language

Verilog

Message Details

Unrecognized pragma (<pragma>) encountered. Design compiler issues syntax error for this. Treating this as translate_off

Severity

Warning

WRN_84

Null range detected.

Language

VHDL

Rule Description

SpyGlass generates this language warning to indicate the presence of a null range.

A range is said to be a null range if the specified subset is empty. The range L to R is called an ascending range; if L>R, then the range is a null range. The range L down to R is called a descending range; if L<R, then the range is a null range.

For example:

(a)

```
entity EIS is
 type i is range 6 to 1;
--ERROR : type declaration is wrong because L > R for "to"
direction
    constant m:i:=3;
end EIS;
```

(b)

```
...
type boolean_v is array (integer range <>) of boolean;
subtype boolean_1 is boolean_v (1 to 1);
subtype boolean_null is boolean_v (1 to 0);
...
```

Suggested Fix

Do not use null range in your VHDL code.

Message Details

```
Range "<left-value-of-a-range> to <right-value-of-a-range>" is
null
```

Severity

LangWarning

WRN_111

No wait statement or sensitivity list in process - possible infinite loop.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate the absence of wait statement as well as sensitivity list in a process. If a process neither has sensitivity list nor wait statement inside it, then it may result in an infinite loop. However you cannot include both wait statement and sensitivity list inside same process statement as wait statement is just an alternate way to suspend the execution of a process.

Message Details

```
No wait statement or sensitivity list in process - possible
infinite loop
```

\Severity

Warning

Suggested Fix

Include either sensitivity list or wait statement inside process statement.

WRN_127

Enumeration literal must not be out of range (by context).

Language

VHDL

Rule Description

This warning is generated by SpyGlass to indicate the presence of out of range enumeration literal. The values of enumeration type are called enumeration literals. When subtypes and objects of these enumeration types are declared, a range constraint can also be specified in an object declaration. This range constraint must not exceed a set of user-defined values consisting of identifiers and character literals defined in an enumeration type declaration.

Message Details

```
Enumeration literal '<literal>' is out of range
```

Severity

LangWarning

WRN_128

Array index must not be out of range.

Language

VHDL

Rule Description

This warning is generated by SpyGlass to indicate array index out of range. An indexed name denotes an element of an array. An indexed name denotes an element of an array.

```
indexed_name::=prefix(expression{,expression})
```

For the evaluation of an indexed name, the prefix and the expressions are evaluated. It is an error if an index value does not belong to the range of the corresponding index range of the array.

Message Details

```
Array index <index> is out of range
```

Severity

LangWarning

Suggested Fix

Do not use array index out of bounds.

WRN_153

The range specified in the slice expression is out of range of the actual range of the prefix array in the corresponding slice expression

Language

VHDL

Rule Description

SpyGlass flags this language warning when the range specified in a slice expression is out of bound of the actual range of the prefix array of the slice expression.

Consider the following example:

```
type BIT_VECTOR is array (positive range <>)of BIT;
variable NUM1:BIT_VECTOR(1 to 10):=B"01_01_01_01_01";
NUM1(7 to 12):=B"010_010";
```

The WRN_153 rule reports violation for the above example since the range, `7 to 12`, for the expression, `NUM1(7 to 12)`, is out of the range, `1 to 10`, for the object, `NUM1`.

Message Details

```
The range "<expr-range>" for expression "<expr>" is out of
range "<actual-range>" of the object "<obj>"
```

Severity

LangWarning

WRN_170

Size of the target must match the size of the RHS expression.

Language

VHDL

Rule Description

This rule reports a warning message when SpyGlass finds a mismatch between sizes of the target and RHS expressions.

A warning is reported in case of signal, variable assignments, or assigning initial values to constants (using <= OR := assignment operators).

The warning is also reported when sizes of the RHS and LHS do not match or even if types of the RHS and LHS do not match.

The size of RHS is calculated for the whole expression so if RHS is a concatenation operation, size of RHS (bounds of concatenation) is compared with LHS and if they do not match warning is issued.

Such warnings can also be issued for locally static expressions.

For more clarity on the applicability of WRN_170 and WRN_531 rules, see the *WRN_531* rule description.

Message Details

```
The size of the target "<target>" does not match the size of
the RHS expression "<expression>"
```

Suggested Fix

Match the size of target with size of the RHS expression.

Severity

LangWarning

WRN_215

Size of RHS expression should match that of LHS expression.

Language

VHDL

Message Details

```
Size of rhs expression does not match that of lhs expression
```

Severity

LangWarning

WRN_220

Size mismatch detected.

Language

VHDL

Message Details

```
Size does not match
```

Severity

LangWarning

WRN_249

Function may return without executing a return statement.

Language

VHDL

Message Details

```
Function may return without executing a return statement
```

Severity

LangWarning

WRN_261

Physical path for logical library not given.

Language

VHDL

Rule Description

SpyGlass generates this warning when the physical location of a precompiled library dump could not be found.

The logical to physical mapping needs to be provided by using the `set_option lib <logical> <physical-path>` command in the project file and the physical path must exist.

Message Details

```
Missing logical to physical mapping for logical library
'<lib-name>'; Please provide it via set_option lib
<logical_library_name> <library_path>
```

Severity

LangWarning

WRN_265

Out of range value found (in context).

Language

VHDL

Message Details

`Value <value> is out of range <r1> to <r2>`

Severity

LangWarning

WRN_281

Object attached to port (in context) must be a signal (as per VHDL-87).

Language

VHDL

Rule Description

This warning occurs when SpyGlass does not find signal class for the object attached to port. The language requires that the object attached to port must be of signal class.

For example, the following code will produce this error:

```
    entity ent is
    end ent;
    architecture arch of ent is
        component comp
            port( in1: in boolean);
        end component;
        signal sig : boolean;
        for inst1 : comp use entity work.comp(arch)
            port map(true); -- ERROR Actual argument in port
map must be a signal
        begin
            inst : comp port map(true);
            inst1 : comp port map(true);
        end;
    configuration conf of ent is
        for arch
        for inst : comp use entity work.comp(arch)
            port map(true); -- ERROR Actual argument in port
map must be a signal
        end for;
    end for;
end;
```

Message Details

```
Object "<object>" attached to port "<port>" must be a signal
(VHDL-87)
```

Severity

LangWarning

WRN_283

Slice size of object should match constraint size (by context).

Language

VHDL

Message Details

```
Slice size of <size> of object "<object>" does not match
constraint size of <constraint-size>
```

Severity

LangWarning

WRN_384

Identifier used for Library or Package name must correspond to a valid Library or Package

Language

VHDL

Message Details

`<identifier> does not denote a library or package`

Severity

LangWarning

WRN_405

The specific directive has been ignored

Language

VHDL

Rule description

SpyGlass generates this language warning to indicate that the directive has been ignored. This occurs when a pragma block is left un-terminated by mistake. For example, SpyGlass flags a warning in the following case:

```
entity e is
port (a: in bit;
  b: out bit);
end e;

architecture rtl of e is
begin
  -- pragma translate_off
  -- CODE WITHIN PRAGMA BLOCK
  -- pragma translate_off
  -- OFF has come when, already OFF was applicable.
  -- Maybe, the code was supposed to have
  -- translate_on, rather than translate_off
  -- SOME CODE HERE
  -- This is within the scope of translate_off, but,
  -- may be, it was desired to be outside the scope of
  -- translate_off
  --pragma translate_on
end;
```

Message Details

```
Directive "<directive>" is ignored
```

Severity

LangWarning

WRN_406

Range constraint in alias declaration ignored

Language

VHDL

Message Details

```
Range constraint in alias declaration ignored
```

Severity

LangWarning

WRN_435

Unexpected Object Type

Language

VHDL

Rule Description

SpyGlass generates this warning to indicate an unexpected Object type in statement.

Message Details

```
Unexpected Object Type <type> in <statement>. Please contact
Atrenta Support. Please contact Atrenta Support.
```

Severity

INTERNAL_WARNING

WRN_443

No default binding for component found

Language

VHDL

Message Details

```
No default binding for component "<component>" found
```

Severity

LangWarning

WRN_471

Sub-element association for formal should be complete

Language

VHDL

Message Details

`Incomplete sub-element association for formal <formal>`

Severity

LangWarning

WRN_489

All statements within the force on-off directives must be preceded by the comment marker '--'

Language

VHDL

Message Details

```
All statements within the force on-off directives must be
preceded by the comment marker '--'
```

Severity

LangWarning

WRN_493

Re-declaration / redefining of design unit is not allowed

Language

VHDL

Message Details

```
Ignoring <design-unit-type> <design-unit-name> - already
encountered in File <file-name> Line <line-num>
```

Severity

LangWarning

WRN_499

Using 1076-1987 syntax for file declaration

Language

VHDL

Message Details

`Using 1076-1987 syntax for file declaration`

Severity

LangWarning

WRN_500

VHDL-93 syntax used; will not work in VHDL-87 environment

Language

VHDL

Message Details

```
Use of VHDL-93 syntax "<syntax>" will not work in VHDL-87
environment
```

Severity

LangWarning

WRN_501

VHDL-93 construct used; will not work in VHDL-87 environment

Language

VHDL

Message Details

```
VHDL-93 construct '<construct>' will not work in VHDL-87
environment
```

Severity

LangWarning

WRN_502

Variable declaration in this declarative region; will not work in VHDL-87 environment

Language

VHDL

Message Details

```
Variable declaration in this declarative region will not work
in VHDL-87 environment
```

Severity

LangWarning

WRN_503

Use of modes other than IN and OUT; will not work in VHDL-87 environment

Language

VHDL

Message Details

```
Use of modes other than IN and OUT will not work in VHDL-87
environment
```

Severity

LangWarning

WRN_504

Parameter should associate with actual to work in VHDL-87 environment

Language

VHDL

Message Details

```
Parameter <parameter> should associate with actual <actual> to
work in VHDL-87 environment
```

Severity

LangWarning

WRN_519

Cannot parse empty file

Language

VHDL

Message Details

`Cannot parse empty file.`

Severity

LangWarning

WRN_531

RHS expression size should match with the size of LHS expression

Language

VHDL

Rule Description

SpyGlass reports this message to indicate that the size of the RHS expression should match with the size of LHS expression.

A warning message is reported in case of signal, variable assignments, assigning initial values to constants (using <= OR := assignment operators), or in case of concatenation of expressions in any of these assignments.

A warning is also reported when an expression size does not match its expected size.

The rule matches the size between different expressions, irrespective of whether that expression if LHS, RHS or any intermediate expression forming LHS/RHS. Sometimes, this expression could be RHS and expected size could be LHS.

The size of RHS is calculated for each individual expression. So, if RHS of a statement is a concatenation operation, then while processing concatenation, for each pair of concatenation operation, sizes are compared and *WRN_531* is reported if they do not match. Such warnings can also be issued for locally static expressions.

For example, the SpyGlass flags the WRN_531 message in the following case:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity test is
  port(
    CLK: in std_logic;
    CLR: in std_logic;
```

```
    A1: in std_logic_vector(14 downto 0);
    A2: in std_logic
  );
end test;

architecture RTL of test is
  type array6_32 is array(5 downto 0) of
  std_logic_vector(13 downto 0);
  signal AOUT : array6_32;

begin
  AOUT <= AOUT(4 downto 0) & A1;
  -- Size (15) of RHS expression does not match with
  -- the size (14) of LHS expression
end RTL;
```

See the following scenarios for clarity between the applicability of *WRN_170* and WRN_531 rules:

Scenario 1:

```
Signal a :std_logic_vector(15 downto 0);
a (7 downto 0) <= x"ffff"; --WRN_531
```

WRN_531: Size 16 of expression X"ffff" does not match with the expected size 8.

Scenario 2:

```
SUBTYPE  subtype_t2 IS STD_LOGIC_VECTOR(7  DOWNTO 4);
CONSTANT C5   : STD_LOGIC_VECTOR (4 downto 3) :=  (4 downto 3
=> '0') ;
CONSTANT C3 : subtype_t2 := C5; --WRN_531
```

WRN_531: Size 2 of expression C5 does not match with the expected size 4.

Scenario 3:

```
CONSTANT xmas : xmasMemType := (
      STD_LOGIC_VECTOR(TO_SIGNED(512,12)),
      STD_LOGIC_VECTOR(TO_SIGNED(302,12)),
      STD_LOGIC_VECTOR(TO_SIGNED(0,12)));
  variable phase1     : std_logic_vector(7 downto 0);
  phase1 := work.xmasP.xmas(itr); --WRN_170 & WRN_531
```

WRN_170: The size of the target "8" does not match the size of the RHS expression "12".

Wrn_531: Size 12 of expression work.xmasP.xmas(itr) does not match with the expected size 8.

Scenario 4:

```
port( A1: in std_logic_vector(14 downto 0);
           A3: in std_logic);
  type array6_32 is array(5 downto 0) of std_logic_vector(13
downto 0);
  signal AOUT : array6_32;
  AOUT <= AOUT(4 downto 0) & (A1(11 downto 0) & A3); --
WRN_531
```

WRN_531: Size 13 of expression (A1(11 downto 0)& A3) does not match with the expected size 14

Scenario 5:

```
type array1 is array (0 downto 0) of bit_vector (0 to 0);
  signal out1 : array1;
  out1 <= x"1" & x"1"; --WRN_170 & WRN_531
```

WRN_170: The size of the target "1" does not match the size of the RHS expression "2"

WRN_531: Size 4 of expression X"1" does not match with the expected size 1

Scenario 6:

```
type write_pack is
      record
            valid : std_logic;
            operation : std_logic_vector (2 downto 0);
      end record;
  variable a_l : std_logic_vector (a'length - 1 downto 0);
  variable q : write_pack;
  q.operation := a_l (4 downto 0); --WRN_170
```

WRN_170: The size of the target "3" does not match the size of the RHS expression "5".

Message Details

```
Size <size> of expression <expression> does not match with the
expected size <size>
```

Severity

LangWarning

WRN_541

Design Unit should not be defined more than once

Language

VHDL

Message Details

```
<design-unit-type1> <design-unit-name> is defined multiple
times. Previous <design-unit-type2> declaration found in File:
"<file-name>" at Line no:'<line-number>'
```

Severity

LangWarning

WRN_542

Multiple design units with similar port/generic-interface ignored

Language

VHDL

Message Details

```
Ignoring ENTITY <entity-name1> because the port/generic-
interface of this ENTITY match to the port/generic-interface of
the ENTITY <entity-name2> already encountered in File:
"<file-name>" at Line no:'<line-num>'
```

Severity

LangWarning

WRN_547

Binding indication containing configuration as entity aspect is ignored for configuration specification during synthesis. Default binding is used for such cases during synthesis

Language

VHDL

Rule Description

Message Details

```
Binding indication containing configuration as entity aspect is
ignored for configuration specification during synthesis.
Default binding is used for such cases during synthesis
```

Severity

LangWarning

WRN_549

VHDL-93 does not allow use of OPEN as an actual for individual sub-element association

Language

VHDL

Message Details

VHDL-93 does not allow use of OPEN as an actual for individual sub-element association. Some simulators do not support this

Severity

LangWarning

WRN_553

Physical path corresponding to logical library does not exist

Language

VHDL

Message Details

```
Physical path (<path>) corresponding to logical library
(<library>) does not exist
```

Severity

LangWarning

WRN_554

Identifier must be declared before usage.

Language

VHDL

Message Details

`Use of undeclared identifier '<identifier>'`

Severity

LangWarning

WRN_563

Component instance is explicitly unbounded in the configuration

Language

VHDL

Message Details

```
Component instance (<instance>) is explicitly unbounded in file
'<file>' at line <line-num>
```

Severity

LangWarning

WRN_564

The actual associated with the variable parameter must be a variable

Language

VHDL

Message Details

`Actual associated with the variable parameter <parameter> must be a variable`

Severity

LangWarning

WRN_568

If design units with same name exist, proper configuration should be provided when they are used.

Language

VHDL

Message Details

```
Design unit '<design-unit>' is present in multiple precompiled
libraries <libraries>
```

Severity

LangWarning

WRN_600

Replacement character used should be valid (in context)

Language

VHDL

Message Details

```
Replacement character '<character1>' is used instead of
'<character2>' in <based literal | string literal | choices>
```

Severity

LangWarning

WRN_601

PSL constructs should appear only inside entity or architecture bodies

Language

VHDL

Message Details

```
Ignoring PSL construct as it is outside entity or architecture
body
```

Severity

LangWarning

WRN_602

PSL constructs should not appear inside process blocks or subprogram bodies

Language

VHDL

Message Details

```
Ignoring PSL construct as it is inside a <block | subprogram>
```

Severity

LangWarning

WRN_606

String corresponding to ENUM_ENCODING attribute can only contain 0/1/Z/D/U

Language

VHDL

Rule Description

The error message is reported by the tool to indicate that at the specified location string literal specified for ENUM_ENCODING is non-binary. Only binary numbers are supported in the string literals corresponding to the ENUM_ENCODING attribute of the enumerations.

Message Details

```
Literal may contain only '0', '1', 'Z', 'D' or 'U', ignoring
the declaration
```

Severity

LangWarning

WRN_609

Valid binding must exist for each instance used during elaboration

Language

VHDL

Rule Description

This elaboration error message is generated by SpyGlass for the case of default binding of instance, when the entity does not have default architecture.

Message Details

```
No binding exists for instance <instance-name(label)> during
elaboration of <inst-hierarchical-name><binding-message>
```

Severity

LangWarning

WRN_610

Out of range values between array size and index passed to the same are found

Language

VHDL

Rule Description

SpyGlass flags this warning if it detects out of range value for index passed to arrays, as shown in the following example:

```
entity t003 is
  port (
    idx     : in  integer range 0 to 15;
    b       : in  bit;
    iarray1 : in  bit_vector(17 downto 0);
    iarray2 : in  bit_vector(32 downto 17);
    array1  : out bit_vector(17 downto 0);
    array2  : out bit_vector(32 downto 17)
  );
end;
architecture t003 of t003 is --
  signal a1 : bit;
  signal a2 : bit;
  begin
  process (idx , b)
  begin
    array1(idx) <= b; --ranges specified are overlapping
    a1 <= iarray1(idx); --ranges specified are overlapping
  end process;
  process (idx , b)
  begin
    array2(idx) <= b; --ranges specified are disjoint
    a2 <= iarray2(idx); --ranges specified are disjoint
  end process;
end;
```

Message Details

```
Possible out of bounds condition as array index range
(<index-range>) does not match array range (<array-range>)
```

Severity

LangWarning

Suggested Fix

Specify the range of index and the size of array as overlapping, and do not specify disjoint set of values.

WRN_612

Source file is newer than it's SpyGlass precompiled (library) dump this may lead to incorrect elaboration errors, please recompile the specified file in the same library.

Language

VHDL

Rule Description

The source file is newer than its SpyGlass Precompiled (library) dump. This may lead to incorrect elaboration errors. Thus, you need to recompile the specified file in the same library.

NOTE: *The WRN_612 rule is switched off by default.*

Message Details

```
Source file "<file>" is newer than it's pre-compiled (library)
dump, please recompile this file; otherwise it may lead to
incorrect elaboration errors
```

Severity

LangWarning

WRN_613

Dependent Package CheckSum Error: Different version of a package used in the current session.

Language

VHDL

Rule Description

This dependent package checksum error is generated by SpyGlass to indicate that a design unit used a different package when it was compiled earlier and in the current run it refers a different package with the same name.

For example:

There are two versions of a file package.vhd compiled into logical library lib1 at two different physical locations.

Step1

Compile package.vhd (version1) into logical library lib1 at physical path ./dir1

```
package pack is
  CONSTANT VAL0 : NATURAL := 0;
        end;


set_option work lib1
set_option lib lib1 ./dir1
set_option project wdir run1
read_file -type vhdl package.vhd
```

Step2

Compile package.vhd (version 2) into logical library lib1 at physical path ./dir2

```
package pack is
  CONSTANT VAL1 : NATURAL := 1;
  CONSTANT VAL0 : NATURAL := 0;
end;
```

```
set_option work lib1
set_option lib lib1 ./dir2
set_option project wdir run2
read_file -type vhdl package.vhd
```

Step3

Compile middle.vhd in library libm at physical path ./dirm;

Use: lib1 from ./dir1 via `set_option lib lib1 ./dir1`

```
library lib1;
 use lib1.pack.all;
 entity ent is
 port(a : in  bit_vector (VAL0 downto 0));
 begin
 end ent;

architecture rtl of ent is
begin
end;

set_option -work libm
set_option lib libm ./dirm
set_option projectwdir run3
read_file -type vhdl middle.vhd
set_option lib lib1 ./dir1
```

Step4

Elaborate entity `ent` using `set_option top` <*top-name*>;

Use `set_option lib lib1 ./dir2` and `set_option lib libm ./dirm`

```
set_option top ent
set_option lib lib1 ./dir2
set_option lib libm ./dirm
```

```
set_option project wdir run4
```

In the above scenario, because two different versions of the same library lib1 have been used (versions 1 and 2 above), the external references of libm in the SpyGlass dumps were created using version 1 and not version 2 and hence the CheckSum error is flagged.

To remove this warning, provide the correct library mappings by using the `set_option lib <logical-lib-name> <physical-path>` command in the project file.

Message Details

```
The package '<package-name>' in the library '<library-name>' is
not the same when it was used earlier while compiling
<design-unit-type> '<design-unit-name>'
```

Severity

LangWarning

WRN_623

Choice expression should be locally static.

Language

VHDL

Rule Description

SpyGlass flags this warning to specify that choice expression should be locally static but when actually the choice expression is globally static and evaluable. A locally static expression is an expression that can be evaluated during the analysis of the design unit in which it appears. The language requires that selector expression of a case statement must result in a value of a discrete type or a one-dimensional array of character elements, such as a character string or bit string, and it must be locally static.

For example, SpyGlass flags the WRN_623 warning in the following case:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_unsigned.ALL;

entity myentity is
  port (
    inputA : in std_logic_vector(3 DOWNTO 0);
    inputB : in std_logic_vector(3 DOWNTO 0);
    inputSEL : in std_logic_vector(1 DOWNTO 0);
    outputZ : out std_logic_vector(3 DOWNTO 0)
  );
end entity myentity;

architecture myarch of myentity is
  CONSTANT ROT_0      : std_logic_vector(1 DOWNTO 0) :=
                        conv_std_logic_vector(0, 2);
begin
  myproc:
  process (inputSEL)
  begin
```

```
    case inputSEL is
      when ROT_0 =>  outputZ <= inputA;
      -- ROT_0 is statically evaluable
      when others => outputZ <= inputB;
    end case;
  end process myproc;
end myarch;
```

Message Details

```
Choice expression should be locally static
```

Severity

```
LangWarning
```

WRN_624

De-compilation of encrypted RTL not allowed

Language

VHDL

Rule Description

Encrypted RTL is not allowed to be shared. Therefore, de-compilation is not allowed.

Message Details

```
De-compilation of encrypted RTL not allowed
```

Severity

LangWarning

WRN_637

Unable to decrypt data block in decryption envelope

When to Use

Use this rule to check the correctness of a Decryption Envelope.

Description

The *WRN_637* rule reports a violation if the data block in a Decryption Envelope cannot be decrypted.

For example, a data block is not decrypted due to some missing information, such as a missing key block.

Language

VHDL

Messages and Suggested Fix

This rule reports the following message:

[LangWarning] `Data block not decryptable, ignoring decryption protected envelope. Please contact the vendor of this file for resolution of this problem.`

Consequences of Not Fixing

The reported decryption envelope is ignored from SpyGlass analysis.

How To Debug and Fix

Check if all the required information is provided in the decryption envelope.

Example Code and/or Schematic

Not applicable

Default Severity Label

LangWarning

WRN_639

value of option allow_recursion_limit is ignored.

Language

VHDL

Rule Description

The *WRN_639* rule reports a violation if the value specified to the `allow_recursion_limit` command is not in allowed range of 1 to 2147483647.

For example, this rule reports a violation in the following case:

```
set_option allow_recursion_limit -10
set_option allow_recursion_limit 2147483649
```

Message Details

```
Value of allow_recursion_limit is ignored as it is outside
permissible range of integer value.
```

Severity

LangWarning

WRN_645

VHDL syntactic or semantic error / warnings detected in the encrypted source file

Language

VHDL

Rule Description

The *WRN_646* rule reports a violation if VHDL syntactic or semantic error/ warnings are detected in the encrypted source file. The rule suppresses the actual message for security reasons.

Message Details

```
VHDL syntactic or semantic error/warnings detected in the
encrypted source file. The actual error message has been
suppressed for security reasons. Please contact the vendor of
this file for resolution of this problem
```

Consequences of Not Fixing

If you do not fix this violation, the SpyGlass run does not proceed further.

How To Debug and Fix

Contact the vendor of the encrypted file for resolving this violation.

Severity

LangWarning

WRN_822

The specified API will be deprecated in future release

Language

Verilog

Rule Description

The specified API will be deprecated in future release.

Message Details

```
API '<API-name>' will be deprecated in future release. Please
contact Atrenta Support.
```

Severity

INTERNAL_WARNING

WRN_901

The argument passed to a function must of the specified type.

Language

Verilog

Message Details

The argument passed (type : <type>) to the function (<function>) is not of expected type. Please contact Atrenta Support.

Severity

INTERNAL_WARNING

Note

If an array bit-select is used in the sensitivity list of an *always* construct, SpyGlass reports the *WRN_901* warnings.

For example, SpyGlass reports such warnings if the following *always* construct is used:

```
module test(in1,clk,out1);
  input [3:0] in1;
  input [3:0][3:0] clk [3:0];
  output reg [3:0] out1 [1:0];
  always@(posedge clk[0][0][0])
    out1[0]<=in1;
endmodule
```

In such cases, ignore or waive such warnings.

WRN_1021

Array index is out-of-bounds.

Language

Verilog

Rule Description

SpyGlass generates this language warning to indicate that an array index is out-of-bounds. It means that the index value is out of permissible range.

Message Details

```
Array index ( <index> ) out-of-bounds, index value ( <value> )
is out of permissible range ([<r1>:<r2>])
```

Severity

Warning

WRN_1022

Encoding is the wrong size for this enumerated type.

Language

Verilog

Message Details

```
Encoding '<parameter-type-identifier>' for
'<assigned-value-expression-to-parameter>' is the wrong size
for this enumerated type
```

Severity

Warning

WRN_1023

enum directive requires parameter to have size specified.

Language

Verilog

Message Details

```
enum directive requires parameter to have size specified
```

Severity

Warning

WRN_1024

Signed argument is passed to $signed system function call, or unsigned argument passed to $unsigned system function call.

Language

Verilog

Rule Description

`$signed` and `$unsigned` system function calls convert the argument expression to signed and unsigned values, respectively. SpyGlass flags this warning in the following cases:

- When a signed expression is passed to the `$signed` system function call
- When an unsigned expression is passed to the `$unsigned` system function call

For example:

```
module test;
  parameter signed p1 = -4;
  parameter p2 = $signed (p1);
  // Converting signed to signed
 parameter p3 = $unsigned (p1);
 // Converting signed to unsigned
endmodule
```

To fix the warning flagged in the above declaration of the `p2` parameter, replace that declaration with the following:

```
parameter p2 = p1;
```

Message Details

```
<signed | unsigned> argument '<argument-name>' passed to
<unsigned | signed> system function call
```

Severity

Warning

WRN_VE_INACTIVE_1024

Signed argument is passed to $signed system function call, or unsigned argument passed to $unsigned system function call.

Language

Verilog

Rule Description

`$signed` and `$unsigned` system function calls convert the argument expression to signed and unsigned values, respectively. SpyGlass flags this warning in the following cases:

- When a signed expression is passed to the `$signed` system function call
- When an unsigned expression is passed to the `$unsigned` system function call

For example:

```
module test;
  parameter signed p1 = -4;
  parameter p2 = $signed (p1);
  // Converting signed to signed
 parameter p3 = $unsigned (p1);
 // Converting signed to unsigned
endmodule
```

To fix the warning flagged in the above declaration of the `p2` parameter, replace that declaration with the following:

```
parameter p2 = p1;
```

NOTE: *The WRN_VE_INACTIVE_1024 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
<signed | unsigned> argument '<argument-name>' passed to
<unsigned | signed> system function call
```

Severity

Warning

WRN_1026

Incorrect usage of @*/@(*) token

Language

Verilog

Message Details

`Incorrect usage of @*/@(*) token`

Severity

Warning

WRN_1027

Improper second argument in line compiler directive.

Language

Verilog

Message Details

```
Improper second argument in line compiler directive
```

Severity

Warning

WRN_1028

There is a name conflict. The name has been previously defined at some other location.

Language

Verilog

Message Details

```
There is a name conflict for ( <name> ).The name has been
previously defined at file ( <file-name> ), line ( <line-num> )
```

Severity

Warning

WRN_1029

Nested interface name is same as parent interface name.

Language

Verilog

Rule Description

This rule reports a violation if a nested interface name collides with the parent interface name. It is possible to instantiate the parent interface inside the nested one. However, if both of them have the same name, then inside the nested interface, the nested interface name shadows the parent name. That is, only the nested interface can be instantiated within itself.

Message Details

```
Nested interface name (<name>) is same as parent <parent> name
( <pname> )
```

Severity

Warning

WRN_1030

The specified library does not exist.

Language

Verilog

Rule Description

The logical library name specified in the `uselib directive does not exist.

Message Details

```
Library '<library-name>' does not exist
```

Severity

Warning

WRN_1031

There are more than one module definitions in between a single `celldefine and `endcelldefine

Language

Verilog

Rule Description

There are more than one module definitions in between a single `celldefine and `endcelldefine

Message Details

`More than one (<module-count>) module definitions found between `celldefine and `endcelldefine`

Severity

Warning

WRN_1032

There is a `celldefine in the source before a `endcelldefine to close the previous `celldefine

Language

Verilog

Rule Description

There is a `celldefine in the source before a `endcelldefine to close the previous `celldefine

Message Details

```
Nested `celldefine directive found
```

Severity

Warning

WRN_1033

A selected bit is out of range in the hierarchical identifier.

Language

Verilog

Rule Description

If bit select is used in the hierarchical reference to an element, warning is issued if the selected bit is out of range. The hierarchical value inferred is unknown for such a case.

For example,

```
struct { bit [3:0] b;} [2:0] bb [3:0];

assign o=bb[5][2].b[1]; //bb[5] is out of range
```

However, it would be an error if the number of dimensions of the hierarchical identifier is incorrect.

Message Details

```
Out-of-range bit selection in hierarchical identifier
'<identifier>'
```

Severity

Warning

WRN_1034

Modules containing unsupported SystemVerilog constructs are not compiled into the work library.

Language

Verilog

Rule Description

SpyGlass reports this warning if a module contains an unsupported SystemVerilog construct.

If you do not fix this warning, the following consequences occur:

- The module is not compiled by SpyGlass because it is not complete.
- The module is not elaborated.
- The violations reported by the *ErrorAnalyzeBBox* rule in further SpyGlass runs can be different.

Message Details

```
Module ( <module-name> ) is not compiled into a library due to
unsupported SystemVerilog construct ( <construct-name> ) in
file ( <file-name> ) at line ( <line-number> )
```

Severity

Warning

WRN_1035

Array size is too large, some simulators might not support this.

Language

Verilog

Message Details

```
Array size is too large, some simulators might not support this
```

Severity

Warning

WRN_1036

Expecting explicit event control immediately after always_ff.

Language

Verilog

Rule Description

This warning is issued if event control is not present immediately after always_ff. For example,

```
always_ff
begin
 @(posedge in2) //Warning
  out2 = ~in3;
end
```

However, it would be an error to give multiple event control statements.

Message Details

```
Expecting explicit event control immediately after always_ff
```

Severity

Warning

WRN_1037

End of source encountered inside a paired compiler directive, e.g., between `celldefine and `endcelldefine

Language

Verilog

Rule Description

SpyGlass generates this language warning when the end of source is encountered before the closing compiler directive corresponding to its opening compiler directive. This may not be a problem if the source code following the opening compiler directive, till the end of source, was intended to be included inside the compiler directive. However, if this is not the case, SpyGlass generates unintended results. For example, a missing `endcelldefine corresponding to a `celldefine may unintentionally make some modules as cells.

Message Details

```
End of source encountered inside <compiler-directive> directive
(in file <file-name> at line <line-num>)
```

Severity

Warning

WRN_1038

`celldefine -`endcelldefine spanning across multiple files encountered

When To Use

Use this rule to detect `celldefine -`endcelldefine spanning across multiple files.

Description

The *WRN_1038* rule reports a violation if `celldefine -`endcelldefine spans across multiple files.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[WARNING] `celldefine - `endcelldefine spanning across multiple files encountered (`celldefine found in file <file1> at line <line1>, `endcelldefine found in file <file2> at line <line2>)

Consequences of Not Fixing

If `celldefine region covers the top-level module, SpyGlass fails to detect the top-level module.

How To Debug and Fix

Recheck the design to see if `celldefine -`endcelldefine are intentionally kept across multiple files.

If this is intentional then ignore this violation. Else, keep the `celldefine -`endcelldefine block in the same file.

Example Code and/or Schematic

This rule reports a violation in the following case because:

```
--- test1.v-------
`celldefine
module mid(input in, ...);
...
endmodule
-----------------
--test2.v-----
module top(input in1, ..);
mid m1 (...);
endmodule
`endcelldefine
-----------
```

Default Severity Label

Warning

WRN_1039

It is highly recommended to have the entire ifdef/ifndef region in the same file

Language

Verilog

Rule Description

SpyGlass generates this warning if `` `ifdef/`ifndef `` in a file does not have a matching `endif` in the same file.

Message Details

```
The `ifdef/`ifndef specified in file ( <file-name> ) at line (
<line-num> ) does not have the matching `endif in the same file
```

Severity

Warning

WRN_1040

Improper usage of statement label.

Language

Verilog

Rule Description

Improper usage of label. Such usage is allowed only in SystemVerilog with assert, assume, and cover directives.

Message Details

```
Improper usage of label ( <label> ). Such usage is allowed only
in SystemVerilog with assert, assume and cover directives
```

Severity

Warning

WRN_1041

Underscore (_) present at the end of a numeric value will be ignored.

Language

Verilog

Rule Description

SpyGlass reports this warning when an underscore is placed at the end of numeric values.

For example, this warning is reported if you specify `4'b1110_`. However, it is not reported if you specify `4'b11_01`.

Message Details

```
Underscore (_) present in the end of a numeric value will be ignored
```

Severity

Warning

WRN_1042

Identifier not declared for implicit port mapping using .*,an empty connection would be created for this port.

Language

Verilog

Rule Description

SpyGlass reports this warning when an identifier is not declared for an implicit port mapping by using `.*`.

For such ports, an empty connection is created.

For example, this rule reports a violation in the following case:

```
module low(input a, output b1);
  ...
endmodule
module top(input a, output b);
  low l1(.*);
  ...
endmodule
```

In the above example, `.*` is expanded to `(.a(a), .b1())`. In this case, the `b1` port is left unconnected.

Message Details

```
Identifier ( <identifier> ) of implicit .* port connection not declared, an empty connection would be created for this port
```

Severity

Warning

WRN_VE_INACTIVE_1042

Identifier not declared for implicit port mapping using .*,an empty connection would be created for this port.

Language

Verilog

Rule Description

SpyGlass reports this warning when an identifier is not declared for an implicit port mapping by using `.*`.

For such ports, an empty connection is created.

For example, this rule reports a violation in the following case:

```
module low(input a, output b1);
  ...
endmodule
module top(input a, output b);
  low l1(.*);
  ...
endmodule
```

In the above example, `.*` is expanded to `(.a(a), .b1())`. In this case, the `b1` port is left unconnected.

NOTE: *The WRN_VE_INACTIVE_1042 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Identifier ( <identifier> ) of implicit .* port connection not declared, an empty connection would be created for this port
```

Severity

Warning

WRN_1043

Existing precompile dump of a package is being replaced.

Language

Verilog

Description

The *WRN_1043* rule reports a violation when an existing precompile output of a package is being overwritten, which may cause problem in further restoration runs.

Message Details

```
Overriding precompile dump of package '<package>' in library
'<lib>' (<path>). This may result in STX_VE_415 during further
restoration runs
```

Severity

Warning

Example

Example 1

Consider the following example:

```
//test1.v
module test1;
  import pack::*;
endmodule

//test2.v
module test2;
  import pack::*;
endmodule

//pack.v
Package pack;
Endpackage
```

```
set_option libhdlfiles L1 {pack.v test1.v}
set_option libhdlfiles L1 {pack.v test2.v}
```

In the above example, the `pack` package is getting compiled in `L1` twice. This results in the *WRN_1043* violation.

Example 2

Consider the following example:

```
set_option libhdlfiles L1 {pack.v test1.v}
set_option libhdlfiles L2 {pack.v test2.v}
set_option lib L1 precompile_lib
set_option lib L2 precompile_lib
```

In the above example, the name of libraries is different, that is, `L1` and `L2`, but their actual path is the same.

WRN_1044

Existing precompile dump of a design unit is being replaced.

Language

Verilog

Description

SpyGlass reports a violation message when an existing precompile output is being overwritten, which may cause problem in further restoration runs.

Message Details

`Overriding an existing precompile dump can cause problem in further restoration runs and may result in STX_VE_415`

Severity

Warning

Example

Example 1

Consider the following example:

```
//test1.v

module test1;
buf i1();
endmodule


//test2.v

module test2;
buf i2();
endmodule


//buf.v
module buf;
……
```

```
Endmodule


set_option libhdlfiles L1 {test1.v buf.v}
set_option libhdlfiles L1 {test2.v buf.v}
```

In the above example, module, `buf` is getting compiled in libhdlfile, `L1` twice, which causes the WRN_1044 rule to report a violation.

Example 2

Consider the following example:

```
set_option libhdlfiles L1 {test1.v buf.v}
set_option libhdlfiles L2 {test2.v buf.v}
set_option lib L1 precompile_lib
set_option lib L2 precompile_lib
```

In the above example, the name of libhdlfiles is different, that is, L1 and L2, but the actual path of these libhdlfiles is the same.

WRN_1045

VECTORED keyword is not allowed for trireg declaration.

Language

Verilog

Message Details

`VECTORED keyword not allowed for trireg declaration`

Severity

Warning

WRN_1046

Assert final is not a standard as per IEEE 1800-2009.

Language

Verilog

Rule Description

The WRN_1046 rule reports a violation for the assert final construct.

This construct is not the standard IEEE 1800-2009 construct. Such constructs are considered as deferred assertions.

For example, consider the following construct:

```
a1: assert final (out1 == ~in1)
    $fatal (2,"System Task FATAL executed");
```

The above construct is considered as the following:

```
a1: assert #0 (out1 == ~in1)
    $fatal (2,"System Task FATAL executed");
```

Message Details

```
The 'final' syntax is non-standard and will be treated as '#0'
```

Severity

Warning

WRN_1047

The 'assignment pattern without apostrophe' syntax is non-standard

Language

Verilog

Rule Description

The WRN_1047 rule reports a violation if an assignment pattern is not prefixed with an apostrophe.

The IEEE 1800 standard requires assignment patterns to be prefixed with an apostrophe.

The following example shows the violating and non violating cases of this rule:

```
int a[4] = {default:0};  // Warning
int a[4] = '{default:0}; // Fine
```

Message Details

```
The 'assignment pattern without apostrophe' syntax is
non-standard
```

Severity

Warning

WRN_1048

Redeclaration of genvar is not allowed

Language

Verilog

Rule Description

The WRN_1048 rule reports a violation for redeclaration of a genvar variable, as shown in the following example:

```
genvar i;
genvar i;    // Warning
```

As per the IEEE standard, redeclaration of genvar variables is not allowed.

Message Details

```
Re-declaration of genvar (<variable-name>) is non-standard,
previous declaration in file ( <file-name> ), at line ( <line-
num> )
```

Severity

Warning

WRN_1049

An implicit .* port connection cannot be used to refer a generic interface port.

Rule Description

An implicit `.*` port connection cannot be used to refer a generic interface port. For example, SpyGlass flags the WRN_1049 message in the following case:

```
module mid( interface i1);
endmodule

module top ();
  intf i1();
  mid m1(.*);
endmodule
```

Language

Verilog

Message Details

```
An implicit .* port connection cannot be used to refer a
generic interface port ( <port> )
```

Severity

Syntax

WRN_1050

Insufficient arguments are specified for the function call.

Language

Verilog

Message Details

`Too few arguments for function call (<function-call>)`

Severity

Warning

WRN_1051

Too many arguments are specified for the function call.

Language

Verilog

Rule Description

Message Details

`Too many arguments for function call (<function-call>)`

Severity

Warning

WRN_1052

A function should have at least one input argument.

Language

Verilog

Rule Description

A function should have at least one input argument. In SystemVerilog, however, functions may not have any arguments. Hence, this message shall not be flagged in SystemVerilog mode.

Message Details

```
Function '<function>' should have at least one input argument
```

Severity

Warning

WRN_1053

'x' or 'z' used in vector index.

Language

Verilog

Message Details

```
'x' or 'z' used in vector index.
```

Severity

Warning

WRN_1054

Out of bounds level code. Level code should either be 0, 1, or 2.

Language

Verilog

Message Details

```
Out of bounds level code. Level code should be (0/1/2)
```

Severity

Warning

WRN_1055

Too many arguments are specified to call the task.

Language

Verilog

Message Details

`Too many arguments to call task (<task>)`

Severity

Warning

WRN_1056

Incorrect hierarchical instance name in the instance clause of a configuration.

Description

The *WRN_1056* rule reports a violation if the hierarchical instance name in an instance clause does not begin with the name of the top-level cell mentioned in the design statement of the configuration.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

```
[WARNING] Illegal hierarchical instance name ( <name>) in an instance clause
```

Consequences of Not Fixing

The reported instance is ignored from SpyGlass analysis.

How to Debug and Fix

Specify the instance whose name matches with the name of the top-level cell mentioned in the design statement of the configuration

Example Code and/or Schematic

Consider the following example:

```
config cfg;
  design top;
  instance top.I1 liblist L1 L2;     // Fine
  instance xyz.I1 liblist L1 L2;       // Warning
endconfig
```

Default Severity Label

Warning

WRN_1057

Insufficient arguments are specified to call task.

Language

Verilog

Message Details

`Too few arguments to call task (<task>)`

Severity

Warning

WRN_1059

Assignment to a variable of data type enum, unpacked structure, or unpacked union must be of same type

Language

Verilog

Rule Description

The WRN_1059 rule reports a violation when the assignment to a variable of data type enum, unpacked structure, or unpacked union is not of the same type.

Two types having same RTL definition string but defined at different locations in a design are considered as different types.

To consider them of the same type, specify the following project-file command:

```
set_option allow_non_lrm yes
```

Message Details

```
Incompatible assignment to enum/structure/union in <port/
argument> (<port-name>/<argument-name>) of <module/interface/
function> (<module-name>/<interface-name>/<function-name>)
```

Severity

Warning

WRN_1060

Duplicate instance/cell clause found

Description

The *WRN_1060* rule reports a violation if a duplicate instance/cell clause with the same hierarchical/cell name is specified.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

```
[WARNING] Duplicate <instance> clause ( cell-clause> ) in the configuration. Previously mentioned in file ( <file-name> ) at line ( <line-num> )
```

Consequences of Not Fixing

Only the first found instance/cell clause is considered for SpyGlass analysis.

How to Debug and Fix

Remove the duplicate instance/cell clause.

Example Code and/or Schematic

The following example shows the violating scenarios of this rule:

```
config cfg;
  design top;
  instance top.I1 use L1.mid;
  instance top.I1 liblist L2 L3;      // Warning
  cell mid liblist L1;
  cell mid liblist L2;                // Warning
endconfig
```

Default Severity Label

Warning

WRN_VE_INACTIVE_1059

Assignment to a variable of data type enum, unpacked structure, or unpacked union must be of same type

Language

Verilog

Rule Description

The WRN_1059 rule reports a violation when the assignment to a variable of data type enum, unpacked structure, or unpacked union is not of the same type.

Two types having same RTL definition string but defined at different locations in a design are considered as different types.

To consider them of the same type, specify the following project-file command:

```
set_option allow_non_lrm yes
```

NOTE: *The WRN_VE_INACTIVE_1059 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Incompatible assignment to enum/structure/union in <port/
argument> (<port-name>/<argument-name>) of <module/interface/
function> (<module-name>/<interface-name>/<function-name>)
```

Severity

Warning

WRN_1451

Outputs for edge-sensitive paths have to be associated with data source.

Language

Verilog

Message Details

```
Outputs for edge-sensitive paths have to be associated with
data source
```

Severity

Warning

WRN_1452

Inconsistent range is found in the IO declaration for port.

Language

Verilog

Message Details

`Inconsistent range in IO declaration for port ( <port> )`

Severity

Warning

WRN_1453

Output/inout port specification for an instance must be valid as per the port definition.

Language

Verilog

Message Details

`Invalid output port specification ( <portnet-expression> ) for instance ( <inst-name> ), <type> ( <connected-expression> ) expression not permissible`

Where, *<type>* can be `unary`, `binary`, `static`, `concat`, `streaming concat`, `assignment pattern`

Severity

Warning

WRN_1454

Invalid use of an identifier.

Language

Verilog

Message Details

```
Invalid use of identifier ( <identifier> )
```

Severity

Warning

WRN_1455

Invalid void function call of function with non-void return type

Language

Verilog

Message Details

```
Invalid void function call of function with non-void return
type '<return-type>'
```

Severity

Warning

WRN_1456

UDP could not be translated to it's synthesizable model.

Language

Verilog

Rule Description

This UDP could not be translated to its synthesizable RTL model.

The reason can be one of the following:

1. Presence of more than 1 clock.
2. UDP is probably a combination of flip-flop or latch.

Message Details

```
UDP ( <UDP-name> ) could not be translated to its synthesizable
model
```

Severity

Warning

WRN_1457

Macro evaluates to non-static value.

Language

Verilog

Message Details

Macro (<macro-name>) evaluates to <value> value

Severity

Warning

WRN_1458

Dimension argument to SV array query function should be integer or get evaluated to an integer.

Language

Verilog

Rule Description

The dimension argument to SystemVerilog array query functions is the second argument given. It selects the dimension of the array regarding which the function returns the required information. So it has to be an integer, or a type which is convertible to an integer (single-bit and packed types), or a function which returns any of these.

Message Details

```
Dimension argument to SV array query function '<function>'
cannot be evaluated to an integer
```

Severity

Warning

WRN_1459

Assignment to an enum variable must be of the same enum type in port connection and function arguments.

Language

Verilog

Rule Description

SpyGlass reports this warning if assignment to an enum variable is not to the same enum type in port connection and function arguments.

Enums are strongly typed variables and assignment to an enum variable must be to the same enum type.

Some simulators do not allow assignment of a variable of different type to an enum variable.

The following example shows violating and non-violating cases for this rule:

```
typedef enum logic {
ab,
xy
} order;


module first;
  order value;

  top top_1 (.read(value));
  // Violation as enum variable "value" is being assigned
  // "read" of type logic.


  bottom bot_1 (.ext(value));
 // No violation as enum variable "value" is being
 // assigned "ext" of the same enum type.

endmodule

module top (read);
  output logic read;
```

```
endmodule


module bottom (ext);
  output order ext;
endmodule
```

Message Details

Incompatible assignment to enum in <port | argument> (<port/argument-name>) of <function | module | interface> (<function/module/interface-name>).

Severity

Warning

WRN_1460

The specified pragma is unsupported and is ignored.

Description

The *WRN_1460* rule reports a violation if you specify an unsupported pragma in the code.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[WARNING] Unsupported pragma (<pragma>) is ignored

Consequences of Not Fixing

The reported pragma is ignored from SpyGlass analysis.

How to Debug and Fix

Remove the reported pragma from the code.

Example Code and/or Schematic

Consider the following example:

```
function[AMN_width:0] DWF_bin
  // synopsis map_to_operator BINENC_OPS
  input [A_width:0] inp1;
  reg[AMN_width:0] temp;
  reg flag , xflag;
  integer i;
endfunction
```

For the above example, the *WRN_1460* rule reports a violation for the `map_to_operator` pragma because it is not supported in SpyGlass.

Default Severity Label

Warning

WRN_1461

Bind statement is ignored as the specified design unit is either an interface or is not present in the design

Description

The *WRN_1461* rule reports a violation in any of the following cases:

- The design unit specified in the `bind` statement is an interface.
- The design unit specified in the `bind` statement is not present in the design.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[WARNING] Ignoring bind statement as target design unit (<design-unit>) is either an interface or does not exist in design

Consequences of Not Fixing

If you do not fix this violation, the `bind` statement is ignored by SpyGlass.

How to Debug and Fix

Specify only module-level binding.

Example Code and/or Schematic

The *WRN_1461* rule reports a warning in the following example:

```
 interface intf;
   ...
 endinterfaceb
bind top2 intf i1(); // Warning
                     // SpyGlass ignores this bind statement
                     // because top2 is not defined.
module top;
```

```
...
endmodule
```

Default Severity Label

Warning

WRN_1462

Incompatible array assignment due to mismatch in sign or state is non standard

Description

The *WRN_1462* rule reports a violation for incompatible array assignments due to mismatch in the `sign` or `state` types.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[FATAL] `Incompatible array assignment due to mismatch in sign or state is non-standard`

The above violation becomes a **warning** if you specify the following project-file command:

```
allow_non_lrm to yes
```

Consequences of Not Fixing

If you do not fix this violation, the implications at the point of assignment may change due to mismatch in `sign` or `state` of the expressions on LHS and RHS.

How to Debug and Fix

Match the `sign` and `state` types of declarations of the LHS and RHS expressions of the reported array assignment.

Example Code and/or Schematic

The following example shows the violating and non-violating cases of this rule:

```
module top;
```

```
 logic signed conn1[3:0];
 logic conn2[3:0];
 bottom inst1(.a(conn1)); // Violation: Incompatible port
                          // connection due to mismatch
                          // in sign
 bottom inst2(.a(conn2)); // No violation: Compatible port
                  // connection
 endmodule
module bottom(input wire a[3:0]);
endmodule
```

Default Severity Label

Warning

WRN_1463

Design unit defined in a file is not ending in the same file

Description

This rule reports a violation when the design unit defined in a file is not ending in the same file and is spanning across multiple file.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[WARNING] Design unit (<design-unit-name>) is not ending in same file

Example Code and/or Schematic

The following example shows the violating case of this rule:

```
File: test.v
module A(a,b);  // Violation
      input a,b;
File test1.v
      reg c;
endmodule
```

In the above example, the *WRN_1463* rule reports a violation because the module A is begun in file test.v but is not ending in the test.v file.

Default Severity Label

Warning

Syntax Informational Rules

Rules of this category report an informational message for certain areas in the code.

The following rules of this category are switched off by default:

- *INFO_1013*
- *INFO_1018*
- *INFO_1019*

INFO_558

Warning/Error detected within translate off/on block. Remove set_option "disable_hdlin_translate_off_skip_text" yes to suppress this warning/error

Language

VHDL

Rule Description

To suppress the warning or error within the `translate off/on` block, remove the following project-file command:

```
set_option disable_hdlin_translate_off_skip_text yes
```

Message Details

```
<Warning | Error> detected within translate off/on block.
Remove set_option "disable_hdlin_translate_off_skip_text" yes
to suppress this <Warning | Error>
```

Severity

Info

INFO_620

Warning/Error detected within synthesis off/on block. Remove set_option "disable_hdlin_synthesis_off_skip_text" yes to suppress this error

Language

VHDL

Rule Description

To suppress the warning or error within the synthesis off/on block, specify the following project-file command:

```
set_option hdlin_synthesis_off_skip_text yes
```

Message Details

```
<Warning | Error> detected within synthesis off/on block. Use
set_option "hdlin_synthesis_off_skip_text" yes to suppress this
<Warning | Error>
```

Severity

Info

INFO_635

Invalid syntax in the protect directive

When to Use

Use this rule to check the correctness of a decryption envelope.

Description

The *INFO_635* rule reports a violation if a `protect` directive contains an invalid syntax.

Language

VHDL

Messages and Suggested Fix

This rule reports the following message:

[INFO] `Invalid syntax for `protect directive (<directive>)`

How To Debug and Fix

Correct the syntax in the reported directive.

Example Code and/or Schematic

The *INFO_635* rule reports violation in the following cases:

- When the value for a `pragma` expression is missing, as shown in the following example:

  ```
  `protect key_method = ,
  ```

- When the specified `protect` keyword is invalid, as shown in the following example:

  ```
  `protect my_key_method = "rsa"
  ```

Default Severity Label

Info

INFO_636

The specified value is not supported for the mentioned keyword

When to Use

Use this rule to check the correctness of a decryption envelope.

Description

The *INFO_636* rule reports a violation if the value assigned to a keyword in a decryption envelope is not supported by that keyword.

Language

VHDL

Messages and Suggested Fix

This rule reports the following message:

```
[INFO] Value <value> for keyword <keyword> is not supported
```

Consequences of Not Fixing

If you do not fix this violation, the decryption block is not decrypted and is ignored from SpyGlass analysis. This would mean missing RTL that may result in missing functionality.

How To Debug and Fix

Specify a correct value to the reported keyword.

Example Code and/or Schematic

The *INFO_636* rule violation does not appear for the following line because the `key_method` keyword is assigned the `rsa` value, which is supported by this keyword:

```
`protect key_method = "rsa" //RSA algorithm is supported
```

However, a violation appears for the following line because `key_method` does not support the `rs` value:

```
`protect key_method = "rs"
```

Default Severity Label

Info

INFO_994

Assignment of concatenation to unpacked type is non-standard

Language

Verilog

Rule Description

The *INFO_994* rule reports a violation if assignment to an unpacked type is done through concatenation.

As per IEEE 1800-2005, assignment to an unpacked type is done through an assignment pattern and not concatenation.

SpyGlass allows concatenation to be assigned to an unpacked type only if both the following conditions are true:

- The width of concatenation is same as that of its target.
- The `allow_non_lrm` project-file command is set to `yes`.

The following examples show the violating and non violating cases of this rule:

```
int a[4] = {0,1,2,3};  // Violation
int a[4] = '{0,1,2,3}; // No violation
```

Message Details

`Assignment of concatenation to unpacked type is non-standard`

Severity

Info

INFO_995

Package defined in a source file is not read because the same package is already available in the work library.

Language

Verilog

Rule Description

The *INFO_995* rule reports a violation if a package, precompiled in a previous SpyGlass run, is present in the file currently being analyzed.

In such cases, SpyGlass considers the package from the precompiled dump and skips package analysis from the current file.

Message Details

```
Package ( <package> ) is skipped
```

Severity

Info

INFO_996

Invalid syntax detected in the pragma directive

When to Use

Use this rule to check the correctness of a Decryption Envelope.

Description

The *INFO_996* rule reports a violation if an incorrect syntax is detected in the pragma directive.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[INFO] Invalid syntax for 'pragma directive <directive>

Consequences of Not Fixing

Not applicable

How To Debug and Fix

Double-click on the violation to cross-probe to the RTL line where the pragma directive is present. Correct the syntax error in that directive.

Example Code and/or Schematic

Consider the following pragma directive containing a syntax error:

```
`pragma protect version=()
```

For the above directive, this rule reports the following violation message:

```
Invalid syntax for `pragma directive ( near ) )
```

Default Severity Label

Info

INFO_997

The specified value is not supported for the mentioned keyword

When to Use

Use this rule to check the correctness of a Decryption Envelope.

Description

The *INFO_997* rule reports a violation if the specified value is not supported for the mentioned keyword.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

```
[INFO] Value <value> for keyword <keyword> is not supported
```

Consequences of Not Fixing

If you do not fix this violation, the decryption block is not decrypted and is ignored from SpyGlass analysis. This would mean missing RTL that may result in missing functionality.

How To Debug and Fix

Double-click on the violation to cross-probe to the RTL line where the pragma directive is present. Specify a supported value for the keyword.

Example Code and/or Schematic

This rule reports a violation for the following example because the `rs` value is not supported by `key_method`:

```
`pragma protect key_method = "rs"
```

However, no violation appears in the following case because the `rsa` value is supported by `key_method`:

```
`pragma protect key_method = "rsa"
```

Default Severity Label

Info

INFO_999

White space before or after `` syntax is non-standard

Description

The *INFO_999* rule reports a violation when a white space is found before or after the `` (token pasting operator) during macro expansion, which is non-standard.

To ignore the non LRM construct, specify the following option in the project-file:

```
set_option non_lrm_options ignore_tokenpasting_space
```

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[INFO] The white space before or after `` (token pasting operator) is non-standard and is ignored via 'set_option non_lrm_options ignore_tokenpasting_space'

Example Code and/or Schematic

Consider the following code:

```
`define NAME(a,b,c) \ext``a``_reg
module top;
reg `NAME(\X ,Y,Z) ;
endmodule
```

In the above example, the *INFO_999* rule reports a violation as a non-standard syntax is used.

Default Severity Label

Info

INFO_1000

Use of automatic keyword is ignored

Language

Verilog

Rule Description

This rule reports that the keyword `automatic`, used in the non-function declaration, has been ignored.

For example:

```
always_comb
begin
automatic logic[2:0] numcnst = '0;
...
end
```

Message Details

[INFO] 'automatic' keyword is ignored

Severity

Info

INFO_1001

Type parameter cannot be overridden through console options.

Language

Verilog

Rule Description

Type parameter cannot be overridden through Atrenta Console options.

For example, this rule reports a violation in the following case:

```
module top;
  parameter type p1 = shortint;  // p1 is a type parameter
  p1 j = 0;
endmodule
```

Message Details

```
Type parameter (<param-name>) cannot be overridden through
console options
```

Severity

Info

INFO_1004

The specified include-file could not be found or opened in read mode.

Language

Verilog

Rule Description

SpyGlass reports this informational message when an include file specified inside a skipped or stopped library design unit does not exist at the specified path.

SpyGlass ignores this message because its severity is not fatal.

Message Details

```
Include file ( <incl-file-name> ) specified inside skipped/
stopped design unit could not be found or opened in read mode
from current working directory ( <cwd-path> ) or other include
directory paths( if any )
```

Severity

Info

INFO_1006

Module/UDPs having same name technology library cells are ignored from Verilog design files and Verilog precompile dump

Language

Verilog

Rule Description

SpyGlass reports this message when a module/UDP is ignored because a technology library cell of the same name is given a higher priority when the following command is specified in a project file:

```
set_option prefer_tech_lib yes
```

Message Details

```
<module/UDP-name> is ignored as technology library cell
'<cell-name>' has been given higher preference via 'set_option
prefer_tech_lib yes'
```

Severity

Info

INFO_1007

Module/UDP overrides another module/UDP declared in file <file> at line <line>.

Language

Verilog

Rule Description

More than one module/UDP is given with the same name. The definition given last on in the command will override others. Modules will be given preference over UDPs.

Message Details

```
<Module | UDP> '<name1>' overrides <name2> declared in file
'<file-name>' at line '<line-num>'
```

Severity

Info

INFO_1008

AMS keyword found.

Language

Verilog

Message Details

AMS keyword (<keyword>) found

Severity

Info

INFO_1009

Ignoring Covergroup declaration in the design.

Language

Verilog

Rule Description

Covergroup construct is not supported and will be ignored.

Message Details

```
Ignoring Covergroup declaration in the design
```

Severity

Info

INFO_1010

UDP translated to its synthesizable model.

Language

Verilog

Rule Description

This UDP has been successfully translated to its synthesizable RTL model.

Message Details

```
UDP ( <UDP-name> ) translated to its synthesizable model
```

Severity

Info

INFO_1011

Module was previously defined in another file.

Language

Verilog

Rule Description

Reports duplication of module names for modules defined in source and library files (loaded `v` and `y` files).This message is even reported for those modules in the loaded library files which are not even analyzed.

Message Details

```
Module (<module-name>) previously defined in file (<file-name>)
at line (<line-num>)
```

Severity

Info

INFO_1012

SystemVerilog version 1800-2009 is unsupported

Language

Verilog

Rule Description

SpyGlass reports this informational message if you specify the version `1800-2009` with the `` `begin_keywords `` compiler directive. This directive does not support the `1800-2009` version.

If you specify this version for the `` `begin_keywords `` directive, SpyGlass uses the version `1800-2005`.

Message Details

```
Version 1800-2009 specified with `begin_keywords is
unsupported. Version 1800-2005 will be used instead
```

Severity

Info

INFO_1013

SystemVerilog construct found.

Language

Verilog

Rule Description

A SystemVerilog construct has been found in the design.

NOTE: *The INFO_1013 rule is switched off by default.*

Message Details

```
SystemVerilog construct ( <construct> ) found
```

Severity

Info

INFO_1014

There is an empty port in port list of design unit.

Language

Verilog

Rule Description

SpyGlass reports this informational message if you specify an empty port in the port list of a design unit.

NOTE: *This message is reported only in non-ANSI style header.*

The specification of the empty port can be a valid statement. However, it may not be intended. For example, consider the following declaration:

```
module m( a, , c, );
```

In the above declaration, there are four ports out of which two are empty ports. Therefore, SpyGlass reports a violation message in this case.

Message Details

```
Empty port in design unit header
```

Severity

Info

INFO_1015

SystemVerilog flag is enabled due to loading of precompiled SystemVerilog module.

Language

Verilog

Rule Description

In a SpyGlass run in which the `set_option enableSV yes` command is not specified and a precompiled SystemVerilog module is loaded, SpyGlass internally enables the SystemVerilog flag. This rule also flags the library name and it's physical path from where precompiled module is loaded.

Message Details

```
SystemVerilog flag enabled due to precompiled SystemVerilog
module picked from Library <library> (path: <path> )
```

Severity

Info

INFO_1017

Maximum error count exceeded, analysis terminated.

Language

Verilog

Message Details

```
Maximum permissible error count of (<number>) exceeded.
Analysis terminated
```

Severity

Info

INFO_1018

A v2k construct has been used with "set_option disablev2k yes".

Language

Verilog

Rule Description

Please run without the `set_option disablev2k yes` project file command to support the specified v2k construct.

NOTE: *The INFO_1018 rule is switched off by default.*

Message Details

```
v2k construct ( <construct-name> ) used with 'set_option
disablev2k yes'
```

Severity

Info

INFO_1019

Verilog 2001 construct found.

Language

Verilog

Rule Description

NOTE: *The INFO_1019 rule is switched off by default.*

Message Details

```
Verilog 2001 construct ( <construct> ) found
```

Severity

Info

INFO_1020

Generating a new name for duplicate module.

Language

Verilog

Message Details

```
Generating new name ( <name> ) for duplicate module ( <module>
)
```

Severity

Info

INFO_1480

Identifies functions/tasks that have been used before definition.

Language

Verilog

Message Details

```
Function|Task <function/task-name> defined in file <function/
task-definition-file> at line <line-number> is used before
declaration
```

Severity

Info

Elaboration Rules (Analysis Stage)

Rules of this category report a violation during elaboration in the analysis stage.

ELAB_117

Architecture referenced does not exist.

Language

VHDL

Rule Description

This error is generated by SpyGlass to indicate that an architecture (by context) does not exist. Modify the format of the selected name, which is invalid because of a missing or incorrectly formed use clause; a typo; or incorrect use of a record element.

NOTE: *One of the possible causes of the ELAB_117 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Message Details

```
No Architecture <library-name>.<entity-
name>(<architecture-name>) exists
```

Severity

ElaborationError

ELAB_270

Generic/port Map must have an actual corresponding to each formal.

Language

VHDL

Rule Description

NOTE: *One of the possible causes of the ELAB_270 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Message Details

```
<Generic | port> Map has no actual corresponding to formal
<formal>
```

Severity

ElaborationError

ELAB_433

Valid binding must exist for each instance used during elaboration

Language

VHDL

SpyGlass generates this elaboration error message for default binding of instances when any of the following scenarios are encountered:

1. No default value is assigned to formal parameter.
2. Component formal parameter is not present in entity.
3. Component formal parameter has different size in entity.
4. Component formal parameter has different type in entity.
5. Component formal parameter has different direction in entity.
6. The source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the *WRN_612* rule).

Message Details

```
No binding exists for instance <instance-label> during
elaboration of <instance-hierarchy-name> <binding-message>
```

Severity

ElaborationError

ELAB_434

Internal Elaborator Error during elaboration

Language

VHDL

Message Details

```
Internal Elaborator Error while elaborating <instance-label>
during elaboration of <instance-hierarchy-name>. Please contact
Atrenta Support
```

Severity

INTERNAL_FATAL

ELAB_440

Value used/generated should not be out of range of formal

Language

VHDL

Rule Description

NOTE: *One of the possible causes of the ELAB_440 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Message Details

```
Value <value> is out of range of formal <formal> during
elaboration of <name>
```

Severity

ElaborationError

ELAB_441

Formal of unconstrained array type cannot be elaborated if size of corresponding actual is not defined

Language

VHDL

Rule Description

NOTE: *One of the possible causes of the ELAB_441 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Message Details

```
Formal <formal> of unconstrained array type cannot be
elaborated since size of corresponding actual in <name> is not
defined
```

Severity

ElaborationError

ELAB_442

Size of initial value should match size of elaborated declaration during elaboration

Language

VHDL

Rule Description

NOTE: *One of the possible causes of the ELAB_442 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Severity

ElaborationError

Message Details

```
Size of initial value does not match size of elaborated decl
<declaration> during elaboration of <name>
```

ELAB_445

Elaborated width of instance/subprogram port/argument does not match with the width of its corresponding declaration.

Language

VHDL

Rule Description

The ELAB 445 error is reported by SpyGlass in the following situations:

Scenario A: SpyGlass found mismatch (while elaborating) between the width of a signal connected to an instance port and the port width declared in component declaration.

Scenario B: SpyGlass found mismatch (while elaborating) between the width of an argument passed to subprogram and the formal width of the argument declared in subprogram declaration.

For example (scenario A):

```
entity top is
   port ( AddA:    in   std_logic_vector(1 downto 0) );
end entity top;
architecture struct of top is
   component BOTTOM
      generic (SizeIn: integer);
      port ( A: in   std_logic_vector(SizeIn - 1 downto 0) );
   end component;
begin
   I1: BOTTOM
         generic map (SizeIn => 3)
         port map ( AddA => A);
end architecture STRUCT;
```

Here, while instantiating I1 - the size of "A" has become "3 bits wide" - due to the generic map. However, the signal connected to "A" is only 2 bits

wide. Hence, this Elaboration Error is reported.

For example (scenario B):

```
entity ent is
generic (xyz: integer :=1);
end;
architecture rtl of ent is
  function VEC2INT (S: bit_vector ( 1 to 8) )
  return integer is
   begin
    ...
 end VEC2INT;
  signal XBUS:  bit_vector (0 to xyz);
begin
  PROC1: process
  variable XVAL :integer := 1;
  begin
    XVAL := VEC2INT (XBUS);        --    function call
  end process;
end rtl;
```

Suppose, while instantiating "ent", the value of "xyz" is anything but "7". So now, while calling function VEC2INT, the argument being passed has a size different from "8". This situation will result in ELAB_445.

NOTE: *One of the possible causes of the ELAB_445 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Message Details

`Mismatch of width between <string1> <string2>`

Where:

- *<string1>* displays argument (actual) and corresponding width.

- <*string2*> displays argument (formal), corresponding width, instance name, file name, and line number.

Severity

ElaborationError

ELAB_475

Function call recursion stack too deep - possibly infinite recursion

Language

VHDL

Message Details

```
Function call recursion stack too deep - possibly infinite
recursion
```

Severity

ElaborationError

ELAB_497

Variable must not have illegal/NULL range

Language

VHDL

Rule Description

NOTE: *One of the possible causes of the ELAB_497 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Message Details

```
Variable <variable> has invalid/null range - cannot evaluate
the function. Function call made from file '<file-name>' at
line number: <line-num>.
```

Severity

ElaborationError

ELAB_535

Package Body missing

Language

VHDL

Rule Description

NOTE: *One of the possible causes of the ELAB_535 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Severity

ElaborationError

ELAB_540

Actual must be specified for formal port

Language

VHDL

Message Details

No actual specified for formal port "<port-name>" in instance "<inst-name>". Some simulators do not support this.

Severity

ElaborationWarning

ELAB_557

For Generate range specified should not be too large

Language

VHDL

Rule Description

NOTE: *One of the possible causes of the ELAB_557 error is that the source file is newer than the corresponding SpyGlass Precompiled Library (flagged by the WRN_612 rule).*

Message Details

```
For Generate range "<range>" is too large
```

Severity

ElaborationError

ELAB_569

Invalid/null range in the expression is causing an error in the evaluation of an IEEE function.

Language

VHDL

Description

Expression has a null range during evaluation like expression with constraints -1 down to 0.

Message Details

`Invalid/null range in the expression <expression> is causing an error in the evaluation of an IEEE function`

Severity

ElaborationError

ELAB_633

Infinite loop in the code

Description

The *ELAB_633* rule reports a violation if RTL contains an infinite loop.

Language

VHDL

Messages and Suggested Fix

This rule reports the following message:

[ElaborationError] `Possibly infinite recursion in the rtl`

Consequences of Not Fixing

If you do not fix this violation, the infinite loop exhaust system resources, such as memory.

How to Debug and Fix

Update the code to remove the infinite loop.

Example Code and/or Schematic

For the following example, the ELAB_633 rule reports a violation because of the infinite loop highlighted in red:

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
package gp_type is
function func(x:integer) return integer;
end gp_type;


package body gp_type is
function func(x:integer) return integer is
begin
   while (true) loop
   end loop;
```

```
   return x;
end;
end gp_type;

LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
use work.gp_type.all;
ENTITY ent IS
port(
y : out std_logic_vector(2 downto 0)
);
END ent;

ARCHITECTURE arch OF ent IS
BEGIN
y(func(1)) <= '1';
END arch;
```

Default Severity Label

ElaborationError

Synthesis Rules (Analysis Stage)

Rules of this category report a violation during synthesis in the analysis stage.

SYNTH_77

Both blocking & non-blocking assignments should not be done on the variable

Language

Verilog

Message Details

`Both blocking & non-blocking assignments are being done on the variable ( <variable> )`

Severity

SynthesisWarning

SYNTH_78

The specified construct is not synthesizable.

Language

Verilog

Rule Description

SpyGlass generated this synthesis warning for those constructs that are not synthesizable. Such unsynthesizable constructs are ignored for synthesis.

Message Details

```
'<construct>' construct is not synthesizable. Ignoring for synthesis
```

Severity

SynthesisWarning

SYNTH_89

Initial Assignment at Declaration is ignored by synthesis.

Language

Verilog

Message Details

```
Initial Assignment at Declaration for ( <declaration> ) is
ignored by synthesis
```

Severity

SynthesisWarning

SYNTH_92

SPECIFY construct is not supported in synthesis.

Language

Verilog

Rule Description

SPECIFY blocks usually contain delay/timing check-related information. Therefore, they are not supported by synthesis tools.

Message Details

```
SPECIFY construct is not supported in synthesis
```

Severity

SynthesisWarning

SYNTH_93

TIME data type is not supported in synthesis.

Language

Verilog

Rule Description

SpyGlass reports this warning to indicate the declaration of the `time` data type. This data type is not supported during synthesis.

For example, this rule reports a violation in the following case:

```
module top;
  parameter p = 10;
  time t;  // Declaration of the time data type
endmodule
```

Message Details

`TIME data type is not supported in synthesis`

Severity

SynthesisWarning

SYNTH_102

FORCE statements are not synthesizable.

Language

Verilog

Message Details

`FORCE statements are not synthesizable`

Severity

SynthesisError

SYNTH_103

RELEASE statements are not synthesizable.

Language

Verilog

Message Details

`RELEASE statements are not synthesizable`

Severity

SynthesisError

SYNTH_104

DEASSIGN statements are not synthesizable.

Language

Verilog

Message Details

`DEASSIGN statements are not synthesizable`

Severity

SynthesisError

SYNTH_106

WAIT statements are not synthesizable.

Language

Verilog

Message Details

`WAIT statements are not synthesizable`

Severity

SynthesisError

SYNTH_114

A module is not synthesizable due to presence of TRI0 net declaration.

Language

Verilog

Message Details

```
Module not synthesizable due to presence of TRI0 net
declaration
```

SYNTH_115

A module is not synthesizable due to presence of TRI1 net declaration.

Language

Verilog

Message Details

```
Module not synthesizable due to presence of TRI1 net
declaration
```

Severity

SynthesisError

SYNTH_118

Module not synthesizable due to presence of TRIREG net declaration.

Language

Verilog

Message Details

```
Module not synthesizable due to presence of TRIREG net
declaration
```

Severity

SynthesisError

SYNTH_126

Procedural continuous assign statements are not synthesizable.

Language

Verilog

Rule Description

The *SYNTH_126* rule reports a violation if you use procedural continuous `assign` statements.

Such statements are not synthesizable and are ignored during synthesis.

For example, this rule reports a violation in the following case:

```
assign out = in;
  module top (in, out);
    input in;
    output out;
    reg out;
    always
    begin
      assign out = in;  //Violation
    end
  endmodule
```

Message Details

`Procedural continuous assign statements are not synthesizable`

Severity

SynthesisError

SYNTH_130

Gates like NMOS, RNMOS, PMOS, RPMOS, CMOS, RCMOS, RTRAN, TRAN, TRANIF0, RTRANIF0, TRANIF1, and RTRANIF1 are not supported in synthesis.

Language

Verilog

Message Details

```
<gate> gate types are not supported
```

Severity

SynthesisError

SYNTH_131

EVENT declarations are not synthesizable, therefore, an event cannot be triggered in synthesis.

Language

Verilog

Message Details

```
EVENT not synthesizable, so "<event>" cannot be triggered
```

Severity

SynthesisError

SYNTH_132

Hierarchical references are not supported in synthesis.

Message Details

```
Hierarchical references ( <hierarchical-reference> ) are not
supported for synthesis
```

Severity

SynthesisError

SYNTH_VE_INACTIVE_132

Hierarchical references are not supported in synthesis.

Message Details

```
Hierarchical references ( <hierarchical-reference> ) are not
supported for synthesis
```

NOTE: *The SYNTH_VE_INACTIVE_132 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Severity

SynthesisError

SYNTH_133

Only one statement, which can be single if or single if-else or if-else-if chain, is allowed at the top level in the always block.

Language

Verilog

Rule Description

For an asynchronous reset/set `always` block, it is required that there be only one statement, which can be an `if` (or an `if-else` or an `if-else-if` chain) statement.

For example, the following code will violate:

```
wire clk, rst;
reg q;
always @ (posedge clk or negedge rst)
begin
q = 1'b0;
if (!rst)
...
end
```

Message Details

`Asynchronous reset/set always block should have only one statement (single if or single if-else or if-else-if chain) at the top level`

Severity

SynthesisError

SYNTH_VE_INACTIVE_133

Only one statement, which can be single if or single if-else or if-else-if chain, is allowed at the top level in the always block.

Language

Verilog

Rule Description

For an asynchronous reset/set `always` block, it is required that there be only one statement, which can be an `if` (or an `if-else` or an `if-else-if` chain) statement.

For example, the following code will violate:

```
wire clk, rst;
reg q;
always @ (posedge clk or negedge rst)
begin
q = 1'b0;
if (!rst)
...
end
```

NOTE: *The SYNTH_VE_INACTIVE_133 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Asynchronous reset/set always block should have only one statement (single if or single if-else or if-else-if chain) at the top level`

Severity

SynthesisError

SYNTH_135

Vector reference to scalar net is invalid.

Description

Vector reference to a scalar net is invalid.

The following example shows the violating cases of this rule:

```
wire w;
assign w[2:1] = 2'b11;   //Violation
assign w[0] = 1'b1;      //Violation
```

Language

Verilog

Message Details

`Invalid vector reference to scalar net - (<net-name>)`

Severity

SynthesisError

SYNTH_VE_INACTIVE_135

Vector reference to scalar net is invalid.

Description

Vector reference to a scalar net is invalid.

The following example shows the violating cases of this rule:

```
wire w;
assign w[2:1] = 2'b11;   //Violation
assign w[0] = 1'b1;      //Violation
```

NOTE: *The SYNTH_VE_INACTIVE_135 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Language

Verilog

Message Details

`Invalid vector reference to scalar net - (<net-name>)`

Severity

SynthesisError

SYNTH_137

System function $cast is not supported in synthesis.

Language

Verilog

Rule Description

System function $cast is not supported in synthesis.

Message Details

```
System function $cast is not supported
```

Severity

SynthesisError

SYNTH_138

Illegal vector reference to scalar register is not allowed.

Description

Illegal vector reference to scalar register is not allowed.

The following example shows a violating case of this rule:

```
reg  a;
wire  b;
assign b = a[2];     //Violation
```

Language

Verilog

Message Details

`Illegal vector-reference to scalar register - (<register-name>)`

Severity

SynthesisError

SYNTH_VE_INACTIVE_138

Illegal vector reference to scalar register is not allowed.

Description

Illegal vector reference to scalar register is not allowed.

The following example shows a violating case of this rule:

```
reg  a;
wire  b;
assign b = a[2];     //Violation
```

NOTE: *The SYNTH_VE_INACTIVE_138 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Language

Verilog

Message Details

`Illegal vector-reference to scalar register - (<register-name>)`

Severity

SynthesisError

SYNTH_147

An asynchronous reset/set always block may have the reset/set condition only as a simple identifier or its negation.

Language

Verilog

Rule Description

For an asynchronous reset/set block, it is required that the conditional expression of any 'if' statement should use the asynchronous reset/set signal only as a whole or its negation (that is, in conjunction with the unary operators ! or ~ only).

Message Details

```
Asynchronous reset/set always block may have the reset/set
condition only as a simple identifier or its negation (! or ~)
```

Severity

SynthesisError

SYNTH_VE_INACTIVE_147

An asynchronous reset/set always block may have the reset/set condition only as a simple identifier or its negation.

Language

Verilog

Rule Description

For an asynchronous reset/set block, it is required that the conditional expression of any 'if' statement should use the asynchronous reset/set signal only as a whole or its negation (that is, in conjunction with the unary operators ! or ~ only).

NOTE: *The SYNTH_VE_INACTIVE_147 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Asynchronous reset/set always block may have the reset/set
condition only as a simple identifier or its negation (! or ~)
```

Severity

SynthesisError

SYNTH_148

For the reset/set condition in an asynchronous reset/set always block, comparison of asynchronous signal can only be made with a constant expression.

Language

Verilog

Rule Description

It is illegal to compare an asynchronous reset/set signal with a non-constant expression.

For example, the following code will violate:

```
wire clk, rst, abc;
always @ (posedge clk or negedge rst)
begin
    if (rst != abc)
    ...
end
```

In the above case, "abc" is non-constant.

Message Details

`In asynchronous reset/set always block, comparison is being made to non-constant expression ( <expression> ) in reset/set condition`

Severity

SynthesisError

SYNTH_VE_INACTIVE_148

For the reset/set condition in an asynchronous reset/set always block, comparison of asynchronous signal can only be made with a constant expression.

Language

Verilog

Rule Description

It is illegal to compare an asynchronous reset/set signal with a non-constant expression.

For example, the following code will violate:

```
wire clk, rst, abc;
always @ (posedge clk or negedge rst)
begin
    if (rst != abc)
    ...
end
```

In the above case, "abc" is non-constant.

NOTE: *The SYNTH_VE_INACTIVE_148 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
In asynchronous reset/set always block, comparison is being
made to non-constant expression ( <expression> ) in reset/set
condition
```

Severity

SynthesisError

SYNTH_149

Only '==' and '!=' binary operators are allowed in validation of the reset/set condition.

Language

Verilog

Rule Description

For an asynchronous reset/set block, only the operators "==" and "!=" are allowed to be used for validating the reset/set condition. No other binary operator can be used as part of the conditional expression.

Message Details

```
Only '==' and '!=' binary operators are allowed in validation
of the asynchronous reset/set condition
```

Severity

SynthesisError

SYNTH_VE_INACTIVE_149

Only '==' and '!=' binary operators are allowed in validation of the reset/set condition.

Language

Verilog

Rule Description

For an asynchronous reset/set block, only the operators "==" and "!=" are allowed to be used for validating the reset/set condition. No other binary operator can be used as part of the conditional expression.

NOTE: *The SYNTH_VE_INACTIVE_149 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
Only '==' and '!=' binary operators are allowed in validation
of the asynchronous reset/set condition
```

Severity

SynthesisError

SYNTH_87

Invalid use of Tristate value must be avoided.

Language

Verilog

Rule Description

SpyGlass flags this warning to indicate that tristate value is used in an expression in an invalid way.

Message Details

```
Invalid use of Tristate value
```

Severity

SynthesisWarning

SYNTH_VE_INACTIVE_87

Invalid use of Tristate value must be avoided.

Language

Verilog

Rule Description

SpyGlass flags this warning to indicate that tristate value is used in an expression in an invalid way.

NOTE: *The SYNTH_VE_INACTIVE_87 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

`Invalid use of Tristate value`

Severity

SynthesisWarning

SYNTH_154

Both edge control & non-edge control expressions cannot be specified in the sensitivity list.

Language

Verilog

Rule Description

It is illegal to have an `always` block sensitive to both edge control and non-edge control expressions.

For example, the following code will violate:

```
wire clk, rst;
always @ (posedge clk or rst)
begin
...
end
```

Message Details

```
Both edge control and non-edge control expressions cannot be
specified in the sensitivity list
```

Severity

SynthesisError

SYNTH_155

Constant expressions cannot be used in association with edge control expressions.

Language

Verilog

Rule Description

Constant expressions (such as, parameters or localparams or genvars) cannot be used as edge control expressions.

For example, the following code will violate:

```
wire clk;
parameter rst = 3;
always @ (posedge clk or negedge rst)
begin
...
end
```

Message Details

`Constants cannot be used as edge control expressions`

Severity

SynthesisError

SYNTH_162

The if-elseif statement chain conditional expressions should check for all the asynchronous reset/set signals first.

Language

Verilog

Rule Description

For an asynchronous reset/set `always` block, it is required that ALL the asynchronous reset/set signals should be checked in the `if-elseif` statement chain before any other checking is done.

For example, the following code will violate:

```
wire a, clk, rst1, rst2;
always @ (posedge clk or negedge rst1 or negedge rst2)
begin
    if (!rst1)
    ...
    else if (!a) <-- will violate here
    ...
    else if (!rst2)
    ...
    else
    ...
end
```

Message Details

```
The 'if-else if' statement chain conditional expression should
check for all the asynchronous reset/set signals first
```

Severity

SynthesisError

SYNTH_VE_INACTIVE_162

The if-elseif statement chain conditional expressions should check for all the asynchronous reset/set signals first.

Language

Verilog

Rule Description

For an asynchronous reset/set `always` block, it is required that ALL the asynchronous reset/set signals be checked for, in the `if-elseif` statement chain before any other checking is done.

For example, the following code will violate:

```
wire a, clk, rst1, rst2;
always @ (posedge clk or negedge rst1 or negedge rst2)
begin
    if (!rst1)
    ...
    else if (!a) <-- will violate here
    ...
    else if (!rst2)
    ...
    else
    ...
end
```

NOTE: *The SYNTH_VE_INACTIVE_162 rule is flagged when the* `set_option enable_inactive_rtl_checks yes` *command is used in the project file.*

Message Details

```
The 'if-else if' statement chain conditional expression should
check for all the asynchronous reset/set signals first
```

Severity

SynthesisError

SYNTH_164

In the implicit style sequential state machine, states can only be updated if controlled by same clock phase.

Language

Verilog

Message Details

```
In the implicit style sequential state machine, states can only
be updated if controlled by same clock phase
```

Severity

SynthesisError

SYNTH_165

In the implicit style sequential state machine, only one edge controlled event expression is allowed.

Language

Verilog

Message Details

```
In the implicit style sequential state machine, only one edge-
controlled event expression is allowed
```

Severity

SynthesisError

SYNTH_166

It is illegal for an implicit style sequential state machine to have more than one edge specification in its event control statement.

Language

Verilog

Rule Description

In an implicit style sequential state machine, only one edge specification should be there in the event control statement.

For example, the following code will violate:

```
always
begin
 @(posedge clk or negedge rst) a <= b; <--will
                                          violate here
end
```

Message Details

```
In the implicit style sequential state machine, the event
control statement may not have more than one edge specification
```

Severity

SynthesisError

SYNTH_167

Only simple variables are allowed in the event expression with 'posedge' or 'negedge' qualifier

Language

Verilog

Rule Description

Only simple variables are allowed for event expression with 'posedge' or 'negedge' qualifier.

For example, the following code will violate:

```
wire clk, rst;
always @ (posedge clk or posedge ~rst) //Error - Here ~rst
is not a simple variable
begin
...
end
```

Message Details

[SynthesisError] In the event expression with 'posedge' or 'negedge' qualifier, only simple variables are allowed

Severity

SynthesisError

SYNTH_168

Both edges of the same variable are not allowed in the event control list.

Language

Verilog

Rule Description

For an `always` block, it is illegal to specify a variable as both a `posedge` and `negedge`.

For example, the following code will violate:

```
wire clk, rst;
always @ (posedge clk or negedge rst or posedge rst)
begin
...
end
```

Message Details

`Both edges of the same variable (<variable>) are not allowed in the event control list`

Severity

SynthesisError

SYNTH_169

The use of non-constant bit-select expression as part of an edge control expression is not synthesizable.

Language

Verilog

Rule Description

If the bit-select index for an expression used in the `always` block sensitivity list is not constant, then such a construct is not synthesizable.

For example, the following code is not synthesizable:

```
wire i, clk, [2:0] rst;
always @ (posedge clk or negedge rst[i])
begin
...
end
```

Message Details

`Bit-select index for expression ( <expression> ) used in always block sensitivity list is not constant. Such a construct is not synthesizable`

Severity

SynthesisError

SYNTH_196

Task must not have event control statements for synthesis.

Language

Verilog

Message Details

```
Task should not have event control statements
```

Severity

SynthesisError

SYNTH_538

Synthesis error due to "Unterminated pragma block", an OFF pragma was encountered but the file has ended without any corresponding ON pragma

Language

VHDL

Rule description

SpyGlass generates this synthesis error to indicate the presence of an un-terminated block. This may happen if you forget to terminate a pragma block. For example, SpyGlass flags an error in the following case:

```
entity e is
end e;

architecture rtl of e is
begin
  -- pragma translate_off
end;
--end of file encountered before pragma block is
--terminated with a corresponding translate_on
```

Message Details

```
Un-terminated '<pragma-block>' block
```

Severity

SynthesisError

SYNTH_1001

Statement will be ignored (in context)

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that the following statements are not supported for synthesis and therefore ignored by SpyGlass for synthesis:

- Assertion statement
- Report statement
- Entity declaration statement

Message Details

```
<statement> will be ignored
```

Severity

SynthesisWarning

SYNTH_1002

Default initial value of signal/variable will be ignored if further assignment is found

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that default initial value of signal/variable will be ignored if further assignment is found. If no further assignment is found, that initial default value would be used.

Message Details

```
Default initial value of <signal | variable> will be ignored
```

Severity

SynthesisWarning

SYNTH_1003

Declaration types that should not be used in the present context are ignored for synthesis

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that the following declaration types are not supported for synthesis and thus ignored for synthesis.

1. Physical Type
2. Floating Point Type
3. File Type
4. Access Type

Message Details

```
<declaration> declaration will be ignored
```

Severity

SynthesisWarning

SYNTH_1004

Keyword TRANSPORT ignored in signal assignment

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that the keyword TRANSPORT would be ignored in signal assignment for synthesis. The right-hand side of a signal assignment may optionally specify a delay mechanism. A delay mechanism consisting of the reserved word TRANSPORT specifies that the delay associated with the first waveform element is to be construed as transport delay.

Message Details

```
Keyword TRANSPORT ignored in signal assignment
```

Severity

SynthesisWarning

SYNTH_1006

Right operand of operator must be static and have a value, which is a power of 2

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the right operand of an operator must be static and must have a value which is a power of 2.

Message Details

```
Right operand of operator "<operator>" must be static and have
a value which is a power of 2
```

Severity

SynthesisError

SYNTH_1007

Statement inside a subprogram body is not permitted (in context) for synthesis.

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that a certain statement inside a subprogram body is not permitted from synthesis point of view.

For example:

```
procedure proc (in1,in2 : integer; signal clk : bit;
    output : out integer) is
  begin
    wait until clk'event and clk='1';
 -- Error : 'WAIT' statement inside a subprogram body is not
permitted.
    output:= in1+in2;
end ;
```

Message Details

```
<statement> statement inside a subprogram body is not permitted
```

Severity

SynthesisError

SYNTH_1008

Waveform in assignment statement having multiple waveform elements is not permitted

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that a waveform having multiple waveform elements is not permitted in the assignment statement and would not be synthesized.

For example:

```
signal c : BIT;

c <= '0','1' after 8 ns, '0' after 13 ns, '1' after 50 ns;
 -- Error : not synthesizable
```

Message Details

```
Waveform in assignment statement having multiple waveform
elements is not permitted
```

Severity

SynthesisError

SYNTH_1009

PORT/GENERIC is not permitted in the block statement header

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that PORT/GENERIC declaration is not permitted in the block statement header. If present, it would not be synthesized.

For example:

```
L1 : block
    port(x,y : IN bit; z : OUT bit);
 -- Error
    port map(a,b,c);
-- Error
  begin
    com1 : block
      port(x,y : IN bit; z : OUT bit);
      port map(a,b,c);
    begin
  end block com1;
end block L1;
```

Message Details

```
PORT/GENERIC is not permitted in the block statement header
```

Severity

SynthesisError

SYNTH_1010

At least one WAIT statement is needed within a LOOP/WHILE-LOOP statement

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that at least one WAIT statement is needed within a LOOP/WHILE-LOOP statement otherwise it will give synthesis error.

Message Details

```
At least one WAIT statement is needed within a LOOP/WHILE-LOOP statement
```

Severity

SynthesisError

SYNTH_1011

WAIT statement is not permitted inside a FOR-loop statement

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the WAIT statement is not permitted inside a FOR-loop statement otherwise it will give synthesis error.

For example:

```
entity loop is
    port(in1: integer;
      clk: bit;
    out1: buffer integer);

end loop;

architecture loop of loop is
    begin
      process
    begin
      out1<=in1;
    for i in 0 to 100 loop
      wait until clk'event and clk='1';
 -- Error
      out1<=out1+1;
    end loop;
      end process;
end;
```

Message Details

```
WAIT statement is not permitted inside a FOR-loop statement
```

Severity

SynthesisError

SYNTH_1012

More than one clock is not permitted inside a process

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that more than one clock is not permitted inside a process for synthesis.

For example:

```
entity two_clocks is
    port(in1:integer;
      clk1,clk2:bit;
    out1:out integer
      );
end two_clocks;

architecture two_clocks of two_clocks is
begin
    process
  begin
    wait until clk1'event and clk1='1';
-- Error
      out1<=in1;
    wait until not clk2'stable and clk2='1';
-- Error
      out1<=in1+2;
  end process;
end;
```

Message Details

```
More than one clock is not permitted inside a process
```

Severity

SynthesisError

Synthesis Rules (Analysis Stage)

SYNTH_1013

Same clock driving a process on two different edges is not permitted

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the same clock driving a process on two different edges is not permitted for synthesis. All wait statements in the same process must be triggered by the same clock edge.

For example:

```
entity ee is
    port(  i1, i2: in bit ;
      clk : in bit ;
      out : out bit );
end ee;

architecture aa of ee is
begin
    process begin
      wait until clk'event and clk = '1' ;
 -- Error : this wait statement is triggered by rising clock
edge
    out <= i1 ;
      wait until clk'event and clk = '0' ;
 -- Error : this wait statement is triggered by falling clock
edge
    out <= i2 ;
    end process;
end aa;
```

Message Details

```
Same clock driving a process on two different edges is not
```

permitted

Severity

SynthesisError

SYNTH_1014

Missing signals in a process sensitivity list may lead to a potential simulation mismatch

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that a reading signal which is not part of the process sensitivity list may lead to a potential simulation mismatch.

For example:

```
entity ee is
    port(a, b: in bit; z: out bit);

end ee;

architecture aa of ee is
    begin
      process(a) begin
    z <= a xor b;
-- Warning
      end process;
    end aa;
```

In the above example, the synthesizer synthesizes a XOR gate sensitive to both 'a' and 'b', while the simulator, in contrast, executes the process only when 'a' changes, since 'b' was left out of the sensitivity list. This can cause a simulation/synthesis mismatch.

Message Details

```
Reading signal <signal> which is not part of the process
sensitivity list may lead to a potential simulation mismatch
```

Severity

SynthesisError

Synthesis Rules (Analysis Stage)

SYNTH_1016

WAIT ON statement is not synthesizable

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the WAIT ON statement should be avoided because it is non-synthesizable.

Message Details

```
WAIT ON statement is not synthesizable
```

Severity

SynthesisError

SYNTH_1017

Reading signal which has been used as the clock is not permitted

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the reading signal which has been used as the clock is not permitted as it is not synthesizable.

Message Details

```
Reading signal <signal> which has been used as the clock is not permitted
```

Severity

SynthesisError

SYNTH_1018

Avoid use of unsupported constructs

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that the following unsupported constructs should be avoided for synthesis.

- ACCESS type
- FILE type
- FLOATING point type
- Linkage port
- Pre-defined type TIME
- Pre-defined type SEVERITY_LEVEL
- Incomplete Type Declaration
- Disconnection specification
- Attribute specification using OTHERS/ALL
- Allocator expression (using NEW)
- Signal declaration of kind BUS/REGISTER.

Message Details

```
<construct> is not supported
```

Severity

SynthesisWarning

SYNTH_1019

Use of pre-defined attribute is not supported (in context)

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that the use of following pre-defined attributes is not supported for synthesis and thus ignored for synthesis.

1. VAL
2. SUCC
3. PRED
4. LEFTOF
5. RIGHTOF
6. DELAYED
7. QUIET
8. TRANSACTION
9. ACTIVE
10. LAST_EVENT
11. LAST_ACTIVE
12. STRUCTURE
13. BEHAVIOR

Message Details

```
Use of pre-defined attribute <attribute> is not supported
```

Severity

SynthesisWarning

SYNTH_1020

Resolution function is not synthesizable (in context)

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that use of the resolution function should be avoided as it is not synthesizable.

Message Details

`Resolution function <function> is not synthesizable`

Severity

SynthesisError

SYNTH_1021

Left operand of operator "" should have a static value which is a power of 2**

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the left operand of the operator "**" should have a static value which is a power of 2. Only 2 can be raised to a non-constant power in routine "**".

Message Details

```
Left operand of operator "**" should have a static value which
is a power of 2
```

Severity

SynthesisError

SYNTH_1022

The indices used to denote the slice name size must be statically computable

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the indices used to denote the slice name size must be statically computable, else it would not be synthesized.

Message Details

```
The indices used to denote the slice name size must be
statically computable
```

Severity

SynthesisError

SYNTH_1023

IF statements should conform with synthesizable description styles

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that IF statements should conform to synthesizable description styles.

Message Details

```
The IF-statement does not conform with any synthesizable
description style
```

Severity

SynthesisError

SYNTH_1024

A resolution function should not be used before its body is defined

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the resolution function should not be used before its body is defined.

Message Details

```
Resolution function <function> should not be used before its body is defined
```

Severity

SynthesisError

SYNTH_1025

Potential clock should be part of the process sensitivity list

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that the potential clock should be a part of the process sensitivity list.

This warning message implicitly points to the mismatch in pre-synthesis and post-synthesis simulation behavior due to absence of clock signal in the sensitivity list of a process block.

For example:

```
entity ee is
    port(a,b : in bit;
    c : out bit);
end ee;

architecture aa of ee is
  begin
    process (b)
      begin
        if(a'event and a = '1') then
 -- Warning : potential clock 'a' should be part of process
sensitivity list.
        end if;
  end process;
end aa;
```

Message Details

```
Potential clock <clock> is not part of the process sensitivity
list
```

Severity

SynthesisWarning

SYNTH_1026

An unconstrained type port is not synthesizable

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that an unconstrained type port is not synthesizable. For synthesis, all values must have a fixed bit width.

For example:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity ee is
    port(a,b : in BIT_VECTOR;
-- Error : Port 'a' and 'b' are of unconstrained type
    c : out bit);
end ee;
```

Message Details

```
Port <port> of unconstrained type is not synthesizable
```

Severity

SynthesisError

SYNTH_1027

AFTER clause is ignored; may cause potential simulation mismatch

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that the AFTER clause may cause a potential simulation mismatch, thus SpyGlass is ignoring it. Signal assignment delays are not supported for synthesis.

For example:

```
entity transport is
    port(in1 : integer;
      in2 : integer;
      output : out integer);
end transport;


architecture transport1 of transport is

begin
    process(in1,in2)
  begin
      output<= transport in1 + in2 after 10 ns;
 -- Error
  end process;
end;
```

NOTE: *The SYNTH_1027 rule for VHDL/mixed designs and the W257 of the SpyGlass lint solution report the same warning. Therefore, when the W257 rule is enabled for VHDL/mixed designs, the W257 violations (if any) are reported and no SYNTH_1027 violations are reported in such cases.*

Message Details

```
AFTER clause is ignored, may cause potential simulation
mismatch
```

Severity

SynthesisWarning

SYNTH_1028

Array defined using enumeration type as index is not supported

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that if an array is defined using enumeration type as index, it is not supported for synthesis. It would be ignored for synthesis by SpyGlass.

Message Details

```
Array defined using enumeration type as index is not supported
```

Severity

SynthesisError

SYNTH_1029

Avoid unsynthesizable clocking styles

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the use of unsynthesizable clocking styles should be avoided.

Message Details

```
This clocking style is not synthesizable
```

Severity

SynthesisError

SYNTH_1030

Multiple wait on same clock in an asynchronous process is not supported

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that in an asynchronous process, multiple wait on the same clock is not supported/synthesizable.

Message Details

```
Multiple wait on same clock in asynchronous process is not
supported
```

Severity

SynthesisError

SYNTH_1031

Clock in concurrent statement other than process is not permitted

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the usage of clock in concurrent statement other than process should be avoided as it is not permitted.

Message Details

```
Clock in concurrent statement other than process is not
permitted
```

Severity

SynthesisError

SYNTH_1032

Clock in sub-program body is not supported

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the use of clock in sub-program body should be avoided as it is not supported/synthesizable.

Message Details

```
Clock in sub-program body is not supported
```

Severity

SynthesisError

SYNTH_1033

WAIT on clock should be the first executable statement of a process

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that WAIT on clock should be the first executable statement of a process.

Message Details

```
This wait on clock is not first executable statement of the process
```

Severity

SynthesisWarning

SYNTH_1034

Timeout expression in wait statement is not supported

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the timeout expression in a wait statement should be avoided as it is not supported for synthesis by SpyGlass.

```
library IEEE;
    use IEEE.std_logic_1164.all;

entity MUX is
    port (A, B, SEL : in    std_logic;
    Z         : out  std_logic);
end MUX;

architecture OK_2 of MUX is
  begin
    process
      begin
    wait on SEL until SEL'EVENT and SEL = '1' for 10 ns;
-- Error
    end process;
end OK_2;
```

Message Details

```
Timeout expression in wait statement is not supported
```

Severity

SynthesisError

SYNTH_1035

Avoid wait statements with invalid clocking patterns

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that wait statements with invalid clocking patterns should be avoided.

Message Details

```
This wait statement does not conform to any valid clocking
pattern
```

Severity

SynthesisError

SYNTH_1036

Default initial value of out/inout port will be ignored

Language

VHDL

Rule Description

SpyGlass generates this synthesis warning to indicate that the default initial value of an OUT/INOUT port will be ignored for synthesis.

Message Details

```
Default initial value of out/inout port will be ignored
```

Severity

SynthesisWarning

SYNTH_1037

Use of pre-defined attribute is not supported

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the use of pre-defined attribute is not supported for synthesis.

Message Details

```
Use of pre-defined attribute <attribute> is not supported
```

Severity

SynthesisError

SYNTH_1038

Avoid use of unsynthesizable types

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that use of the following unsynthesizable types should be avoided.

1. Time Type
2. Floating Point Type
3. File Type
4. Severity_Level Type
5. Access Type

For example, the SYNTH_1038 error is flagged in the highlighted portions of the following code:

```
use std.textio.all;
entity e is
  function f (s : string) return bit_vector is
    file m : text;
    variable l       : line;
    variable bool    : boolean;
    variable b      : string(2 downto 1);
  begin

  FILE_OPEN(m, S, read_mode);
  READLINE(m, l);
  READ(L => l, VALUE => b, good => bool);
  return "01";
  end f;
end e;

architecture a of e is
begin
end a;
```

To avoid this error, specify pragmas as shown below.

```
use std.textio.all;
entity e is
  function f (s : string) return bit_vector is
    -- pragma synthesis_off
    --NO ERROR FLAGGED DUE TO USAGE OF PRAGMA
    file m : text;
    variable l       : line;
    --  pragma synthesis_on
    variable bool    : boolean;
    variable b      : string(2 downto 1);
  begin
    --  pragma synthesis_off
    FILE_OPEN(m, S, read_mode);
    READLINE(m, l);
    READ(L => l, VALUE => b, good => bool);
    --  pragma synthesis_on
  return "01";
  end f;
end e;

architecture a of e is
begin
end a;
```

Message Details

`Use of <unsynthesizable-type> type is not synthesizable`

Severity

SynthesisError

Note

For non synthesizable types, such as access type, file type, and physical type, the *SYNTH_1038* violations are reported on declaration of these types, even if these types have not really been used in the code.

The **solution to this problem** is to specify pragmas surrounding such usage. For example, consider the following code:

```
use std.textio.all;
entity ent is
end entity;
architecture rtl of ent is
function conv_string( i:integer) return string is variable l:
line; begin write(l,i);
end conv_string;
begin
end architecture;
```

To fix the *SYNTH_1038* violation in the above code, enclose the `conv_string` function within the `translate_off` and `translate_on` pragmas.

SYNTH_1039

Multiple element waveforms are not supported

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that waveform in assignment statement having multiple waveform elements is not permitted and would not be synthesized.

For example:

```
signal sig_1 : BIT;
sig_1 <= '0','1' after 18 ns, '0' after 23 ns, '1' after 56
ns;
-- Error : not synthesizable
```

Message Details

```
Multiple element waveform is not supported
```

Severity

SynthesisError

SYNTH_1040

Sub-element association for formal must be complete

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that sub-element association for formal must be complete for synthesis.

Message Details

`Incomplete sub-element association for formal <formal>`

Severity

SynthesisError

SYNTH_1041

Illegal use of object declared in a region is ignored for synthesis

Language

VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that the illegal use of an object declared in a certain region (by context) is ignored for synthesis.

For example, if any identifier is declared first in a region ignored for synthesis, using TRANSLATE_ON/TRANSLATE_OFF or SYNTHESIS_ON/ SYNTHESIS_OFF pragmas, then it is a synthesis error to use that identifier outside the ON/OFF region.

Message Details

```
Invalid use of object <object> declared in a region ignored for synthesis
```

Severity

SynthesisError

SYNTH_1042

Empty string constants are not supported for synthesis

Language

VHDL

Rule Description

SpyGlass generates this synthesis error due to the usage of empty string constants in VHDL code.

For example, the SYNTH_1042 error is flagged in the highlighted portions of the following code:

```
entity TOP is
end entity TOP;


architecture ARCH of TOP is
  procedure TERMINATOR (in0 : bit; blkdata :   string:="")is
  begin
  end TERMINATOR;
  procedure TERMINATOR_2 (in0 : bit; blkdata :
  string:="") is
  begin
  end TERMINATOR_2;
  function TERMINATOR_func (in0 : bit; blkdata
  :string:="") return boolean is
  begin
    return true;
  end TERMINATOR_func;

  signal qntemp:bit;
  signal bool:boolean;
  signal out_blkdata:bit_vector(0 to 2);
begin
  TERMINATOR( QNTEMP,"");
  TERMINATOR_2(in0=>QNTEMP,blkdata=>"");
  bool<=TERMINATOR_func( QNTEMP,"");
end architecture ARCH;
```

You can avoid this synthesis error by using pragma.

Message Details

```
Empty string constants are not supported for synthesis
```

Severity

SynthesisError

SYNTH_1081

Only an edge-sensitive sensitivity list is permissible for always_ff blocks.

Language

Verilog

Rule Description

The always_ff procedural block requires that every signal in the sensitivity list is qualified with either posedge or negedge. This rule helps ensure that simulation results will match synthesis results.

Message Details

```
Only edge-sensitive list is permissible for always_ff
```

Severity

SynthesisError

SYNTH_1082

Multiple event control statements associated with always_ff are not permissible.

Language

Verilog

Rule Description

The always_ff block imposes the restriction that it contains one and only one event control. More than one event control in an always block is an error.

For example, an error would be issued for the following always_ff procedural block.

```
always_ff
begin
  @(posedge clk) out1 = in1;
  @(posedge clk) out2 = in1;
end
```

Message Details

```
Multiple event control statements associated with always_ff is
not permissible
```

Severity

SynthesisError

SYNTH_1084

parameter/localparam cannot be assigned a Time literal value.

Language

Verilog

Rule Description

Declaring a parameter or a localparam where the RHS value is a time literal is invalid.

```
parameter x = 2ns;
```

Message Details

```
parameter/localparam cannot be assigned a Time literal value ( <value> )
```

Severity

SynthesisError

SYNTH_1111

Union [unpacked] is not synthesizable.

Language

Verilog

Rule Description

A union only stores a single value, regardless of how many data type representations are in the union. To realize the storage of a union in hardware, all the members of the union must be stored as the same vector size using the same bit alignment. An unpacked union does not guarantee that each data type will be stored in the same way, and is therefore not synthesizable.

By default, a union is unpacked.

Message Details

```
Union [unpacked] is not synthesizable
```

Severity

SynthesisError

Elaboration Built-In Rules

Overview

This section covers all elaboration Built-In rules.

ELAB_3002

Design unit is not found in library

Description

The *ELAB_3002* rule reports a violation if you specify the name of a module (design unit) that does not exist in the library being used.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Design unit is not found in library`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Specify a correct module that exists in the library.

Example Code and/or Schematic

Consider the following example:

Verilog Module

```
########################################
module m1 (in1,out2);
input  [3:0] in1;
output [3:0] out2;
endmodule
########################################
```

VHDL Top Entity

```
########################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0));
end top;

architecture struct of top is
begin
I1 : entity work.m2 port map (in1,out1);
end struct;
########################################
```

In the above example, the *ELAB_3002* rule reports a violation because the `m2` module is not present in the library being used.

To fix this violation, specify the module name as `m1`.

Default Severity Label

ElaborationError

ELAB_3003

Design unit not found.

Language

Verilog, VHDL

Message Details

```
Design unit is not found in library
```

Severity

ElaborationError

ELAB_3502

Verilog module does not contain any time scale, but it is passing an integer to a generic of the type time

Description

The *ELAB_3502* rule reports a violation when a Verilog module passes an integer value to a time-type VHDL generic, but that Verilog module does not have a `timescale directive.

Language

Verilog, VHDL

Message Details

This rule reports the following message:

[ElaborationWarning] `Verilog module does not contain any time scale, however it is passing an integer to generic of type time`

Consequences of Not Fixing

If you do not specify any time scale, the default time scale of the simulation tool is used during simulation.

How to Debug and Fix

To fix this violation, use the `timescale` directive inside the Verilog module.

The `timescale` statement is used to define a time unit and precision for simulation.

Example Code and/or Schematic

Consider the following example:

VHDL entity

```
########################################
library ieee;
use ieee.std_logic_1164.all;
```

```
entity e1 is
generic (ID_TIME : time );
port (in1: in std_logic;
out1 : out std_logic);
end e1;

architecture struct of e1 is
begin
out1 <= in1;
end struct;
########################################
```

Verilog Top Module

```
########################################
module top (in1,out1);
input in1;
output out1;
e1 #(.ID_TIME(1)) inst(in1,out1);
endmodule
########################################
```

In the above example, the *ELAB_3502* rule reports a violation as no `timescale` directive is being used in the Verilog module, `top`.

Default Severity Label

ElaborationWarning

ELAB_3503

Time scale unit inside Verilog module is not allowable unit in VHDL

Language

Verilog, VHDL

Rule Description

SpyGlass generates this elaboration warning to indicate that SpyGlass software has encountered an internal error. Please contact SpyGlass Support (*spyglass_support@synopsys.com*) for advice on how to proceed.

Message Details

```
Time scale unit inside Verilog module is not allowable unit in VHDL
```

Severity

ElaborationWarning

ELAB_3504

Index value is out of range.

Language

Verilog, VHDL

Message Details

```
Index value is out of range for (<object>)
```

Severity

ElaborationWarning

ELAB_3505

Infinite recursion found in the cell clause of an instance

Rule Description

The *ELAB_3505* rule reports a violation if infinite recursion is found while searching for the master of an instance specified in a Verilog configuration.

Language

Verilog and VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationWarning] `Infinite recursion found in cell clause for instance '<instance-name>'`

Consequences of Not Fixing

If you do not fix this violation, the reported instance is not bound using the Verilog configuration rules. It is bound based on the default SpyGlass flow.

How to Debug and Fix

Remove the infinite recursion in the cell clause of the reported instance.

Example Code and/or Schematic

Consider the following example:

```
#################################################
 Top Configuration
#################################################
 config cfg1;
 design aLib.top;
 instance top.I1 use L1.test1;
 cell L1.test1 use L2.test1;
 cell L2.test1 use L1.test1;
 endconfig
```

```
##################################################
```

In the above example, an infinite recursion occurs due to the following steps specified for finding the master of the `top.I1` instance:

1. For the `top.I1` instance, the cell clause suggests finding the master in the `test1` master of the `L1` library.
2. In the `test1` master of the `L1` library, the cell clause suggests finding the master in the `test1` master of the `L2` library.
3. In the `test1` master of the `L2` library, the cell clause suggests finding the master back in the `test1` master of the `L1` library.

The above steps lead to infinite recursion. Therefore, the `top.I1` instance is not bound using the Verilog configuration rules. Instead, it is bound based on the default SpyGlass flow.

Default Severity Label

ElaborationWarning

ELAB_3506

Multiple top-level modules specified in the configuration used by an instance

Rule Description

The *ELAB_3506* rule reports a violation if multiple top-level designs are specified the design statement of the configuration used by an instance.

Language

Verilog and VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationWarning] Multiple tops cannot be specified in design statement of configuration '<configuration>' in Library '<library>' for instance '<instance>'

Consequences of Not Fixing

If you do not fix this violation, the reported instance is not bound using the Verilog configuration rules. It is bound based on the default SpyGlass flow.

How to Debug and Fix

Remove multiple top-level modules from the design statement of the configuration used by the reported instance.

Example Code and/or Schematic

Consider the following example:

```
##################################################
 Top Configuration
##################################################
 config cfg1;
 design aLib.top;
 instance top.I1 use bLib.cfg2:config;
 endconfig
##################################################
```

```
 Definition of Non-top explicit Configuration
################################################
 config cfg2;
 design L1.test1 L2.test1;
 default liblist L1 L2;
 endconfig
################################################
```

In the above example, the instance clause for the `top.I1` instance uses the explicit configuration `cfg2`. This clause suggests that for all the instances below the hierarchy of `top.I1`, the rules specified in `cfg2` should be used.

However, in `cfg2`, as multiple top-level modules are specified, the master of the `top.I1` instance cannot be found. Therefore, this instance is not bound using Verilog Configuration rules. Instead, it is bound by using the default SpyGlass flow.

Default Severity Label

ElaborationWarning

ELAB_3507

Different configuration rules specified for an instance and its parent

Rule Description

The *ELAB_3507* rule reports a violation if the configuration rules applied on an instance do not match with the configuration rules applied to the parent of that instance.

Language

Verilog and VHDL

Messages and Suggested Fix

This rule reports the following violation:

```
[ElaborationWarning] Selection clause cannot be applied to
instance '<instance>' using configuration '<config1>' for which
parent hierarchical instance '<parent-instance>' is using
explicit config
```

Consequences of Not Fixing

If you do not fix this violation, the reported instance is not bound using the Verilog configuration rules. It is bound based on the default SpyGlass flow.

How to Debug and Fix

Remove the selection clause of a different configuration from the instances under the hierarchy of the reported instance.

Example Code and/or Schematic

Consider the following example:

```
#####################################################
 Top Configuration
#####################################################
 config cfg1;
 design aLib.top;
```

```
 instance top.I1 use bLib.cfg2:config;
 instance top.I1.I2 liblist L2 L3;
 instance top.I1.I3.I4 liblist L4 L5;
 endconfig
################################################
 Definition of Non-top explicit Configuration
################################################
 config cfg2;
 design L1.test1;
 default liblist L1 L2;
 endconfig
################################################
```

In the above example, the `cfg2` configuration is specified for the `top.I1` instance. This indicates that all the instances under the hierarchy of `top.I1` should be bound using the configuration rules specified only in `cfg2`.

However, the `cfg1` configuration is applied for the instances under `top.I1`. Therefore, `top.I1` is not bound using the rules of the `cfg2` configuration. Instead, it is bound by using the default SpyGlass flow.

Default Severity Label

ElaborationWarning

ELAB_3508

Master of an instance cannot be found using the Verilog configuration rule

Rule Description

The *ELAB_3508* rule reports a violation if the master of an instance is not found in the Verilog configuration rule applied to the instance.

Language

Verilog and VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationWarning] Cannot find Master of instance '<instance>' using Verilog configuration using (<configuration-rule>) rule

Consequences of Not Fixing

If you do not fix this violation, the reported instance is not bound using the Verilog configuration rules. It is bound based on the default SpyGlass flow.

How to Debug and Fix

Specify the correct Verilog configuration so that the instance master is found in the Verilog configuration rule.

Example Code and/or Schematic

Consider the following example:

```
################################################
 Top Configuration
################################################
 config cfg1;
 design aLib.top;
 instance top.I1.I2 liblist L2 L3;
```

```
 endconfig
################################################
```

In the above example, the instance clause is applied to the `top.I1.I2` instance. However, no instance master is not found in the `L2` and `L3` libraries in this case.

Therefore, the `top.I1.I2` instance is not bound using the Verilog configuration rules. Instead, it is bound using the default SpyGlass flow.

Default Severity Label

ElaborationWarning

ELAB_3509

Same configuration recursively specified in an explicit configuration

Rule Description

The *ELAB_3509* rule reports a violation if the same configuration is recursively specified in an explicit configuration, possibly resulting in an infinite recursion.

Language

Verilog and VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationWarning] Infinite recursion found in explicit configuration for instance '<instance-name>'

Consequences of Not Fixing

If you do not fix this violation, the reported instance is not bound using the Verilog configuration rules. It is bound based on the default SpyGlass flow.

How to Debug and Fix

Update the code to remove the specification of recursive configuration.

Example Code and/or Schematic

Consider the following example:

```
#####################################################
 Top Configuration
#####################################################
 config cfg1;
 design aLib.top;
 instance top.I1.I2 use bLib.cfg2:config;
 endconfig
```

```
################################################
 Definition of non-top explicit configuration
################################################
 config cfg2;
 design L1.test;
 instance test.I1 use aLib.cfg1;
 endconfig
################################################
```

In the above example, in the `cfg1` configuration for the `top.I1.I2` instance, the explicit configuration `cfg2` is specified. However, in `cfg2` for the `test.I1` instance, the explicit configuration is specified which again suggests use of `cfg1`. This may lead to infinite recursion.

Therefore, the `test.I1` instance is not bound using Verilog Configuration rules. Instead, it is bound by using the default SpyGlass flow.

Default Severity Label

ElaborationWarning

ELAB_3510

Unconnected port in an instance

Description

The *ELAB_3510* rule reports a violation when a port in a Verilog instance (whose master is in VHDL) is unconnected.

During instantiation of a Verilog module, some ports may remain unconnected because of any of the following reasons:

- The port is not mentioned during instantiation, as shown in the following example:

```
e1 inst2 (.in1(in1),.out1(out1)) // Correct
e1 inst2 (.in1(in1))             // Incorrect
                                 //out1 port is missing
```

- The port is mentioned during instantiation, but it does not have a net connected to it, as shown in the following example:

```
e1 inst1 (.in1(in1),.out1(out1)) // Correct
e1 inst1 (.in1(in1),.out1())   // Incorrect
        //No connected net specified with the out1 port
```

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationWarning] Nothing is connected to port (<port>) in instance (<instance>)

Consequences of Not Fixing

If you do not fix this violation, the reported ports will remain unconnected. As a result, the design may not function as desired.

How to Debug and Fix

Check if you have intentionally left the reported port unconnected. If this was not intentional, add the missing connection.

Default Severity Label

ElaborationWarning

Example Code and/or Schematic

Consider the following example:

```
VHDL entity
#####################################################
library ieee;
use ieee.std_logic_1164.all;
entity e1 is
port (in1: in std_logic;
out1 : out std_logic);
end e1;

architecture struct of e1 is
begin
out1 <= in1;
end struct;
#####################################################


Verilog Top Module
#####################################################
module top (in1,out1);
input in1;
output out1;
e1 inst1 (.in1(in1),.out1());
e1 inst2 (.in1(in1));
endmodule
#####################################################
```

In the above example, the *ELAB_3510* rule reports a violation for the following instances:

- `inst1`: No object is connected to the `out1` port of the `inst1` instance.

- `inst2`: No object is connected to the `out1` port of the `inst2` instance.

ELAB_3511

The number of parameters/generics used in the instantiation of is greater than the number of generics/parameters defined in the master.

Language

Verilog, VHDL

Message Details

```
Too many connections (<num> more) for instance ( <instance> )
```

Severity

ElaborationWarning

ELAB_3512

Generic not found in entity.

Language

Verilog, VHDL

Message Details

```
Generic <generic> not found in entity <entity> (approx position
is <position>)
```

Severity

ElaborationWarning

ELAB_3513

Nothing is connected to input port of the component instance.

Description

The *ELAB_3513* rule reports a violation when a Verilog component is being instantiated in a VHDL design unit, but nothing is connected to the input port of the instantiated component.

NOTE: *This rule reports violations only for missing input ports and not output ports.*

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationWarning] Nothing is connected to input port (<input-port>) in instance (<instance>). Some simulators do not support it

Consequences of Not Fixing

If you do not fix this violation, the reported ports will remain unconnected. As a result, the design may not function as desired.

How to Debug and Fix

To fix this violation, add the missing ports in component declaration. You might also need to add ports during component instantiation.

Example Code and/or Schematic

Consider the following example:

```
Verilog module
##############################################
module m1 (in1,out1);
input in1;
output out1;
```

```
endmodule
##############################################

Top VHDL Entity
##############################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic;
out1 : out std_logic);
end top;
architecture struct of top is
component m1 is
end component;
begin
I1: m1 ;
end struct;
##############################################
```

In the above example, the *ELAB_3513* rule reports a violation because the `in1` input port in the Verilog module is not mentioned in the component declaration of `m1`.

NOTE: *As the ELAB_3513 violation appears only for input ports, therefore, no ELAB_3513 violation appears for the output port of m1 in this example.*

Default Severity Label

ElaborationWarning

ELAB_3514

Design unit(referenced through Foreign Attribute) is not found in library.

Language

Verilog, VHDL

Message Details

```
Design unit <design-unit> (referenced through Foreign
Attribute) is not found in library <library>
```

Severity

ElaborationWarning

ELAB_3515

Design unit (referenced through Foreign Attribute) is not found.

Language

Verilog, VHDL

Message Details

```
Design unit <design-unit> (referenced through Foreign
Attribute) is not found
```

Severity

ElaborationWarning

ELAB_3517

The master of instance with implicit port declaration is not found in Verilog Domain.

Language

Verilog, VHDL

Message Details

```
The master of instance (<instance>) with implicit port
declaration is not found in Verilog, which is not allowed.
Considering it as a black box
```

Severity

ElaborationWarning

ELAB_3552

Message does not exist.

Language

Verilog, VHDL

Message Details

`Message <message> does not exist`

Severity

ElaborationError

ELAB_3553

Message severity cannot be changed.

Language

Verilog, VHDL

Message Details

`Message <message> severity cannot be changed`

Severity

ElaborationError

ELAB_3554

Severity is unknown.

Language

Verilog, VHDL

Message Details

```
Severity <severity> is unknown
```

Severity

ElaborationError

ELAB_3557

The defparam-referenced parameter or generic does not exist in the design unit.

Description

The *ELAB_3557* rule reports a violation if a defparam-referenced parameter or a generic is not present in the design unit given in the hierarchy specified by `defparam`.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `The defparam referenced parameter/generic (<name>) doesn't exist in DU (<design-unit>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Specify a correct generic/ parameter in `defparam`.

Example Code and/or Schematic

Consider the following example:

VHDL entity

```
########################################
library ieee;
use ieee.std_logic_1164.all;
```

```
entity e1 is
generic (gen1 : natural:= 2);
port (in1: in std_logic;
out1 : out std_logic);
end e1;
architecture struct of e1 is
begin
out1 <= in1;
end struct;
########################################
```

Verilog Top Module

```
########################################
module top (in1,out1);
input in1;
output out1;
e1 inst1 (.in1(in1),.out1(out1));
defparam inst1.gen2 =4;
endmodule
########################################
```

In the above example, the *ELAB_3557* rule reports a violation because `defparam` is used for the `gen2` generic that is not defined on the master entity `e1`.

Default Severity Label

ElaborationError

ELAB_3558

The instance referenced by defparam doesn't exist in DU.

Language

Verilog, VHDL

Message Details

```
The defparam referenced instance (<instance>) doesn't exist in
DU [<design-unit>]
```

Severity

ElaborationError

ELAB_3559

Unmatched object type passed to parameter/generic in DU.

Language

Verilog, VHDL

Message Details

```
Unmatched object type passed to parameter/generic (<name>) in
DU [<design-unit>]
```

Severity

ElaborationError

ELAB_3565

Too many connections to an instance

Description

The *ELAB_3565* rule reports a violation if the number of connections on an instance is greater than the number of ports on the instance master.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] Too many connections (<num> more) for instance (<instance>)

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Check the master definition of the instance and specify correct number of ports on the master.

Alternatively, update the connections in the master instance such that the number of ports on the instance match with the number of ports on the instance master.

Example Code and/or Schematic

Consider the following example:

VHDL entity
```
################################################
```

```
library ieee;
use ieee.std_logic_1164.all;
entity e1 is
port (in1: in std_logic;
out1 : out std_logic);
end e1;
architecture struct of e1 is
begin
out1 <= in1;
end struct;
##############################################
```

Verilog Top Module

```
##############################################
module top (in1,out1,out2);
input in1;
output out1,out2;
e1 inst1 (in1,out1,out2);
endmodule
##############################################
```

In the above example, the *ELAB_3565* rule reports a violation because three connections are specified for the `inst1` instance while only two ports are defined on the master entity `e1`.

Default Severity Label

ElaborationError

ELAB_3566

Port not found in entity

Description

The *ELAB_3566* rule reports a violation when you make a connection to a named port, such as `.in1(in1)` on an instance, but that port name is not defined on the instance master.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Port <port> not found in entity <entity> (approx position is <position>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Check the ports mapped to the instance with the ports defined on the instance master. Then either define the missing port on the instance master, or remove the connection from instantiation.

Example Code and/or Schematic

Consider the following example:

VHDL entity

```
#######################################################
library ieee;
```

```
use ieee.std_logic_1164.all;
entity e1 is
port (in1: in std_logic;
out1 : out std_logic);
end e1;
architecture struct of e1 is
begin
out1 <= in1;
end struct;
#####################################################
```

Verilog Top Module

```
#####################################################
module top (in1,out1,out2);
input in1;
output out1,out2;
e1 inst1 (.in1(in1),.out1(out1),.out2(out2));
endmodule
#####################################################
```

In the above example, the *ELAB_3566* rule reports a violation because the connection to the named port `out2` is made, but `out2` is not defined on the master entity `e1`.

Default Severity Label

ElaborationError

ELAB_3569

Port type mismatch

Description

The *ELAB_3569* rule reports a violation when you try to connect a VHDL signal/port and a Verilog port of dissimilar types.

For example, a violation appears if you try to connect a VHDL signal of the `std_logic` type to a Verilog vector port.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Port type mismatch. Trying to associate VHDL port (<port1>) of type <type> to Verilog <type> port (<port2>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Check the master definition to know the correct port definition and then connect a VHDL signal of appropriate type.

Example Code and/or Schematic

Consider the following example:

Verilog Module

```
##################################################
```

```
module m1 (in1,out1);
input in1;
output  [1:0] out1;
endmodule
#################################################

Top VHDL Entity
#################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic;
out1 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
component m1 is
port (in1: in std_logic;
out1: out std_logic);
end component;
begin
I1 : m1 port map (in1,out1(0));
end struct;
#################################################
```

In the above example, the *ELAB_3569* rule reports a violation because the `out1` port is defined as a scalar port in the component declaration of `m1`. However, the `out1` port is defined as a vector port (`[1:0]`) on the master module `m1`.

Default Severity Label

ElaborationError

ELAB_3573

For example, you may be trying to connect a 1-bit Verilog signal to a port of type std_vector on a VHDL master.

Language

Verilog, VHDL

Rule Description

For example, you may be trying to connect a 1-bit Verilog signal to a port of type std_vector on a VHDL master.

Message Details

```
Type mismatch. Type of Verilog connection doesn't match type of port <type>
```

Severity

ElaborationError

Suggested fix

Check the master definition for the correct port definition and connect a Verilog signal of the appropriate type. Remember that not all VHDL types can be connected directly to Verilog - for example VHDL records cannot be connected directly (each simple component of the record must be connected separately).

ELAB_3574

Type mismatch. The type of Verilog connection does not match with the type of generic.

Description

The *ELAB_3574* rule reports a violation when you specify a string parameter in Verilog to match with an integer generic on a VHDL master.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationError] `Type mismatch. Type of Verilog connection doesn't match type of generic <generic>`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Check the master definition for the correct parameter/generic definition and use a Verilog parameter of an appropriate type.

Note that all VHDL types cannot be handled in Verilog, particularly structured (multicomponent) types.

Example Code and/or Schematic

Consider the following example:

`VHDL entity`

```
################################################
library ieee;
use ieee.std_logic_1164.all;
entity e1 is
generic (gen1 :integer :=2);
port (in1: in std_logic;
out1 : out std_logic);
end e1;
architecture struct of e1 is
begin
out1 <= in1;
end struct;
################################################
```

Top Verilog Module

```
################################################
module top (in1,out1);
input  in1;
output   out1;
e1 #(.gen1("ATRENTA"))inst1 (.in1(in1),.out1(out1));
endmodule
################################################
```

For the above example, the *ELAB_3574* rule reports a violation because a string value is passed to the `ID_WIDTH` parameter that is defined as an integer in the master entity `e1`.

Default Severity Label

ElaborationError

ELAB_3575

For example, you may have specified a VHDL string value for a generic map to a Verilog master in which the corresponding parameter is defined as an integer.

Language

Verilog, VHDL

Rule Description

For example, you may have specified a VHDL string value for a generic map to a Verilog master in which the corresponding parameter is defined as an integer.

Message Details

```
Type mismatch. Type of VHDL connection doesn't match type of
parameter <parameter>
```

Severity

ElaborationError

Suggested fix

Check the Verilog master to find the correct definition for the parameter, then modify the VHDL generic map appropriately.

ELAB_3576

The master entity or module has defined one or more ports that you are not referencing on the instance. This is not allowed in VHDL or at mixed language interfaces.

Language

Verilog, VHDL

Rule Description

The master entity or module has defined one or more ports that you are not referencing on the instance. This is not allowed in VHDL or at mixed language interfaces.

Message Details

`No actual specified for formal port (<port>) in instance (<instance>)`

Severity

ElaborationError

Suggested fix

Check the master definition and include a reference to each port on that master, in your instance definition.

ELAB_3577

Reference to an unknown port or a parameter/generic in an instance

Description

The *ELAB_3577* rule reports a violation when an instance contains a reference to a port or a parameter/generic that does not exist in the instance master.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Unknown formal identifier '<identifier>' in instance (<instance>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Due to width mismatch in module header and module definition, the reported identifier is not visible.

Therefore, make the width in the module header and module definition equal for the identifier to be visible.

Example Code and/or Schematic

Consider the following example:

Verilog module

```
################################################
module m1 (in1,out1[1:0]);
input  in1;
output  [2:0] out1;
endmodule
################################################
```

Top VHDL entity

```
################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic;
out1 : out std_logic_vector(2 downto 0));
end top;
architecture struct of top is
begin
I1 :entity work.m1 port map (in1=>in1,out1=>out1);
end struct;
################################################
```

In the above example, the *ELAB_3577* rule reports a violation because of width mismatch, `2([1:0])` and `3([2:0])`, of the `out1` output port in the module header and definition, respectively, for the `m1` module.

Due to this width mismatch of the `out1` port in module header and module definition, a new port is created with an internally generated name. The width of the generated port is similar to the `out1 [1:0]` port.

Default Severity Label

ElaborationError

ELAB_3580

Port size mismatch

Description

The *ELAB_3580* rule reports a violation if the width of a connected signal in an entity/module is different from width declared in the master of the entity/module.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Port size mismatch. actual:size <size1> mapped to formal (<formal>):size <size2>`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Check the signal width in the master definition and maintain that width in the entity/module.

Example Code and/or Schematic

Consider the following example:

VHDL entity

```
#####################################################
library ieee;
use ieee.std_logic_1164.all;
```

```
entity e1 is
port (in1: in std_logic;
out1 : out std_logic);
end e1;
architecture struct of e1 is
begin
out1 <= in1;
end struct;
################################################
```

Top Verilog Module

```
################################################
module top (in1,out1);
input [3:0] in1;
output [3:0]out1;
e1  inst1 (.in1(in1),.out1(out1));
endmodule
################################################
```

In the above example, the *ELAB_3580* rule reports a violation because of width mismatch between the `in1` port (width 4) defined in `top` and the `in1` port (width 1) defined in the master entity `e1`.

Default Severity Label

ElaborationError

ELAB_3581

For example, the master module has a pin defined as input [3:0] in1, and you have connected the character literal '1' to that pin in the VHDL instantiation of that module. The character literal has width 1, so cannot match the width of the Verilog vector port.

Language

Verilog, VHDL

Rule Description

For example, the master module has a pin defined as input [3:0] in1, and you have connected the character literal '1' to that pin in the VHDL instantiation of that module. The character literal has width 1, so cannot match the width of the Verilog vector port.

Message Details

```
Port type mismatch. Character literal '<literal>' mapped to
vector net (<net>)
```

Severity

ElaborationError

Suggested fix

Check the pin width in the master definition. If you want to connect a fixed value, use a string literal of the appropriate width (e.g. "0110").

ELAB_3582

Port type mismatch. String literal mapped to scalar.

Description

The *ELAB_3582* rule reports a violation when a scalar-type port in the master definition is mapped to a string literal in the master instance.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] Port type mismatch. String literal "<literal>" mapped to scalar net (<net>)

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Check the pin width in the master definition.

Example Code and/or Schematic

Consider the following example:

Verilog Module

```
################################################
module m1 (in1,out1);
input in1;
output [3:0]out1;
endmodule
```

```
################################################

Top VHDL Entity
################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic;
out1 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
begin
I1 :entity work.m1 port map ("1010",out1);
end struct;
################################################
```

For the above example, the *ELAB_3582* rule reports a violation because the string value `"1010"` is mapped to the scalar port `in1` defined in the master module `m1`.

Default Severity Label

ElaborationError

ELAB_3584

Port type mismatch. Character literal is not of the expected type (STD_LOGIC).

Description

The *ELAB_3584* rule reports a violation when a scalar-type port in the master definition is mapped to a string literal in the master instance.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Port type mismatch. Character literal '<literal>' is not of expected type STD_LOGIC as of net (<net>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Map the reported port with a defined `STD_LOGIC` type, such as '1'.

Example Code and/or Schematic

Consider the following example:

Verilog

```
################################################
module m1 (in1,out1);
input  in1;
output [3:0]out1;
```

```
endmodule
################################################

Top VHDL Entity
################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic;
out1 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
begin
I1 :entity work.m1 port map ('a',out1);
end struct;
################################################
```

In the above example, the *ELAB_3584* rule reports a violation as because the character literal `'a'` is mapped to the scalar input port `in1` defined on the master `m1`.

Default Severity Label

ElaborationError

ELAB_3585

Port type mismatch. String literal is not of the expected type, STD_LOGIC_VECTOR.

Description

The *ELAB_3585* rule reports a violation when a vector port in the master definition is mapped to a non-constant string literal in the master instance.

For example, a violation appears if you map the `"abc"` string literal with the Verilog port `input [3:0] in1`.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Port type mismatch. String literal "<literal>" is not of expected type STD_LOGIC_VECTOR as of net (<net>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Map the reported string literal with an `STD_LOGIC_VECTOR` type, such as `"1010"`.

Example Code and/or Schematic

Consider the following example:

```
Verilog module
#####################################################
module m1 (in1,out1);
input  [3:0] in1;
output [3:0] out1;
endmodule
#####################################################

Top VHDL entity
#####################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
begin
I1 :entity  work.m1 port map ("ATRENTA",out1);
end struct;
#####################################################
```

In the above example, the *ELAB_3585* rule reports a violation because the non-constant string `ATRENTA` is mapped to the `STD_LOGIC_VECTOR` type port where only constant values, such as 0000 and 1010 are allowed.

Default Severity Label

ElaborationError

ELAB_3586

Unknown port direction for Verilog port.

Language

Verilog, VHDL

Message Details

`Unknown port direction for Verilog port (<port>)`

Severity

ElaborationError

ELAB_3587

Unknown port direction for VHDL port.

Language

Verilog, VHDL

Message Details

`Unknown port direction for VHDL port (<port>)`

Severity

ElaborationError

ELAB_3588

Port direction mismatch. Input port of a master is used as an output port in the master instance.

Description

The *ELAB_3588* rule reports a violation if an input port defined in a master is used as an output port in the master instance.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Port direction mismatch. Trying to read output port (<port>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Change the direction of the reported port as input in the component declaration.

Example Code and/or Schematic

Consider the following example:

```
Verilog module
##################################################
module m1 (in1,out1);
input [3:0] in1;
```

```
output [3:0] out1;
endmodule
##################################################

Top VHDL entity
##################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (out1: out std_logic_vector(3 downto 0);
out2 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
component m1 is
port (in1: out std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0));
end component;
begin
I1 : m1 port map (out1,out2);
end struct;
##################################################
```

In the above example, the *ELAB_3588* rule reports a violation because the `in1` port is defined with the `out` direction in the component declaration. But, this port is defined with the `in` direction in the master module `m1`.

Default Severity Label

ElaborationError

ELAB_3589

Port direction mismatch. Trying to map an input port with an inout port.

Rule Description

The *ELAB_3589* rule reports a violation if an inout port defined in a master is used as an input port in the master instance.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Port direction mismatch. Trying to associate input port (<input-port>) to inout port (<inout-port>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Change the direction of the reported port as inout in the component declaration.

Example Code and/or Schematic

Consider the following example:

```
Verilog module
##################################################
module m1 (inout1,out1);
inout  [3:0] inout1;
```

```
output [3:0] out1;
endmodule
#################################################

Top VHDL entity
#################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic_vector(3 downto 0);
out2 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
component m1 is
port (inout1: in std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0));
end component;
begin
I1 : m1 port map (in1,out2);
end struct;
#################################################
```

In the above example, the *ELAB_3589* rule reports a violation because the `inout1` port is defined with the `in` direction in component declaration while the same port, `inout1`, is defined with the direction `inout` on the master module `m1`.

Default Severity Label

ElaborationError

ELAB_3590

Port direction mismatch. Trying to map an output port with an inout port.

Description

The *ELAB_3590* rule reports a violation if an output port defined in a master is used as an inout port in the master instance.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Port direction mismatch. Trying to associate output port (<output-port>) to inout port (<inout-port>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Change the direction of the reported port as output in the component declaration.

Example Code and/or Schematic

Consider the following example:

```
Verilog module
################################################
module m1 (inout1,out1);
inout  [3:0] inout1;
```

```
output [3:0] out1;
endmodule
##################################################
Top VHDL entity
##################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (out1: out std_logic_vector(3 downto 0);
out2 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
component m1 is
port (inout1: out std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0));
end component;
begin
I1 : m1 port map (out1,out2);
end struct;
##################################################
```

In the above example, the *ELAB_3590* rule reports a violation because the `inout1` port is defined with the `out` direction in component declaration while the same port, `inout1`, is defined with the `inout` direction in the master module `m1`.

Default Severity Label

ElaborationError

ELAB_3591

Port direction mismatch. Trying to map an output port with an input port.

Rule Description

The *ELAB_3591* rule reports a violation if an output port defined in a master is used as an input port in the master instance.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Port direction mismatch. Trying to write input port (<input-port>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Change the direction of the reported port as output in the component declaration.

Example Code and/or Schematic

Consider the following example:

```
Verilog module
##################################################
module m1 (out1,out2);
output  [3:0] out1;
```

```
output [3:0] out2;
endmodule
##################################################

Top VHDL entity
##################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
component m1 is
port (out1: in std_logic_vector(3 downto 0);
out2 : out std_logic_vector(3 downto 0));
end component;
begin
I1 : m1 port map (in1,out1);
end struct;
##################################################
```

In the above example, the *ELAB_3591* rule reports a violation because the `out1` port is defined with the `in` direction in component declaration while the same port, `out1`, is defined with the `out` direction on the master module `m1`.

Default Severity Label

ElaborationError

ELAB_3592

Component instance has one or more ports that are not defined on the corresponding entity / module definition.

Description

The *ELAB_3592* rule reports a violation if a component instance has one or more ports connected but such ports are not defined in the master entity or module.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] Port (<port>) is not present in module (<module>). No binding exists for instance <instance1> during elaboration of instance <instance2> - Mismatch between component "<component>" and it's module declaration in file <file> at line <line>

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider x as the top-level module that contains the instance y. The parent of y is z. The consequence of not fixing this violation will be that x and z will become a black box.

How to Debug and Fix

Check the master definition and remove the excess port connections from the instance definition.

Example Code and/or Schematic

Consider the following example:

```
Verilog Module
##############################
module m1 (in1,out1);
input in1;
output out1;
endmodule
##############################

Top VHDL entity
##############################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic;
out1 : out std_logic);
end top;
architecture struct of top is
component m1 is
port (in2: in std_logic;
out1 : out std_logic);
end component;
begin
I1 : m1 port map (in2 =>in1, out1 => out1);
end struct;
##############################
```

In the above example, the *ELAB_3592* rule reports a violation because the `in2` port is defined as an input port in the component declaration but `in2` is not defined on the master module `m1`.

Default Severity Label

ElaborationError

ELAB_3593

For example, if the Verilog port is defined as an input, it would be incorrect to pass as string value to the corresponding VHDL instance connection.

Language

Verilog, VHDL

Rule Description

For example, if the Verilog port is defined as an input, it would be incorrect to pass as string value to the corresponding VHDL instance connection.

Message Details

```
Invalid Type mapped to Verilog port (<port>)
```

Severity

ElaborationError

Suggested fix

Check the master definition and pass a corresponding value to the VHDL instance connection.

ELAB_3594

For example, you specified a parameter foo = 17, and passed that parameter to a VHDL master, in which the corresponding generic foo is defined as std_logic.

Language

Verilog, VHDL

Rule Description

For example, you specified a parameter foo = 17, and passed that parameter to a VHDL master, in which the corresponding generic foo is defined as std_logic.

Message Details

```
Out of range. Verilog value associated with VHDL generic/port
(<name>) of subtype (<subtype>) is out of range
```

Severity

ElaborationError

Suggested fix

Check the master definition and pass a corresponding value to the VHDL generic or instance connection.

ELAB_3595

No actual is specified for a formal generic in an instance

Description

The *ELAB_3595* rule reports a violation if an uninitialized generic in a master is not initialized in the master instance.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `No actual specified for formal generic (<generic>) in instance (<instance>)`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Either initialize the reported generic in the master definition, or, initialize it in the instantiation of the reported entity.

Example Code and/or Schematic

Consider the following example:

```
VHDL entity
##################################################
library ieee;
use ieee.std_logic_1164.all;
entity e1 is
```

```
generic (gen1 :integer );
port (in1: in std_logic_vector(gen1 downto 0);
out1 : out std_logic_vector(gen1 downto 0));
end e1;
architecture struct of e1 is
begin
out1 <= in1;
end struct;
##################################################


Top Verilog Module
##################################################
module top (in1,out1);
input [1:0] in1;
output   [1:0]out1;
e1 inst1 (.in1(in1),.out1(out1));
endmodule
##################################################
```

In the above example, the *ELAB_3595* rule reports a violation because the uninitialized generic, `gen1`, defined in the `e1` entity is not initialized in the `inst1` instance.

Default Severity Label

ElaborationError

ELAB_3596

Binding error. Instance is explicitly unbounded.

Description

The *ELAB_3596* rule reports a violation if an entity is explicitly defined as open.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Binding error. Instance (<instance>) is explicitly unbounded`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Do not use the configurations that make an instance explicitly unbounded.

Example Code and/or Schematic

Consider the following example:

VHDL

```
#######################################################
library ieee;
use ieee.std_logic_1164.all;
entity ent is
port (in1: in std_logic_vector(3 downto 0);
```

```
out1 : out std_logic_vector(3 downto 0));
end ent;
architecture struct of ent is
begin
out1<= in1;
end struct;
#######################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic_vector(3 downto 0);
out1,out2 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
component ent is
port (in1: in std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0));
end component;
begin
I1 : ent port map (in1,out1);
I2 : ent port map (in1,out2);
end struct;
configuration top_cfg of top is
for struct
for I1 : ent
use open;
end for;
end for;
end;
#######################################################
```

In the above example, the *ELAB_3596* rule reports a violation because the `I1` instance of the `ent` entity is defined as open using the `use` clause in `configuration`.

Default Severity Label

ElaborationError

ELAB_3597

Some constructs are not being supported across mixed language boundary.

Language

Verilog, VHDL

Rule Description

SpyGlass flags this elaboration error if some object is encountered in port mapping or parameter/generic mapping which is not supported by SpyGlass.

NOTE: *If you encounter this error, please send the code snippet, which is causing this error, to spyglass_support@synopsys.com.*

Currently, SpyGlass supports only some specific constructs at the Verilog-VHDL interface.

SpyGlass supports character literal, base literal, string literal, and constants. Port width of actual and formal must match.

Message Details

```
Unrecognized Object. Not supported by SPYGLASS till now
```

Severity

ElaborationError

ELAB_3598

Parameter is not present in module

Rule Description

The *ELAB_3598* rule reports a violation when a parameter is defined in a component declaration, but that parameter is not declared in the master module of the component.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] Parameter (<parameter>) is not present in module (<module>)

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Declare the reported parameter in the master module of the component.

Example Code and/or Schematic

Consider the following example:

```
Verilog module
##################################################
module m1 (in1,out2);
parameter param1 = 2;
input  [3:0] in1;
```

```
output [3:0] out2;
endmodule
##################################################

Top VHDL entity
##################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
component m1 is
generic (param1:integer := 4; param2:integer := 5);
port (in1: in std_logic_vector(3 downto 0);
out2 : out std_logic_vector(3 downto 0));
end component;
begin
I1 : m1 generic map (7) port map (in1,out1);
end struct;
##################################################
```

In the above example, the *ELAB_3598* rule reports a violation because `Param2` is defined in component declaration but it is not declared in the master module `m1`.

Default Severity Label

ElaborationError

ELAB_3600

Range can't be calculated.

Language

Verilog, VHDL

Rule Description

The range of a signal or other object is defined by a generic, for which no value has been specified.

Message Details

`Range can't be calculated`

Severity

ElaborationError

Suggested fix

Define a value for the generic in the VHDL.

ELAB_3601

Out of range. Actual during instantiation is out of range.

Description

The *ELAB_3601* rule reports a violation if the value of an object exceeds the range defined in master declaration.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `Out of range. Actual for (<object>) is out of range`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Change the range limit of the reported object in the master declaration, or modify the object value in the master instantiation so that the value falls within the range.

Example Code and/or Schematic

Consider the following example:

```
VHDL entity
#####################################################
library ieee;
use ieee.std_logic_1164.all;
```

```
entity e1 is
generic (gen1 :integer range 0 to 2 );
port (in1: in std_logic_vector(gen1 downto 0);
out1 : out std_logic_vector(gen1 downto 0));
end e1;
architecture struct of e1 is
begin
out1 <= in1;
end struct;
##################################################
Top Verilog Module
##################################################
module top (in1,out1);
input  [3:0]in1;
output  [3:0] out1;
e1  #(.gen1(3))inst1(.in1(in1),.out1(out1));
endmodule
##################################################
```

In the above example, the *ELAB_3601* rule reports a violation because `gen1` is assigned the value 3 in the `inst1` instance of the `e1` entity.

However, this value is out of range because in the `e1` entity, the range of `gen1` is declared from 0 to 2.

Default Severity Label

ElaborationError

ELAB_3602

Invalid connection type. Output port externally connected to a register type net.

Description

The *ELAB_3602* rule reports a violation when an output port on a VHDL entity is connected to a register (reg) net in a Verilog module. This is not a valid connection.

NOTE: *This violation is not reported in the System Verilog flow, when the* ***set_option enableSV yes*** *is specified in the project file.*

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] Invalid connection type. Output port (<port>) externally connected to register type net

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

In VHDL, there is no concept of a `reg` net. Therefore, there is no way to make a direct connection with such a net.

To fix this violation, perform the following steps:

1. Connect the output port of the VHDL instance with a Verilog wire net.
2. Connect that wire net through a buffer or an assign statement to the `reg` net.

Example Code and/or Schematic

Consider the following example:

```
VHDL entity
#####################################################
library ieee;
use ieee.std_logic_1164.all;
entity e1 is
port (in1: in std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0));
end e1;
architecture struct of e1 is
begin
out1<= in1;
end struct;
#####################################################


Top Verilog Module
#####################################################
module top (in1,out1);
input [3:0] in1;
output reg [3:0]out1;
e1 I1 (in1,out1);
endmodule
#####################################################
```

In the above example, the *ELAB_3602* rule reports a violation because in instantiation of the `e1` entity, the `out1` output port is connected to the `out1` output port of the `reg` type in the `top` module.

Default Severity Label

ElaborationError

ELAB_3603

You are referencing, from Verilog, a component in a VHDL logical library but the library cannot be found (or has not been defined).

Language

Verilog, VHDL

Rule Description

You are referencing, from Verilog, a component in a VHDL logical library but the library cannot be found (or has not been defined). For example, if you reference:

```
\mylib.cpu cpu(...)
```

Then the cpu entity should be found under the VHDL logical library mylib

Message Details

`Invalid logical library. Library (<library>) is not visible`

Severity

ElaborationError

Suggested fix

Make sure the component you are looking for has been compiled into a library and that library has been referenced in your analysis with the required logical name.

ELAB_3604

Constraint can't be set. Actual size is not divisible by formal size.

Language

Verilog, VHDL

Message Details

```
Constraint can't be set. Actual size is not divisible by formal
size
```

Severity

ElaborationError

ELAB_3605

Constraint can't be set. More than one unconstraint range.

Language

Verilog, VHDL

Rule Description

The instance is connecting objects with unconstrained ranges to the Verilog master. For example:

```
  type foo is array(natural range <>, natural range <>);
  ...
  entity X is port(signal input: in foo; signal output: out
foo);
```

There is no way for any tool to determine where the input signal will end and the output signal will start, when the port list is viewed as a list. Moreover, this is both unsynthesizable and cannot be connected to a Verilog master (since Verilog does not support a concept of unconstrained ranges)

Message Details

```
Constraint can't be set. More than one unconstraint range
```

Severity

ElaborationError

Suggested fix

Use fixed range signals only.

ELAB_3606

SystemVerilog tick literal is mapped to an unconstrained port.

Language

Verilog, VHDL

Rule Description

The *ELAB_3606* violation appears if a SystemVerilog tick literal is mapped to an unconstrained port.

Consider the following example:

```
################################################
        VHDL entity
################################################
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;
ENTITY test1 IS
PORT (
in8 : IN unsigned
);
END ENTITY test1;
architecture rtl of test1 is
begin
end rtl;


################################################
        Top Verilog Module
################################################
module test ();
generate
genvar i;
test1 t1('1);
endgenerate
endmodule
################################################
```

In the above example, the width of the VHDL entity port `in8` is not specified. Therefore, it is considered as unconstrained. During binding of the Verilog instance, the *ELAB_3606* violation appears because the width of the SystemVerilog tick literal depends on the size of the `in8` port, which is not defined.

Message Details

```
Constraint can't be set. SV Tick Literal expression <expr>
mapped to unconstraint port: <port-name>
```

Severity

ElaborationError

ELAB_3607

Non static expression used in a generic map

Description

The *ELAB_3607* rule reports a violation when a non static expression is used in a generic map of an object.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] Non static expression used in the generic map of (<object>)

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Define a value for the generic in VHDL.

Example Code and/or Schematic

Consider the following example:

```
Verilog module
##########################
module m1 (in1,out1);
input  in1;
output out1;
parameter pa =4;
```

```
endmodule
##########################

VHDL Top entity
##########################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic;
out1 : out std_logic);
end top;
architecture struct of top is
signal s1 : integer ;
begin
I1 :entity work.m1 generic map (s1) port map ('1',out1);
end struct;
##########################
```

In the above example, the *ELAB_3607* rule reports a violation because the `s1` signal, which is non-static, is mapped to the `pa` parameter of the `m1` module.

Default Severity Label

ElaborationError

ELAB_3608

An input port in a module is specified as open in the module instance

Description

The *ELAB_3608* rule reports a violation if an input port declared in a module is specified as open in the module instance.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] `OPEN is not a legal actual part of an element association for an input port`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Assign an initial value to the input in the component/entity declaration for the VHDL entity and in the component declaration for the verilog module.

Example Code and/or Schematic

Consider the following example:

```
Verilog module
#######################################################
module m1 (in1,out1);
input  in1;
```

```
output [3:0]out1;
endmodule
#######################################################

VHDL Top entity
#######################################################
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (in1: in std_logic;
out1 : out std_logic_vector(3 downto 0));
end top;
architecture struct of top is
begin
I1 : entity work.m1 port map (open,out1);
end struct;
#######################################################
```

In the above example, the *ELAB_3608* rule reports a violation because the `in1` input port of the `m1` module is specified as *open* for the `I1` instance in the `top` entity.

Default Severity Label

ElaborationError

ELAB_3609

You are trying to base a bit string (e.g. "1010") to a scalar port on a Verilog module

Language

Verilog, VHDL

Rule Description

You are trying to base a bit string (e.g. "1010") to a scalar port on a Verilog module

Message Details

`Port type mismatch. Bit string "<string>" mapped to scalar net (<net>)`

Severity

ElaborationError

Suggested fix

Check the master definition for the correct port definition and connect a character literal ('1' or '0') of the appropriate value.

ELAB_3610

You have directly referenced a UDP from VHDL. We do not support direct references to UDPs.

Language

Verilog, VHDL

Rule Description

You have directly referenced a UDP from VHDL. We do not support direct references to UDPs.

Message Details

`Imported Verilog unit (<unit>) is not a module`

Severity

ElaborationError

Suggested fix

Create a wrapper module around the UDP, then reference that module from VHDL.

ELAB_3611

Configurations are not supported for direct instantiation during synthesis.

Language

Verilog, VHDL

Message Details

```
Configurations are not supported for direct instantiation
during synthesis
```

Severity

ElaborationError

ELAB_3612

Infinite recursion occurred while searching for the master of an instance

Description

The *ELAB_3612* rule reports a violation when SpyGlass encounters an infinite recursion while searching for the master of an instance specified in Verilog configuration.

Language

Verilog, VHDL

Messages and Suggested Fix

The following message appears in case of an infinite recursion:

[ElaborationError] `Infinite recursion found in cell clause for instance '<instance-name>'`

Consequences of Not fixing

If you do not fix this violation, the reported instance is considered as a black box.

How to Debug and Fix

Avoid infinite recursion for the cell clause of the reported instance.

Example Code and/or Schematic

Consider the following example:

```
##################################################
 Top Configuration
##################################################
config cfg1;
design aLib.top;
instance top.I1 use L1.test1;
cell L1.test1 use L2.test1;
```

```
cell L2.test1 use L1.test1;
endconfig
#####################################################
```

In the above example, an infinite recursion occurs because of the following steps:

1. The following selection clause for the `top.I1` instance causes SpyGlass to find the master in the `L1` library and `test1` master:

   ```
   instance top.I1 use L1.test1;
   ```

2. The following `cell` clause for the `test1` master name and `L1` library causes SpyGlass to find the master in the `L2` library and `test1` master:

   ```
   cell L1.test1 use L2.test1;
   ```

3. The following cell clause for the `L2` library and `test1` master causes SpyGlass to find the master back in `L1`.

   ```
   cell L2.test1 use L1.test1;
   ```

Default Severity Label

ElaborationError

ELAB_3613

Multiple top-level designs cannot be specified in a design statement for a configuration that is used in an explicit configuration

Description

The *ELAB_3613* rule reports a violation if multiple top-level designs are specified in a configuration that is being explicitly used for an instance.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationError] `Multiple tops cannot be specified in design statement of configuration '<configuration-name>' in Library '<library-name>' for instance '<instance-name>'`

Consequences of Not fixing

If you do not fix this violation, the reported instance is considered as a black box.

How to Debug and Fix

Avoid specifying multiple top-level designs in the configuration that is being used for the reported instance.

Example Code and/or Schematic

Consider the following example:

```
##################################################
 Top Configuration
##################################################
 config cfg1;
 design aLib.top;
 instance top.I1 use bLib.cfg2:config;
```

```
 endconfig

################################################
 Definition of Non-top explicit Configuration
################################################
 config cfg2;
 design L1.test1 L2.test1;
 default liblist L1 L2;
 endconfig
################################################
```

In the above example, the instance clause for the `top.I1` instance uses the `cfg2` configuration (`instance top.I1 use bLib.cfg2:config`). This means that for all the instances below the hierarchy of `top.I1`, the configuration in `cfg2` should be used.

However, in `cfg2`, multiple top-level designs are specified. As a result, the master of the `top.I1` instance cannot be found because of multiple top-level designs.

Default Severity Label

ElaborationError

ELAB_3614

Different configurations for the instance and the hierarchy under that instance.

Description

The *ELAB_3614* rule reports a violation when a specific configuration is applied to an instance but different configuration rules are applied to the hierarchy under that instance.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationError] `Selection clause cannot be applied to instance '<instance-name>' using configuration '<configuration-name>' for which parent hierarchical instance '<instance-parent-name>' is using explicit config`

Consequences of Not fixing

If you do not fix this violation, the reported instance is considered as a black box.

How to Debug and Fix

Avoid specifying different configuration rules under the hierarchy for the reported instance.

Example Code and/or Schematic

Consider the following example:

```
#################################################
 Top Configuration
#################################################
 config cfg1;
 design aLib.top;
```

```
 instance top.I1 use bLib.cfg2:config;
 instance top.I1.I2 liblist L2 L3;
 instance top.I1.I3.I4 liblist L4 L5;
 endconfig

################################################
 Definition of Non-top explicit Configuration
################################################
 config cfg2;
 design L1.test1;
 default liblist L1 L2;
 endconfig
################################################
```

In the above example, the `cfg2` configuration is explicitly applied to the `top.I1` instance (`instance top.I1 use bLib.cfg2:config;`). However, the `cfg1` configuration is applied to all the instances under the hierarchy of the `top.Ins1` instance. This is not allowed.

Default Severity Label

ElaborationError

ELAB_3615

Master of the instance cannot be found in the Verilog configuration

Description

The *ELAB_3615* rule reports a violation when an instance is specified in a Verilog configuration, but the master of that instance does not exist in that configuration.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

```
[ElaborationError] Cannot find Master of instance
'<instance-name>' using Verilog configuration using
(<configuration-name>) rule
```

Consequences of Not fixing

If you do not fix this violation, the reported instance is considered as a black box.

How to Debug and Fix

Either replace the reported instance with the instance whose master exists in the Verilog configuration, or update the configuration to include the master of the reported instance.

Example Code and/or Schematic

Consider the following example:

```
#####################################################
 Top Configuration
#####################################################
 config cfg1;
 design aLib.top;
```

```
 instance top.I1.I2 liblist L2 L3;
 endconfig
################################################
```

In the above example, the `top.I1.I2` instance clause is specified. Now, if the master of this instance is not found in `liblist L2 L3`, the *ELAB_3615* rule will report a violation.

Default Severity Label

ElaborationError

ELAB_3616

Infinite recursion found in the explicit configuration for an instance

Description

The *ELAB_3616* rule reports a violation to indicate an infinite recursion caused by specifying a configuration recursively.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[ElaborationError] Infinite recursion found in explicit configuration for instance '<instance-name>'

Consequences of Not fixing

If you do not fix this violation, the reported instance is considered as a black box.

How to Debug and Fix

Remove the recursion in the configuration of the reported instance.

Example Code and/or Schematic

Consider the following example:

```
##################################################
 Top Configuration
##################################################
 config cfg1;
 design aLib.top;
 instance top.I1.I2 use bLib.cfg2:config;
 endconfig


##################################################
```

```
 Definition of non-top explicit configuration
################################################
 config cfg2;
 design L1.test;
 instance test.I1 use aLib.cfg1;
 endconfig
################################################
```

In the above example, explicit configuration `cfg2` is specified for the instance `top.I1.I2` in the configuration `cfg1`. However, in for the `test.I1` instance in `cfg2`, explicit configuration `cfg1` is specified for the `test.I1` instance. This leads to an infinite recursion.

Default Severity Label

ElaborationError

ELAB_3619

Incorrect width of port connection for array of module instance

Description

The *ELAB_3619* rule reports a violation for incorrect widths used in port connections.

Language

Verilog, VHDL

Messages and Suggested Fix

[ElaborationError] Incorrect width of port connection (<connection>) for array of module instance (<instance>) of module (<module>). Must be same or (<num>) times the width (<width>) of port (<port>)

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider x as the top-level module that contains the instance y. The parent of y is z. The consequence of not fixing this violation will be that x and z will become a black box.

How to Debug and Fix

Use correct port widths, as specified in port declarations, while making port connections.

Example Code and/or Schematic

Consider the following example:

VHDL entity

```
################################################
library ieee;
```

```
use ieee.std_logic_1164.all;
entity e1 is
port (in1: in std_logic_vector(1 downto 0);
out1 : out std_logic_vector(1 downto 0));
end e1;
architecture struct of e1 is
begin
out1 <= in1;
end struct;
#################################################

Top Verilog Module
#################################################
module top (in1,out1);
input [4:0]in1;
output [4:0] out1;
e1 inst1[1:0](.in1(in1),.out1(out1));
endmodule
#################################################
```

In the above example, the *ELAB_3619* rule reports a violation because the width of `in1` and `out1` is five. In this case, the port width must be same, that is, [2] or two times the port width, that is [2*2=4].

Default Severity Label

ElaborationError

SpyGlass Design Constraints Built-In Rules

Overview

SpyGlass provides the built-in rules for checking the user-specified SpyGlass Design Constraints or automatically-generated SpyGlass Design Constraints (from SpyGlass Waiver pragmas).

checkSGDC

Checks Design Constraints file for valid object specification.

The checkSGDC rule runs *checkSGDC_01*, *checkSGDC_02*, *checkSGDC_03*, and *checkSGDC_04* rules.

checkSGDC_01

Checks validity of current_design names specified in constraints file

Language

Verilog, VHDL

Rule Description

The checkSGDC_01 rule checks the user-specified or internally generated SpyGlass Design Constraints files for correct `current_design` specifications.

The checkSGDC_01 rule flags a message if the design unit specified with the `current_design` keyword does not exist in the design.

Message Details

The following message appears when the design unit *<du-name>* specified with the `current_design` keyword does not exist in the design:

```
current design <du-name> is non-existent in input design
```

Severity

Warning

checkSGDC_02

Checks the validity of constraints nodes specified in constraints file

The checkSGDC_02 rule group is the group of *Constraints-Specific Built-In Rules*.

checkSGDC_03

Checks whether constraints are specified for every top module.

Language

Verilog, VHDL

Rule Description

The checkSGDC_03 rule checks if any top module has a `current_design` specification, then all the top modules should have a `current_design` specification.

This ensures that constraints, if specified for at least one, are specified for all top modules.

However, the checkSGDC_03 rule does not flag for the case when none of the top modules have a `current_design` specification.

Message Details

The following message appears when the design unit *<du-name>* does not have a corresponding `current_design` specification when at least one other design unit has a corresponding `current_design` specification:

```
No SGDC constraints specified for top module '<du-name>'
```

Severity

Warning

checkSGDC_04

Checks if constraints are valid for non-top environments.

Language

Verilog, VHDL

Rule Description

The checkSGDC_04 rule flags constraint specifications of constraints that work only at the top-level and are specified for a design unit that is not effectively a top-level design unit.

A design unit is a top-level design unit when it is a top-level design unit in the design or has been specified to be a top-level design unit (using the `set_option top <du-name>` command in the project file).

The following example is valid only when design unit `top` is effectively a top-level design unit as the `test_mode` constraint is effective at the top-level only:

```
current_design top
  test_mode -name N1 -value 0
```

To specify such constraints for top-level design units, use the hierarchical names with respect to the top. For example, assume that top-level design unit `top` contains an instance `inst1` of design unit `lower`, and you want to specify a `test_mode` constraint on net `n1` in design unit `lower` for instance `inst1`. Then, the correct specification is as follows:

```
current_design top
  test_mode -name top.inst1.n1 -value 0
```

The following specification is incorrect:

```
current_design lower
  test_mode -name n1 -value 0
```

Message Details

The following message appears when the non-top design unit *<du-name>*

has been specified as the `current_design` in constraint *<constr>* which is effective only at the top-level:

```
SGDC command '<constr>' not supported for non-top
current_design '<du-name>'
```

Severity

Warning

checkSGDC_05

Checks if constraints are valid for non-top environments.

Language

Verilog, VHDL

Rule Description

The checkSGDC_05 rule checks if a constraint is specified in a non-top environment.

If a constraint is not effective in a non-top environment, that constraint is migrated to the top and violation is flagged. If such constraints are specified under top-level environment, they are used as it is. For example:

```
current_design "top"
  test_mode -name N1 -value 0
```

In the above example, the `test_mode` constraint specified under the top environment, `top`, will apply value 0 on port `N1` of module `top`.

Consider another example given below:

```
current_design "lower"
  test_mode -name n1 -value 0
```

Here, for the non-top environment, `lower`, the above command will be automatically translated to the following:

```
current_design top
  test_mode -name "top.loweri1.n1" -value 0
  test_mode -name "top.loweri2.n2" -value 0
```

Where, `loweri1` and `loweri2` are two instances of `lower` in top module, `top`.

Message Details

The following message appears when an SGDC command <*cmd*>, which is not supported for non-top current design <*design-name*>, is migrated to the top:

```
SGDC command '<cmd>' not supported for non-top current_design
'<des-name>' has been migrated to top
```

Severity

Info

checkSGDC_06

Reports non-existent scoped module

Language

Verilog, VHDL

Rule Description

The checkSGDC_06 rule flags when the scoped module either does not exist in the design or has never been instantiated under the current design module.

Message Details

The following message appears when the scoped module *<mod-name>* is never instantiated in the current design *<des-name>*:

```
Scoped module ',mod-name>' is never instantiated in current
design '<des-name>'
```

Severity

Fatal

checkSGDC_07

Reports if scoping specification is specified in a field when some other field is of type UNIQUE

Language

Verilog, VHDL

Rule Description

The checkSGDC_07 rule flags when scoping specification is specified in a field when some other field is of type, UNIQUE.

Message Details

The following message appears when scoping specification is specified in a field *<field>* when some other field is of type, UNIQUE:

```
Scoping is not allowed in field '<field>' as some other field
is of type UNIQUE
```

Severity

Fatal

checkSGDC_existence

Checks existence of design object specified in SGDC commands

Language

Verilog, VHDL

Rule Description

The checkSGDC_existence rule checks for existence of design objects specified in the SGDC commands. It reports the design objects that are not found in the design.

The problem normally occurs in the following situations:

- If given net, instance, port, sub-hierarchy module or instance is not present in the design, or
- Specified object in SGDC command field has type which is not allowed as per the field semantic, or
- Specified object has been given as a wild-card and it matched none of the design objects in the design

To fix issues reported by the checkSGDC_existence rule, you need to:

- Check for typing mistakes in the name of design objects and correct them, or
- Provide definition of missing modules, if any, or
- Correct synthesis errors, if any module in the hierarchy is not synthesizable, or
- Fix elaboration errors, if there are any elaboration errors in the hierarchy, or
- Specify a design object of a valid object type specified for a given field. For example, an instance name specified in "test_mode -name" field.

For example, in the following SGDC command:

```
current_design top
  clock_pin -name module.port
```

The command reports a violation if the module specified with the `-name` field of the `clock_pin` command does not exist as `module.port` in the design.

Message Details

An appropriate message will appear

Severity

Fatal

checkSGDC_value

SpyGlass Checks non-design objects specified in SGDC commands to ensure that these values confirm the command field semantic

Language

Verilog, VHDL

Rule Description

The checkSGDC_value rule checks the non-design objects for which the values do not conform to the command field semantic.

This rule validates the following type of field values:

- Basic data type values like integer, float, string, identifier.
- Enumeration type values to ensure that given value falls in enumeration range.
- File type field values to ensure that given file is read/write enabled.

Please note that all the checks mentioned above do not depend on the design.

To fix issue reported by the checkSGDC_value rule, check the valid value set specified by the rule message, and accordingly update the given values.

For example, consider the following SGDC command:

```
current_design top
    clock -name abc -fflimit 0
```

The command reports a violation message because the field `-fflimit` is allowed to take an integer value in range `1` to `INF` and the given value `0` does not fall in the indicated range.

The above error can be fixed by giving values as per the allowed integer range.

Message Details

`An appropriate message will appear`

Severity

Fatal

checkSGDC_wildcardMatch

Checks SGDC field values containing design objects specification in wildcard format

Language

Verilog, VHDL

Rule Description

The checkSGDC_wildcardMatch rule flags improper wildcard specification in the SGDC commands.

The problem normally occurs if given wildcard string matches too many strings where only a single match is expected.

To fix issues reported by the checkSGDC_wildcardMatch rule, you need to:

- Check for typing mistakes in the given wildcard string and correct them.
- Check the validity of the object type specified in the given field. An error can occur if an incorrect object type is specifies. For example, if an instance name is specified in the `test_mode -name` field.
- If only a single match is expected, check if wildcard matches too many objects. In such a case, prune the wildcard to match exactly one object.

For example, the following SGDC command will report a violation if wildcard value specified with the `-name` field of `clock_pin` matches more than one ports in design unit `module`

```
current_design top
  clock_pin -name module.<port>
```

where, `<port>` is a wild card pattern.

Message Details

`An appropriate message will appear.`

Severity

Fatal

checkSGDC_together05

Given fields cannot be specified together in same SGDC command

Language

Verilog, VHDL

Rule Description

The checkSGDC_together05 rule flags the fields that cannot be specified together in a SGDC command, if they are specified together.

For example, the following command will report a violation because the fields `-import` and `-file` cannot be specified together in the waive constraint.

```
waive -import waiver1.sgdc -file test.v
```

Message Details

The following message appears when a field *<field-name1>* of the constraint *<constraint-name>* cannot be specified with another field *<field-name2>*

```
Constraint 'constraint-name': Field(s) '<field-name1>' cannot be specified with field '<field-name2>'
```

Severity

Fatal

checkSGDC_together

Given fields should be specified together in a SGDC command

Language

Verilog, VHDL

Rule Description

The checkSGDC_together rule flags if the given field in an SGDC command has to be specified with some other field but other required field has not been specified.

For example, the following command will report a violation if the `-value` and `-pin` fields of the keeper constraint are not specified together.

```
keeper -pin P1 -value 0011
```

Message Details

The following message appears when a field <*field-name1*> of the constraint <*constraint-name*> is specified, but another field <*field-name2*> is missing.

```
Constraint 'constraint-name': Field '<field-name1>' can/should be specified with field(s) '<field-name2>' only
```

Severity

Info/Fatal/Warning

SGDC_Abstract09

Specify additional files manually along with the block SGDC

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract09* rule reports a violation if the block SGDC file specified by the `abstract_file` constraint refers to some additional files through any of the following specifications:

- `include <sgdc-file>`
- `sgdc -import <block-name> <block-constraint-file>`
- `sdc_data -type <sdc-file>`
- `power_data -format <cpf|upf> -file <cpf/upf-file>`
- `activity_data -format <vcd|fsdb|saif> -file <file>`

In such cases, package the additional files manually along with the block SGDC file.

Message Details

```
Block '<block-name>' SGDC file(s) refers additional files which
needs to be manually packaged along with the block-level SGDC
file(s) in directory '<directory-name>'
```

Severity

Warning

SGDC_power_data01

Check on field -value of SGDC command power_data

Language

Verilog, VHDL

Rule Description

Check on the `-value` field of SGDC command, power_data

Severity

Fatal

SGDC_sdcschema02

Checks presence of the sdc_data constraint.

Language

Verilog, VHDL

Rule Description

The SGDC_sdcschema02 rule flags if the `sdc_data` constraint required by some of the selected rules is not specified by any of the design constraints.

Message Details

The following message appears if the `sdc_data` constraint required by some of the selected rules is not specified by any of the design constraints:

```
sdc_data is required to run some of the selected rule(s).
```

Severity

Fatal

SGDC_sdcschema03

In the SGDC file, multiple SDC files are specified in different SDC schemas with identical values of "mode/corner options" or have no value specified for these options.

Language

Verilog, VHDL

Rule Description

The SGDC_sdcschema03 rule flags if in the SGDC file, multiple SDC files are specified in different SDC schemas with identical values of `mode/corner` options or if no value is specified for these options.

If multiple SDC files are specified for the same design in different SDC schemas without `mode/corner` option, all the SDC files are read but only the last `sdc_data` constraint is retained. Hence, any dependency on previous SDC files may be flagged as errors. In addition, no two schemas can have identical `mode/corner` options. If identical `mode/corner` options are found, only a single common `sdc_data` constraint should be present.

For example, SpyGlass flags an error in the following cases:

```
sdc_data -file bar1.sdc
sdc_data -file bar2.sdc
sdc_data -file bar3.sdc
// Violation for bar1.sdc, bar2.sdc, and bar3.sdc.
```

```
sdc_data -file bar1.sdc -mode func -corner best
sdc_data -file bar2.sdc -mode func -corner best
sdc_data -file bar3.sdc -mode func1 -corner best
// Violation for bar1.sdc and bar2.sdc.
```

```
sdc_data -file bar1.sdc -mode func -corner best
sdc_data -file bar2.sdc -mode func -corner best
sdc_data -file bar3.sdc -mode func -corner best
// Violation for bar1.sdc, bar2.sdc, and bar3.sdc.
```

The above specifications are not the correct ways to specify multiple SDC files in an SGDC file. The correct way to specify multiple SDC files is based on the following conditions:

- If the SDC files are dependent, specify all the files with a single `sdc_data` constraint as shown in the following example:

  ```
  sdc_data -file bar1.sdc bar2.sdc
  ```

- If the SDC files are independent, specify each file in a different `sdc_data` constraint with its respective mode as shown in the following example:

  ```
  sdc_data -file bar1.sdc -mode test -corner best
  sdc_data -file bar2.sdc -mode func -corner typical
  ```

Message Details

The following message appears if in the SGDC file, multiple SDC files are specified in different SDC schemas with identical values of `mode`/`corner` options:

```
Incorrect way of specifying multiple SDC files for the same design in different SDC schemas due to use of identical values of mode/corner options
```

The following message appears if in the SGDC file, multiple SDC files are specified in different SDC schemas due to the absence of `mode` and `corner` options.

```
Incorrect way of specifying multiple SDC files for the same design in different SDC schemas due to absence of mode and corner options.
```

```
The above messages are flagged for single line errors. However, if the same error is present in multiple lines, SpyGlass appends the following line to the above messages:
```

```
Also violation present in line nos: <line-numbers>
```

Where, *<line-numbers>* refer to a space-separated list of line numbers in which error is flagged.

Severity

Warning

SGDCSTX_001

Failed to read SGDC file

Language

Verilog, VHDL

Rule Description

The SGDCSTX_001 rule flags SpyGlass Constraints files that cannot be read as the corresponding file does not exist at the specified location or the file does not have appropriate read permissions.

Message Details

The following message appears for a SpyGlass Design Constraints file *<file-name>* that cannot be read due to *<reason>*:

```
Failed to read SGDC file '<file-name>' (<reason>)
```

Severity

Syntax

SGDCSTX_002

Use of unknown SGDC command

Language

Verilog, VHDL

Rule Description

The SGDCSTX_002 rule flags unknown keywords used in the SpyGlass Design Constraints files.

The problem generally occurs in the following situations:

- The keyword is misspelled.
- The keyword is supplied in the wrong case.

  Please note that the SpyGlass Design Constraints keywords are case-sensitive.
- You have used a keyword normally supported by other tools (for example, a Synopsys Design Constraints keyword or a Tcl keyword) but is not supported by SpyGlass.

Message Details

The following message appears for an unknown keyword *<keyword>* encountered in a SpyGlass Design Constraints file

`Unknown SGDC command '<keyword>'`

Severity

Syntax

SGDCSTX_003

Use of an incorrect field name for an SGDC command

Language

Verilog, VHDL

Rule Description

The SGDCSTX_003 rule flags unknown fields used in the SpyGlass Design Constraints files.

The problem normally occurs when the field name is misspelled. For example, the `clock` keyword has a `-name` field that may be misspelled as `-names`.

For details of supported fields, refer to the respective product user guide.

Message Details

The following message appears for an unknown field *<field>* encountered for keyword *<keyword>* in a SpyGlass Design Constraints file:

```
Incorrect field '<field>' in SGDC command '<keyword>'
```

Severity

Syntax

SGDCSTX_004

A mandatory field of a SGDC command is not specified

Language

Verilog, VHDL

Rule Description

The SGDCSTX_004 rule flags keywords used in the SpyGlass Design Constraints files without their mandatory fields.

For example, the `asyncdisable` keyword requires both the `-name` and `-value` fields. If any of these fields is missed out, the SGDCSTX_004 rule flags a message.

For details on mandatory fields of different keywords, refer to the respective product user guide.

Message Details

The following message appears for a keyword *<keyword>* specified without its mandatory field *<field>* in a SpyGlass Design Constraints file:

```
Missing mandatory field '<field>' in SGDC command '<keyword>'
```

Severity

Syntax

SGDCSTX_005

Value of an SGDC command field is not of expected type

Language

Verilog, VHDL

Rule Description

The SGDCSTX_005 rule flags fields with incorrect value types in the SpyGlass Design Constraints files.

The value of a field can be either a single object or a list of objects of following types:

1. Strings: any character sequence (even integers or floats are also valid strings)
2. Integer numbers: a whole number like 345
3. Floating numbers: a real number like 1.456
4. Bit values: a sequence of 0 and 1 like 0100
5. Extended bit values: a sequence of 0,1,x, and z like 01xz0z1x

The type of field specified by user should match with the expected type.

NOTE: *You can specify an integer value for fields that expect floating-point numbers as float is a super set of integer. Similarly, you can specify any character sequence whether it is an integer, float, or anything else for string type fields.*

Message Details

The following message appears for a field *<field>* in a SpyGlass Design Constraints file that has been specified a value of type *<value1-type>* when a value of type *<value2-type>* was expected:

```
Field '<field>' data type mismatch (Expected '<value1-type>',
Actual '<value2-type>')
```

Severity

Syntax

SGDCSTX_006

An SGDC command expecting a single value has been passed multiple values.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_006 rule flags fields where a value list is specified when only a single value was expected.

Message Details

The following message appears for a field <*field*> of keyword <*keyword*> in a SpyGlass Design Constraints file that has been specified with a value list when only a single value was expected:

```
Multiple values not allowed for field '<field>' in SGDC command '<keyword>'
```

Severity

Syntax

SGDCSTX_007

SGDC command field cannot be repeated

Language

Verilog, VHDL

Rule Description

The SGDCSTX_007 rule flags repeated fields in the same directive.

You cannot repeat a field in a SpyGlass Design Constraints directive. For single valued fields, it does not make sense to repeat the field. For multi-valued fields, you can specify all the values as a single list. For example, the `test_mode` keyword has the `-value` field that can accept a list of boolean bits. Thus, the following directive is illegal:

```
test_mode -name sig -value 0101 -value 1101
```

The above directive can be correctly written as follows:

```
test_mode -name sig -value 0101 1101
```

Message Details

The following message appears for a field *<field>* of keyword *<keyword>* in a SpyGlass Design Constraints file that has been repeated in the same directive:

```
Field '<field>' in SGDC command '<keyword>' cannot be repeated
```

Severity

Syntax

SGDCSTX_008

Value missing for an argument of an SGDC command field

Description

The *SGDCSTX_008* rule reports a violation if an argument of an SGDC command is specified without a value.

Rule Exception

This rule does not check the following arguments:

- `-testclock` and `-sysclock` arguments of the `clock` constraint
- `-regexp` argument of the `waive` constraint

Language

Verilog, VHDL

Messages and Suggested Fix

The following message appears if no value is specified to the argument *<argument-name>* of the SGDC command *<SGDC-command>*:

[SYNTAX] Value missing for field '<argument-name>' in SGDC command '<SGDC-command>'

Consequences of Not Fixing

If you do not fix this violation, the reported constraint is ignored from SpyGlass analysis.

How to Debug and Fix

To fix this violation, specify a value to the reported argument.

Example Code and/or Schematic

Consider the following constraint:

```
clock -name clk -value
```

For the above constraint, the *SGDCSTX_008* rule reports a violation because no value is specified to the `-value` argument.

Default Severity Label

Syntax

SGDCSTX_009

Missing ending double quote for a quoted string.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_009 rule flags incorrect use of double quotes (“) in quoted strings.

Every quoted string must have both the starting and ending double quotes (“101001”, “hello”, and so on.). If a double quote is to be used in a string, it must be escaped with a backslash (\) as in “Fire the \”Laser\”!”.

NOTE: *A string can be extended to the next line using the backslash line continuation character provided both starting and ending double quotes are correctly provided.*

Message Details

The following message appears for a quoted string with missing ending quote:

```
Missing ending double quote for quoted string
```

Severity

Syntax

SGDCSTX_010

An SGDC command specified without a preceding current_design specification

Language

Verilog, VHDL

Rule Description

The SGDCSTX_010 rule flags design directives without corresponding `current_design` specification.

All the design-related SpyGlass Design Constraints directives must be specified after a `current_design` specification. Only the waiver constraints directives are specified without a `current_design` specification.

Also, the scope of a `current_design` specification is limited within its SpyGlass Design Constraints file. Thus, if you want to continue a set of directives to a new file, you must provide a `current_design` specification before the directives.

Message Details

The following message appears for a SpyGlass Design Constraints directive without the corresponding `current_design` specification:

```
Missing current_design specification for SGDC command
'<keyword>'
```

Severity

Syntax

SGDCSTX_011

current_design command specified without a value

Language

Verilog, VHDL

Rule Description

The SGDCSTX_011 rule flags `current_design` specifications without a design unit name.

The `current_design` specification affects the way SpyGlass Design Constraints directives under it are applied to the design. You must specify one and only one design unit name (generally the top-level module or entity-architecture name) with a `current_design` specification.

Message Details

The following message appears for a `current_design` specification without a design unit name:

```
Missing design unit in current_design specification
```

Severity

Syntax

SGDCSTX_012

current_design command may be specified only with one design unit at a time

Language

Verilog, VHDL

Rule Description

The SGDCSTX_012 rule flags `current_design` specifications with multiple design unit names.

The `current_design` specification affects the way SpyGlass Design Constraints directives under it are applied to the design. You must specify one and only one design unit name (generally the top-level module or entity-architecture name) with a `current_design` specification.

Message Details

The following message appears for a `current_design` specification with multiple design unit names:

```
Multiple design units in current_design specification
```

Severity

Syntax

SGDCSTX_013

Only spaces are allowed as separators for list of values for a field.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_013 rule flags comma-separated field value lists.

Commas are not allowed as value list separators in SpyGlass Design Constraints directives. Only spaces are allowed as separators.

For example, the following directive is illegal:

```
test_mode -name top.din -value 1001,1000,0101
```

The above directive can be correctly written as follows:

```
test_mode -name top.din -value 1001 1000 0101
```

Message Details

The following message appears for a field *<field>* of keyword *<keyword>* in a SpyGlass Design Constraints file that has a comma-separated value list:

```
Only space-separated value list supported for field '<field>'
in SGDC command '<keyword>'
```

Severity

Syntax

SGDCSTX_014

Unexpected end of file encountered

Language

Verilog, VHDL

Rule Description

The SGDCSTX_014 rule flags incomplete SpyGlass Design Constraints directive at the end of a SpyGlass Design Constraints file.

The SGDCSTX_014 rule flags cases where a SpyGlass Design Constraints file is incomplete that is, the end-of-file is encountered while SpyGlass was expecting more tokens for the current SpyGlass Design Constraints directive being processed.

Message Details

The following message appears for an incomplete SpyGlass Design Constraints directive at the end of a SpyGlass Design Constraints file:

```
Unexpected end of file
```

Severity

Syntax

SGDCSTX_015

SGDC Command does not have at least one required field.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_015 rule flags directives without at least one of the required fields.

Message Details

The following message appears for a keyword *<keyword>* in a SpyGlass Design Constraints file that does not have at least one of the required fields (*<field-name-list>*):

```
At least one of '<field-name-list>' must be specified in SGDC command '<keyword>'
```

Severity

Syntax

SGDCSTX_016

An incorrect regular expression is specified in 'waive' command.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_016 rule flags `waive` directives with incorrect regular expressions.

For the `waive` directives, field values are inferred as regular expressions if the `-regexp` field is also specified. The SGDCSTX_016 rule flags incorrect regular expressions in such cases. For example, the following directive has an incorrect regular expression:

```
waive -regexp -du "up{}" -severity Info
```

Here, the regular expression `"up{}"` is incorrect as there is no numeral inside curly brackets `{}`.

Message Details

The following message appears for a `waive` directive where the field *<field>* has an incorrect regular expression *<expr>* as its value:

```
Invalid regular expression '<expr>' specified for field
'<field>'
```

Severity

Syntax

SGDCSTX_018

SGDC command field values contain special characters and are not enclosed in double quotes.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_018 rule flags field values containing special characters without enclosing double quotes.

The following special characters must be enclosed in double quotes:

$	*	\	;	'	"
{	}	?	`	<	>

For the `waive` keyword, we recommended that you always enclose regular expression values in double quotes as regular expressions are likely to contain one of these special characters.

Message Details

The following message appears for a field whose value has one of the special characters without enclosing double quotes:

```
Field values containing special characters must be enclosed in double quotes
```

Severity

Syntax

SGDCSTX_019

Missing filename argument for "include" directive

Language

Verilog, VHDL

Rule Description

The SGDCSTX_019 rule flags `include` directives without corresponding file name specification.

SpyGlass allows inclusion of SpyGlass Design Constraints files in another SpyGlass Design Constraints file using the `include` directive as follows:

```
include <file-name>
```

Then, the included file is expanded inline in the current scope.

Message Details

The following message appears for an `include` directive without a file specified:

```
Missing filename argument for SGDC directive 'include'
```

Severity

Syntax

SGDCSTX_020

File specified using "include" directive in SGDC file could not be read

Language

Verilog, VHDL

Rule Description

The SGDCSTX_020 rule flags `include` directives when the corresponding file cannot be read.

SpyGlass allows inclusion of SpyGlass Design Constraints files in another SpyGlass Design Constraints file using the `include` directive as follows:

```
include <file-name>
```

Then, the included file is expanded inline in the current scope.

Message Details

The following message appears for an `include` directive where the corresponding file *<file-name>* that cannot be read due to *<reason>*:

```
Failed to read SGDC file '<file-name>' (Reason: <reason>)
```

Severity

Syntax

SGDCSTX_021

Including same SGDC file again recursively.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_021 rule flags recursively included SpyGlass Design Constraints files.

SpyGlass allows inclusion of SpyGlass Design Constraints files in another SpyGlass Design Constraints file using the `include` directive as follows:

```
include <file-name>
```

If a SpyGlass Design Constraints file includes itself, or a SpyGlass Design Constraints file includes another file, and that file includes the first file, then this situation would result in infinite inclusion of each file. SpyGlass detects such situation and reports an error.

Message Details

The following message appears for a SpyGlass Design Constraints file *<file-name>* that is included recursively:

```
Recursive inclusion of same file: '<file-name>'
```

Severity

Syntax

SGDCSTX_022

Extra value specified for Boolean type of argument

Language

Verilog, VHDL

Rule Description

The SGDCSTX_022 rule flags Boolean fields specified with a value.

Boolean fields must not be supplied a value in a SpyGlass Design Constraints file. The behavior of Boolean fields is that when specified, the corresponding feature is enabled and when not specified, the corresponding feature is disabled. Additional value is not required.

Message Details

The following message appears for a Boolean field *<field>* of keyword *<keyword>* in a SpyGlass Design Constraints file when it is specified with a value:

```
Extra value specified for Boolean type of argument '<field>' of constraint '<keyword>'
```

Severity

Syntax

SGDCSTX_023

Duplicate Domain Information

Language

Verilog, VHDL

Rule Description

The SGDCSTX_023 rule flags `clock` constraints that are specified with domain information in both formats.

You should specify the domain information by using the `-domain` field of the `clock` constraint in the following format:

```
clock -name <clk-name> -domain <domain-name> ...
```

Message Details

The following message appears for a `clock` constraint in a SpyGlass Design Constraints file that has domain information specified in both formats:

```
Domain name cannot be specified both as part of '-name' field, and with '-domain' field
```

Severity

Syntax

SGDCSTX_024

SGDC command having a UNIQUE type key field cannot be repeated with the same field value.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_024 rule flags constraint specifications where a KEY type field specified as UNIQUE type has been repeated with the same value.

If a KEY type field of a SpyGlass Design Constraint has been specified as a UNIQUE type, then the SpyGlass Design Constraint can be specified only once for the same field value.

Consider the case where the `clock` constraint has `-name` KEY type field registered as UNIQUE type. Then, the following two specifications for the object `top2.clock` under the same `current_design` result in an error message:

```
clock -name top2.clock -period 10 -edge 1 4 -edge 5 8 -edge 9 10
clock -name top2.clock -period 11 -edge 1 4 -edge 5 8 -edge 9 11
```

Message Details

The following message appears for a SpyGlass Design Constraint *<constr>* in a SpyGlass Design Constraints file when another specification of the same constraint exists for the same object *<obj-name>* with the same value of the field *<field-name>* at line *<num>* in SpyGlass Design Constraints file *<file-name>*:

```
'<constr>' constraint already specified for object '<obj-name>'
as value of UNIQUE type field '<field-name>' at file '<file-
name>', line '<num>'
```

Severity

Syntax

SGDCSTX_025

Regular expressions with delimiters, should be followed by a space.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_025 rule flags incorrect regular expressions specified in constraint specifications.

For an SGDC command having regular expressions in the form of `q/.../` or `m/.../`, the ending delimiter should be followed by a space.

For example, the value `m/a.*/b` is an invalid regular expression specification as the ending delimiter after letter `a` is followed by the letter `b` and not white space. The correct specification should be `m/a.*/`.

It is recommended that '/' be always be used as a delimiter. Other characters should be used if and only if the regular expression itself uses a '/' character. The valid delimiters are "!@%^&*;/~?<>+=|".

Message Details

The following message appears for a SpyGlass Design Constraint specification with incorrect regular expression *<expr>*:

```
Invalid regular expression specification '<expr>'
```

Severity

Syntax

SGDCSTX_026

Illegal SGDC variable name

Language

Verilog, VHDL

Rule Description

The SGDCSTX_026 rule flags illegal SDGC variable names.

An SGDC variable name should begin with an alphabet and can contain only alphanumeric characters and underscores.

Message Details

The following message appears for an SGDC variable that contains or starts with an unsupported character <*char*>:

```
Illegal SGDC variable name: '<char>' not supported or not allowed as first character
```

Severity

Syntax

SGDCSTX_027

SGDC variable value list should be enclosed within double quotes

Language

Verilog, VHDL

Rule Description

The SGDCSTX_027 rule flags SGDC variables with incorrect value list.

The value list of an SGDC variable must be a space-separated list of values enclosed in double quotes.

Message Details

The following message appears for an SGDC variable whose value list <value-list> is not enclosed in double quotes:

```
Illegal SGDC variable value: value list '<value-list>' should be enclosed in double quotes
```

Severity

Syntax

SGDCSTX_028

Undefined SGDC variable reference

Language

Verilog, VHDL

Rule Description

The SGDCSTX_028 rule flags undefined SGDC variables.

The SGDCSTX_028 rule flags the following cases:

- The SGDC variable is not defined but is referred.
- The SGDC variable is referred before it is defined.

Message Details

The following message appears for an undefined SGDC variable <*var*>:

`Undefined SGDC variable reference '<var>'`

Severity

Syntax

SGDCSTX_029

Incomplete SGDC variable declaration (name and/or value missing)

Language

Verilog, VHDL

Rule Description

The SGDCSTX_029 rule flags incompletely defined SGDC variables.

An SGDC variable definition requires both a variable name and a variable value. The SGDCSTX_029 rule flags cases where one or both are missing.

Message Details

The following message appears for an incompletely defined SGDC variable:

```
Incomplete SGDC variable declaration (name and/or value missing)
```

Severity

Syntax

SGDCSTX_030

Missing SGDC variable closing bracket

Language

Verilog, VHDL

Rule Description

The SGDCSTX_030 rule flags incorrect SGDC variable references.

An SGDC variable reference can be referred with or without curly brackets around the variable name. The SGDCSTX_030 rule flags cases where only one curly bracket is available.

Message Details

The following message appears for an incompletely enclosed SGDC variable reference:

```
Missing SGDC variable closing bracket
```

Severity

Syntax

SGDCSTX_031

Double quoted string should be followed by a space.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_031 rule flags double-quoted strings that are not followed by a space character.

For a value specified in double quotes, the ending quote should be followed by a space character to mark the end of the string.

For example, the following value is invalid as the ending double quote after module is followed by `.name` and not a space character:

```
'"module".name'
```

Message Details

The following message appears for an incorrect double-quoted field value:

```
Illegal field value specified (requires space after closing double quote)
```

Severity

Syntax

SGDCSTX_032

An invalid field is given for repeat-sequence string type fields.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_032 rule flags invalid repeat-sequence values.

The SGDC arguments that accept repeat-sequence values require the values to be specified in the following format:

<I*S>

Where S is a string to be repeated I (integer value) times.

The string itself can contain repeat sequence strings and spaces.

For example, the value <abc*01> is invalid as abc is not an integer. The value <5*0>*01> is invalid as the <> brackets are not matched. The value <2*01<5*0>> is valid and will be expanded to 01000000100000.

Message Details

The following message appears for an invalid repeat-sequence value <*expr*> specified for field <*name*>:

```
Invalid repeat-sequence expression '<expr>' specified for field
'<name>'
```

Severity

Syntax

SGDCSTX_033

Field '-file_line' expects file name and line number pairs for values.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_033 rule flags incorrect `-file_line` argument values in `waive` constraints.

The `-file_line` argument in `waive` constraint requires a pair of a file name followed by a line number as in the following example:

```
waive -file_line myFile.v 10
```

The SGDCSTX_033 rule flags when both values of the pair are not specified or the line number is not a non-zero integer value.

Message Details

The following message appears for an incorrect `-file_line` argument value in a `waive` constraint:

```
Field '-file_line' expects file name and line number pairs for values
```

Severity

Syntax

SGDCSTX_034

Field '-file_lineblock' expects file name, start line number, and end line number triplets for values.

Language

Verilog, VHDL

Rule Description

The SGDCSTX_034 rule flags incorrect `-file_lineblock` argument values in `waive` constraints.

The `-file_lineblock` argument in `waive` constraint requires a triplet of a file name followed by a starting line number and an ending line number as in the following example:

```
waive -file_lineblock myFile.v 10 20
```

The SGDCSTX_034 rule flags when all three values of the triplet are not specified, the line numbers are not a non-zero integer value, or the starting line number is greater than the ending line number.

Message Details

The following message appears for an incorrect `-file_lineblock` argument value in a `waive` constraint:

```
Field '-file_lineblock' expects file name, start line number, and end line number triplets for values
```

Severity

Syntax

SGDCSTX_035

If-else syntax error -- missing conditional expression after if/elsif

Language

Verilog, VHDL

Rule Description

The SGDCSTX_035 rule flags missing conditional expression after `if` or `elsif` statement.

Message Details

The following message appears at a location where the conditional expression is missing in `if` or `elsif` statement:

`Incomplete if-else statement: no conditional expression after '<stmt>'`

Where, *<stmt>* can be `if` or `elsif`.

Severity

Syntax

SGDCSTX_036

Invalid conditional expression in if-else statement

Language

Verilog, VHDL

Rule Description

The SGDCSTX_036 rule flags invalid conditional expression in an `if-else` statement.

The conditional expression in the `if-else` statement cannot be evaluated. Therefore, you need to ensure that the conditional expression does not contain undefined variables or unsupported operators. Currently, the following operators are supported:

!	==	!=	<
>	<=	>=	

Message Details

The following message appears if an invalid conditional expression *<expr>* is encountered in an `if-else` statement:

```
Invalid conditional expression '<expr>' in if-else statement
```

Severity

Syntax

SGDCSTX_037

If-Else syntax error -- missing then/else block

Language

Verilog, VHDL

Rule Description

The SGDCSTX_037 rule flags missing `then/else` block in `if-else` statement.

The `then/else` blocks should be always enclosed in curly brackets ({}). For example:

```
if {$a == "val1"} then {
  clock -name a -sysclock
}
else {
  clock -name b -sysclock
}
```

Message Details

The following message appears if `then/else` block is missing in `if-else` statement:

```
Incomplete if-else statement: no <then | else> after '<if | else>'
```

Severity

Syntax

SGDCSTX_038

If-else syntax error -- keyword used wrongly

Language

Verilog, VHDL

Rule Description

The SGDCSTX_038 rule flags if an `if-else` construct keyword is encountered where it was not expected.

Message Details

The following message appears if an `if-else` construct keyword is encountered where it was not expected:

```
'<keyword>' keyword encountered where it was not expected:
there may be problems in the if-else statements before this
```

Severity

Syntax

SGDCSTX_039

If-else syntax error in file -- EOF found before matching bracket

Language

Verilog, VHDL

Rule Description

The SGDCSTX_039 rule flags if EOF is found before the matching bracket of `if-else` statement.

Message Details

The following message appears if EOF is found before the matching bracket of `if-else` statement:

```
If-else syntax error in file -- EOF found before matching bracket
```

Severity

Syntax

SGDCWRN_101

Use of '$' without a '\' in double-quoted string.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_101 rule flags un-escaped use of $ character in a quoted string.

SpyGlass Design Constraints may support variable substitution in a future release. Then, use of $ character will have special meaning. Thus, we recommend that you use $ character in a quoted string with a backslash character to ensure future compatibility.

Message Details

The following message appears for a quoted string that has an un-escaped $ character:

```
'$' specified without a '\' in the quoted string
```

Severity

Warning

SGDCWRN_102

Rule name specified as a value to '-rule/-rules' in waiver constraint is not registered.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_102 rule flags unregistered rules specified with SpyGlass Waiver Constraints.

In the `-rules` field of `waive` constraint, you can specify names of the rules to be waived or specify special keywords to waive rules of certain rule types. If a specified rule is not registered or the keyword is not a supported keyword, the SGDCWRN_102 rule flags a message. Similarly, in the `-rule` field of `waive` constraint, you can specify the name of one rule to be waived. If that rule is not registered, the SGDCWRN_102 rule flags a message.

This problem occurs when a rule name or a keyword is misspelled or the rule is obsolete.

Message Details

The following message appears for an unregistered rule or an incorrect keyword *<name>* specified with the `waive` constraint:

```
Rule '<name>' in SGDC command 'waive' not registered
```

Severity

Warning

SGDCWRN_103

A useless waiver pragma to enable a rule for the code following the pragma, while the rule was already enabled.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_103 rule flags redundant `enable_block` waiver pragmas.

A rule is disabled for a block of design code using the `disable_block` and `enable_block` pragmas as follows:

```
--spyglass disable_block <rule-list>
-- design code
--spyglass enable_block <rule-list>
```

If you specify a redundant `enable_block` waiver pragma as follows, the SGDCWRN_103 rule flags a message:

```
--Code Block 1
--spyglass disable_block RULE1
--Code Block 2
--spyglass enable_block RULE1
--Code Block 3
--spyglass enable_block RULE1
--Code Block 4
```

The second `enable_block` pragma is redundant as the RULE1 rule is already enabled by the first `enable_block` pragma.

Message Details

The following message appears for a redundant `enable_block` waiver pragma for rule *<rule-name>*:

```
Trying to enable rule '<rule-name>' which is already enabled
```

Severity

Warning

SGDCWRN_104

Waiver pragma 'spyglass disable/disable_block/enable_block <rule-list>' specifying an unregistered rule in the rule list.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_104 rule flags unregistered rules specified with SpyGlass Waiver pragmas.

This problem occurs when the rule name is misspelled or the rule has been obsolete.

Message Details

The following message appears for an unregistered rule *<rule-name>* specified with a SpyGlass Waiver pragma:

```
Rule '<rule-name>' specified in waiver pragma not registered
```

Severity

Warning

SGDCWRN_105

"-scanshift" argument of "test_mode" constraint is of Boolean type, and should not be passed a value.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_105 rule flags `-scanshift` field of the `test_mode` constraint specified with a value.

The `-scanshift` field of the `test_mode` constraint is a Boolean field. Specifying the `-scanshift` field enables the feature and not specifying disables the feature.

If you supply a value with the `-scanshift` field, SpyGlass ignores the value and prints the SGDCWRN_105 rule message.

Message Details

The following message appears for a `test_mode` constraint where the `-scanshift` field is specified with a value 1 or 0:

```
-scanshift argument of test_mode constraint may not be used with a value
```

Severity

Warning

SGDCWRN_107

A '$' character should be escaped when not indicating a SGDC variable.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_107 rule flags the $ character used without escaping when it is not used to indicate an SGDC variable.

When '$' character is used in the normal sense, that is, it is not indicating an SGDC variable, it should be escaped using a backslash character.

Message Details

The following message appears at the location where the $ character is used without escaping when it is not indicating an SGDC variable:

```
Expected escaped '$' character
```

Severity

Warning

SGDCWRN_108

SGDC variable defined but not used

Language

Verilog, VHDL

Rule Description

The SGDCWRN_108 rule flags SGDC variables defined by not used.

Message Details

The following message appears at the location where an SGDC variable *<var-name>* is defined when it is not used later:

```
SGDC variable '<var-name>' defined but not used
```

Severity

Warning

SGDCWRN_109

SGDC variable is defined with a name that also corresponds to an environment variable.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_109 rule flags SGDC variables with same name as an environment variable.

It is possible to create an SGDC variable with same name (say `abc`) as that of an environment variable. However, the SGDC variable should be used as `$abc` and the environment variable should be used as `$::abc`.

Message Details

The following message appears at the location where an SGDC variable <*name*> is defined with same name as an environment variable:

```
SGDC local variable name '<name>' conflicts with environment
variable. Use $<name> and $::<name> to refer local and
environment variables respectively
```

Severity

Warning

SGDCWRN_110

Value of SG_OPERATING_MODE re-defined, ignoring current specification

Language

Verilog, VHDL

Rule Description

The SGDCWRN_110 rule flags if the value of the SG_OPERATING_MODE variable is being re-defined.

Message Details

The following message appears if the value of the SG_OPERATING_MODE variable is being re-defined when that value has already been defined in file <*file-name*> at line <*line-num*>:

```
SG_OPERATING_MODE is already defined at file '<file-name>',
line <line-num>. Ignoring the current specification
```

Severity

Info

SGDCWRN_111

Waiver pragma 'spyglass disable / disable_block / enable_block' specified with an empty rule list.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_111 rule flags SpyGlass Waiver pragmas with an empty rule list and no ALL keyword.

The syntax of a SpyGlass Waiver pragma is as follows:

```
--spyglass <pragma-name> <rule-list> | ALL
```

```
//spyglass <pragma-name> <rule-list> | ALL
```

Where *<pragma-name>* is one of the SpyGlass Waiver pragma keywords and *<rule-list>* is a list of SpyGlass rules. If the rule list is not specified and the ALL keyword is also not specified, the SGDCWRN_111 rule flags a message.

Message Details

The following message appears for a SpyGlass Waiver pragma with an empty rule list and no ALL keyword:

```
No rule specified in waiver pragma
```

Severity

Warning

SGDCWRN_112

Reports invalid variable usage in the SpyGlass disable pragmas

Language

Verilog, VHDL

Rule Description

The SGDCWRN_112 rule reports invalid variable usage in the following pragmas specified in SGDC or waiver files:

disable	disable_block	enable	enable_block

A variable can be considered invalid due to the following reasons:

- The variable is referred without being first set by using the `setvar` command.

  For example, consider the following command:

  ```
  //spyglass disable_block ab$cd
  ```

  In the above example, the `cd` variable is being referred but is not defined previously by using the `setvar` command.

- Curly brackets are not specified properly while referencing variables.

  For example, consider the following command:

  ```
  setvar myVar myValue
  //spyglass disable_block ${myVar}_abc
  ```

  In the above example, the final replaced value is the following:

  ```
  //spyglass disable_block myValue_abc
  ```

  For such referencing, both the opening and closing brackets are required. Else, SpyGlass reports the SGDCWRN_112 warning.

Message Details

```
Invalid variable (<variable-name>) specification in disable/
```

disable_block/enable/enable_block pragma

Severity

Warning

SGDCWRN_113

An SGDC-command argument registered with the UNIQUE type cannot be repeated with the same value.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_113 rule reports a violation if an SGDC command gets repeated with the same value to an argument that is registered as a `UNIQUE` type.

Such repetitions may occur in the following cases:

- Constraints scoping
- When a block-level command is migrated to the top level

Message Details

```
'<constraint-name>' constraint already specified for object
'<object-name>' which is of UNIQUE type field
'<argument-name>', at file '<file-name>', line '<line-num>'.
This command is ignored for this value
```

Severity

Warning

Example

Consider the case in which the `-name` argument of the `clock` constraint is registered as `UNIQUE`. Now, consider the following `clock` constraints:

Constraint specified in a block-level SGDC	`clock -name clk1 -period 10 -edge 1 4 -edge 5 8 -edge 9 10`
Constraint migrated from top level to the block-level SGDC	`clock -name clk1 -period 11 -edge 1 4 -edge 5 8 -edge 9 11`

In the above case, the `clock` constraint migrated from top level is for the

same object `clk1` for which the `clock` constraint is already defined in the block-level SGDC. Therefore, the SGDCWRN_113 rule reports a violation in this case.

SGDCWRN_114

Reports white-space character found at the end of line after '\'.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_114 rule reports the white space character appearing at the end of the line after '\'.

Message Details

The following message appears if a while space character is found at the end of the line after '\':

```
White-space character found at the end of line after '\'
```

Severity

Warning

SGDCWRN_115

An SGDC-command argument registered with the UNIQUE type cannot be repeated with the same value.

Language

Verilog, VHDL

Rule Description

The SGDCWRN_115 rule reports a violation if you repeat an SGDC command by assigning the same value to an argument that is registered as a UNIQUE type.

Message Details

```
'<constraint-name>' constraint already specified for object
'<object-name>' which is of UNIQUE type field
'<argument-name>', at file '<file-name>', line '<line-num>'.
This command is ignored for this value
```

Severity

Warning

Example

Consider the case in which the -name argument of the clock constraint is registered as UNIQUE. Now, consider the following clock constraints:

```
clock -name clk1 -period 10 -edge 1 4 -edge 5 8 -edge 9 10
clock -name clk1 -period 11 -edge 1 4 -edge 5 8 -edge 9 11
```

In the above case, the clock constraint is specified twice for the same object clk1. Therefore, the SGDCWRN_115 rule reports a violation in this case.

SGDCWRN_117

-tag field is not specified along with -add in clock constraint. Hence ignoring -tag

Language

Verilog, VHDL

Rule Description

The *SGDCWRN_117* rule reports a violation if the `-tag` argument is not specified with the `-add` argument of the `clock` constraint.

Message Details

```
-tag' field is not specified with '-add' field for clock
constraint '<clk-name>' of file '<file-name>' at line
'<line-num>' Hence ignoring this constraint
```

Severity

Warning

Example

Consider the following constraint:

```
clock -name Clk1 -period 10.0 -add
```

For the above constraint, the *SGDCWRN_117* rule reports a violation as the `-tag` argument is not specified with the `-add` argument.

SGDCWRN_118

-add is specified but there is no previous definition for the same logical clock. Hence -add field is ignored

Language

Verilog, VHDL

Rule Description

The *SGDCWRN_118* rule reports a violation if you specify the `-add` argument to the `clock` constraint but there is no previous definition for the clock specified in this constraint.

Message Details

```
'-add' field for the constraint '<clk-name>' of file
'<file-name>' at line '<line-num>' is ignored as there is no
previous definition of the same logical clock
```

Severity

Warning

Example

Consider the following specification:

```
clock -name Clk1 -period 10.0
clock -name Clk2 -perios 10.0 -add
```

For the above constraint specification, the *SGDCWRN_118* rule reports a violation as the `-add` argument is specified but there is no previous definition of the `clk2` clock.

SGDCWRN_120

Multiple clocks are applied on the same object with the same tag

Language

Verilog, VHDL

Rule Description

The *SGDCWRN_120* rule reports a violation if multiple clocks are applied on the same object with the same tag.

In such cases, the last constraint with the same tag on the same object is considered.

Message Details

```
constraint is already specified for object '<obj-name>' with
same tag '<tag-name>' of file '<file-name>', line '<line-num>'.
the earlier command is ignored for this value. to add another
constraint on the same object, kindly use different -tag field
```

Severity

Warning

Example

Consider the following specification:

```
clock -name Clk1 -period 10.0 -tag t1
clock -name Clk1 -period 10.0 -tag t1 -add
```

In the above case, the *SGDCWRN_120* rule reports a violation as the same tag `t1` is used for `clk1`.

SGDCWRN_121

Multiple clocks are applied on the same object with the same tag. Considering the last specification of the virtual clock.

Language

Verilog, VHDL

Rule Description

The *SGDCWRN_121* rule reports a violation if a `clock` constraint specified with a tag is a virtual clock and a `clock` constraint with the same tag is already defined.

In this case, only the last constraint with the same tag is considered. All the other constraints of the same tag are deleted.

Message Details

```
Ignored constraint specified for object '<obj-name>' and tag
'<tag-name>' at file '<file-name1>', line '<line-num1>', as
constraint with same tag is specified at file '<file-name2>',
line '<line-num2>'.
```

Severity

Warning

Example

Example 1

Consider the following constraints:

```
clock -name Clk1 -period 10.0 -tag t1
clock -tag t1
```

In the above example, the *SGDCWRN_121* rule reports a violation as the last constraint specification is a virtual clock (specified with the `t1` tag) and the same tag is used in the previous `clock` constraint.

In this case, SpyGlass considers the last constraint and deletes the first one.

Example 2

Consider the following constraints:

```
current_design  "top"

clock -name CLK1 -domain d0 -edge { "0.000000"
"5.000000"} -period 10 -tag t1

clock -name CLK2 -domain d0 -edge { "0.000000"
"5.000000"} -period 10 -tag t1

clock -name CLK3 -domain d0 -edge { "0.000000"
"5.000000"} -period 10 -tag t2

clock -domain d0 -edge { "0.000000" "5.000000"} -period
10 -tag t1
```

In the above example, the last constraint specification is for a virtual clock specified with the `t1` tag. However, the same tag is used in the first two constraint specifications. Therefore, the *SGDCWRN_121* rule reports a violation.

In this case, only the last constraint specification is considered while the first and second specifications are deleted.

SGDCWRN_122

Multiple clocks are applied on the same object. Considering the last specification on the same object.

Language

Verilog, VHDL

Rule Description

The *SGDCWRN_122* rule reports a violation if a `clock` constraint on an object (specified by the `-name` argument) is specified but a `clock` constraint is already defined without the `-tag` argument for the same object.

In such cases:

- Only last `clock` constraint is considered.
- The other constraints for the same object are deleted.

Message Details

```
constraint is already specified on the same object <obj-name>
without any 'tag' at file '<file-name1>', line '<line-num1>'.
Overriding it with the constraint specified at file '<file-
name2>', line '<line-num2>'.
```

Severity

Warning

Example

Consider the following constraints:

```
clock  -name CLK1
clock  -name CLK1 -domain d0 -edge { "0.000000"
 "5.000000"} -period 10 -tag t1 -add
```

In the above example, the *SGDCWRN_122* rule reports a violation as the last `clock` constraint for `CLK1` is specified but the same object `CLK1` is specified in the previous constraint without the `-tag` argument.

In this case, the last constraint is considered while the first constraint is deleted.

SGDCWRN_123

Missing -add argument for the same clock specified in multiple clock constraints

Language

Verilog, VHDL

Rule Description

The *SGDCWRN_123* rule reports a violation if the same clock is specified in multiple `clock` constraint specifications and the `-add` argument is not specified in the subsequent specifications.

In such cases, the only last `clock` constraint is considered.

Message Details

```
deleting constraint specified on the object '<obj-name>' of
file '<file-name>', line '<line-num>'
```

Severity

Warning

Example

Consider the following specification:

```
clock  -name q%\CLK1 % -domain d0  -edge {"0.000000"
"5.000000"}

-period 10 -tag ck_CLK1 clock  -name q%\CLK1 % -domain d0
-edge { "0.000000" "5.000000"} -period 10 -tag ck_CLK2 -add

clock  -name q%\CLK1 % -domain d0 -edge {"0.000000"
"5.000000"} -period 10 -tag ck_CLK3
```

In the above case, the same clock `q%\CLK1` is specified in all the clock constraint specifications, but the `-add` argument is not specified in the last specification. Therefore, the *SGDCWRN_123* rule reports a violation in this case.

SGDCWRN_125

Found unpaired in-line disable_block waiver pragmas

Language

Verilog, VHDL

Rule Description

The *SGDCWRN_125* rule reports a violation if unclosed `disable_block` waiver pragmas are found in the design.

For example, if a rule is disabled by using `disable_block <rulename>` and not enabled again in that file by using `enable_block <rulename>`, the *SGDCWRN_125* rule reports a warning message.

Message Details

```
Reached EOF. Un-paired block waiver pragma defined for rule
'<rule-name>'.
```

Severity

Warning

SGDCWRN_126

Multiple constraints are applied on same object. Overriding the earlier specification on same object

Description

The *SGDCWRN_126* rule reports a violation if a constraint is overridden by another constraint defined with the `-override` argument for the same object.

In such cases:

- Only last constraint is considered
- The earlier constraints for same objects are deleted
- The argument that can be overridden is specified at the time of constraint registration

Language

Verilog, VHDL

Message Details

The following message is reported:

[WARNING] '<overriding-constraint-name>' constraint specified with -override tag for '<overriding-argument-name>' field at file '<file-name1>', line '<line-num1>'. Overriding earlier constraint(s) '<overridden-constraint-name>' having same object(s) '<overridden-object-names>' for '<overridden-argument-name>' field at file '<file-name2>', line '<line-num2>'

Example

Consider the case in which the `-name` argument of the `set_case_analysis` constraint is registered as override field. Now, consider the following `set_case_analysis` constraints:

```
current_design "test"
set_case_analysis -name test.a[0] -value 0
set_case_analysis -name test.a[1:0] -value 1 -override
```

In the above example, the `set_case_analysis` constraint is specified twice for the same object `test.a[0]`. Therefore, the *SGDCWRN_126* rule reports the following violation in this case for the overridden constraint:

```
'set_case_analysis' constraint specified with -override tag for
'-name' field at file 'test.sgdc', line '3'. Overriding earlier
constraint(s) 'set_case_analysis' having same object(s)
'test.a[0]' for '-name' field at file 'test.sgdc', line '2'
```

Default Severity Label

Warning

SGDCINFO_201

Use of an obsolete SGDC command

Language

Verilog, VHDL

Rule Description

The SGDCINFO_201 rule flags obsolete SpyGlass Design Constraints keywords.

Some SpyGlass Design Constraints keywords become obsolete and are replaced by newer keywords. SpyGlass automatically replaces the obsolete keyword with the newer keyword. The SGDCINFO_201 rule flags such replacements.

For example, obsolete "testclock" command

```
testclock -name clk1 -value rtz
```

is replaced with newer command "clock -testclock"

```
clock -testclock -name clk1 -value rtz
```

While the obsolete keyword may be still supported for some time to ensure backward compatibility, you are expected to migrate to the newer keyword as soon as possible.

Message Details

The following message appears for an obsolete SpyGlass Design Constraints keyword <*keyword_old*> that has been automatically replaced by the newer keyword <*keyword_new*>:

```
Converting obsolete SGDC command '<keyword_old>' to
'<keyword_new>'
```

Severity

Info

SGDCINFO_202

Use of an ignored SGDC command

Language

Verilog, VHDL

The SGDCINFO_202 rule flags SpyGlass Design Constraints keywords that are no longer used.

Some SpyGlass Design Constraints keywords are no longer used by SpyGlass and hence are ignored. For example, the black box keyword is ignored:

```
blackbox -name myblackbox
```

While such SpyGlass Design Constraints keywords may still be supported for some time to ensure backward compatibility, you are expected to remove such specifications from the SpyGlass Design Constraints files.

Message Details

The following message appears for a SpyGlass Design Constraints keyword <*keyword*> that is no longer used:

```
SGDC command '<keyword>' is no longer used by SpyGlass, hence ignored
```

Severity

Info

SGDC_ungroup01

Check on field -all -inst and -small of SGDC command ungroup_cells

Language

Verilog, VHDL

Rule Description

The SGDC_ungroup01 rule reports a violation if none of the `-all`, `-inst`, and `-small` arguments of the `ungroup_cells` SGDC command is specified.

Severity

Fatal

SGDC_ungroup02

ungroup_cells option '-small' or '-level' specified for non-top design unit

Language

Verilog, VHDL

Rule Description

The SGDC_ungroup02 rule reports a violation to indicate that the `-small` or `-level` argument of the `ungroup_cells` SGDC command has been specified for a non top-level design unit.

Message Details

This rule reports the following message:

```
ungroup_cells option '<option-name>' specified for non-top
design unit '<du-name>'
```

SGDC_ungroup03

Reports if hierarchical instances are specified using -inst filed of ungroup_cells constraint

Language

Verilog, VHDL

Rule Description

The SGDC_ungroup03 rule reports a violation if hierarchical instances are specified by using the `-inst` argument of the `ungroup_cells` SGDC command.

Message Details

This rule reports the following message:

```
Hierarchical instance '<inst-name>' specified in -inst field
```

Severity

Warning

SGDC_ungroup04

Reports if design object specified does not exist in design.

Language

Verilog, VHDL

Rule Description

The SGDC_ungroup04 rule reports a violation, if a design object specified in an SGDC constraint does not exist in the design.

The reason for non-existence might be ungrouping.

Message Details

This rule reports the following message:

```
SGDC constraint '<constraint-name>' field '<field-name>' value
'<value>' specified does not exist. It might be dissolved
hierarchy.
```

Severity

Warning

Constraints-Specific Built-In Rules

Overview

The *checkSGDC_02* rule group contains constraints-specific built-in rules described in this chapter.

The following constraints-specific built-in rules are switched off by default:

- *SGDC_define_runflow01*
- *SGDC_define_runflow02*
- *SGDC_define_runflow03*
- *SGDC_define_runflow04*
- *SGDC_define_runflow05*

ReportGeneratedWaiverFiles

Reports the generated compatible waiver file for every input waiver file

Language

Verilog, VHDL

Rule Description

The ReportGeneratedWaiverFiles rule flags the path of generated compatible waiver file for every input waiver file specified. These generated waiver files can be used to waive messages with the current release.

Message Details

```
Output file '<output-file>' containing compatible waivers is
generated corresponding to input file '<input-file>'.
```

Severity

Info

SGDC_asyncdisable01

Existence check for constraint 'asyncdisable'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_asyncdisable02

Value check for constraint 'asyncdisable'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid event (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_assume_path01

Bit select or part select is not allowed for '-input' and '-output' fields of the 'assume_path'.

Language

Verilog, VHDL

Rule Description

Bit select or part select is not allowed for '-input' and '-output' fields of the 'assume_path'.

Message Details

```
'Bit select' or 'part select' is not allowed for '-input' and
'-output' fields of the 'assume_path' command
```

Severity

Fatal

SGDC_assume_path02

Existence check for constraint 'assume_path'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a module in the design.

Severity

Fatal

SGDC_assume_path03

Existence check for constraint 'assume_path'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-input' field does not exist as a port in the module specified in -name field.

Severity

Fatal

SGDC_assume_path04

Existence check for constraint 'assume_path'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-output' field does not exist as a port in the module specified in -name field.

Severity

Fatal

SGDC_balancedClock01

Existence check for constraint 'balancedClock'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT or a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_blackBox01

Existence check for constraint 'blackbox'

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUBMODULE in the current design.

Severity

Fatal

SGDC_breakpoint01

Existence check for '-name' field of constraint 'breakpoint'

Rule Description

A violation is reported if the argument specified with the '-name' field of the breakpoint constraint does not exist as a port or net in the current design.

Message Details

```
<net-name>'[Port + Net] not found on/within module '<du-name>'
```

Severity

Fatal

SGDC_bypass01

Existence check for constraint 'bypass'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE or an INSTANCE in the current design.

Severity

Fatal

SGDC_clock01

Existence check for constraint 'clock'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT or a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_clock02

Value check for constraint 'clock'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid CLOCK_PULSE (that is, rto/rtz).

Severity

Fatal

SGDC_clock03

Value check for constraint 'clock'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the value specified using the '-freq' field is not a valid NUMERIC.

Severity

Fatal

SGDC_clock04

Value check for constraint 'clock'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the value specified using the '-period' field is not a valid NUMERIC.

Severity

Fatal

SGDC_clock05

Value check for constraint 'clock'.

Language

Verilog, VHDL

Rule Description

'-edge/-waveform' specification of constraint 'clock' must be even number of values, increasing, non-repeating, in the range of 0.0 to period-specification (inclusive) and cannot be specified without '-period' specification.

If first value is 0.0, the last value cannot equal the period specification.

Message Details

A violation is reported if the '-edge/-waveform' field is not specified in this format.

```
'-waveform' (or '-edge') field of constraint 'clock' cannot be
specified without '-period' field
```

```
'-waveform' (or '-edge') specification of constraint 'clock'
must have even number of increasing and non-repetitive values
```

```
'-waveform' (or '-edge') specification of constraint 'clock'
must be in the range 0.0 to '-period' specification (inclusive)
```

```
Last '-waveform' (or '-edge') value in 'clock' constraint
should not be same as the '-period' value when the first '-
waveform' (or '-edge') value is 0.0
```

Severity

Fatal

SGDC_clock08

Flip-flop count specified for testclock, should be a non-zero positive integer.

Language

Verilog, VHDL

Rule Description

The flip-flop count specified with '-fflimit' in 'testclock' command, must be positive integer greater than zero. For example, the following is not allowed:

```
testclock top.clk -value 1011 -fflimit "-3.0"
```

Message Details

```
flip-flop count specified with '-fflimit' can only be a
positive integer
```

Severity

Fatal

SGDC_clock09

'-fflimit' specification is valid for 'testclock' only.

Language

Verilog, VHDL

Rule Description

This rule checks that flip-flop count (-fflimit) is specified for testclock only.

It flags a message if it is specified for system clock. For example, the following is not allowed:

```
clock top.clk -value 1011 -fflimit 3
```

Message Details

```
'-fflimit' specified for clock '<clock>', which is not a
testclock
```

Severity

Fatal

SGDC_clock_pin01

Existence check for constraint 'clock_pin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a 'module.port' in the current design.

Severity

Fatal

SGDC_clock_pin02

Value check for constraint 'clock_pin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_define_tag01

Existence check for constraint 'define_tag'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT, NET or HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_define_tag02

Value check for constraint 'define_tag'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_disabletiming01

Cell/instance specified in the -name argument of the disable_timing constraint is not a liberty cell/instance.

Language

Verilog, VHDL

Rule Description

The *SGDC_disabletiming01* rule reports a violation if the cell/instance specified in the `-name` argument of the `disable_timing` constraint is not a liberty cell/instance.

Message Details

```
Name '<cell/instance-name>' specified in -name of
disable_timing constraint is not a liberty cell. Constraint
would be ignored
```

Severity

Warning

SGDC_disabletiming02

Pin specified in the -from/-to argument of the disable_timing constraint does not exit in the cell/instance specified in the -name argument

Language

Verilog, VHDL

Rule Description

The *SGDC_disabletiming02* rule reports a violation if the pin specified in the `-from/-to` argument of the `disable_timing` constraint does not exit in the cell/instance specified in the `-name` argument of this constraint.

Message Details

```
Pin '<pin-name>' specified in <-from | -to> field of
disable_timing constraint is not a valid pin of <cell |
instance> '<cell/instance-name>' specified in -name field
```

Severity

Warning

SGDC_fifo01

Check to see that one of the five fields are given in 'fifo' constraints.

Language

Verilog, VHDL

Rule Description

A violation is reported if none of the five fields are given in 'fifo' constraints.

Severity

Fatal

SGDC_fifo02

Existence check for '-rd_data' field of constraint 'fifo'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the '-rd_data' field of 'fifo' constraint does not exist as a Net or Hierarchical Terminal in the current design.

Severity

Fatal

SGDC_fifo03

Existence check for '-wr_data' field of constraint 'fifo'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the '-wr_data' field of 'fifo' constraint does not exist as a Net or Hierarchical Terminal in the current design.

Severity

Fatal

SGDC_fifo04

Existence check for '-rd_ptr' field of constraint 'fifo'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the '-rd_ptr' field of 'fifo' constraint does not exist as a Net or Hierarchical Terminal in the current design.

Severity

Fatal

SGDC_fifo05

Existence check for '-wr_ptr' field of constraint 'fifo'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the '-wr_ptr' field of 'fifo' constraint does not exist as a Net or Hierarchical Terminal in the current design.

Severity

Fatal

SGDC_fifo06

Existence check for '-memory' field of constraint 'fifo'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the '-memory' field of 'fifo' constraint does not exist as a Net or Hierarchical Terminal or Module or Instance in the current design.

Severity

Fatal

SGDC_fifo07

Togetherness check for '-rd_data' field of constraint 'fifo'

Language

Verilog, VHDL

Rule Description

A violation is reported if the '-rd_data' field is not given with '-wr_data' field of 'fifo' constraint.

Severity

Fatal

SGDC_fifo08

Togetherness check for '-wr_data' field of constraint 'fifo'

Language

Verilog, VHDL

Rule Description

A violation is reported if the '-wr_data' field is not given with '-rd_data' field of 'fifo' constraint.

Severity

Fatal

SGDC_fifo09

Togetherness check for '-rd_ptr' field of constraint 'fifo'

Language

Verilog, VHDL

Rule Description

A violation is reported if the '-rd_ptr' field is not given with '-wr_ptr' field of 'fifo' constraint.

Severity

Fatal

SGDC_fifo10

Togetherness check for '-wr_ptr' field of constraint 'fifo'

Language

Verilog, VHDL

Rule Description

A violation is reported if the '-wr_ptr' field is not given with '-rd_ptr' field of 'fifo' constraint.

Severity

Fatal

SGDC_force_ta01

Existence check for constraint 'force_ta'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_force_ta02

Value check for constraint 'force_ta'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-value' field is not a valid controllability/observability.

Severity

Fatal

SGDC_force_ta03

Value check for field '-control' of constraint 'force_ta'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-control' field is not a valid controllability.

Severity

Fatal

SGDC_force_ta04

Value check for field '-observe' of constraint 'force_ta'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-observe' field is not a valid observability.

Severity

Fatal

SGDC_force_ta05

Check for '-control'/'-value'/'-observe' field of constraint 'force_ta'.

Language

Verilog, VHDL

Rule Description

A violation is reported if '-value' is specified with the either '-observe' or '-control' field or if none of '-value'/'-observe' /'-control' are specified.

Message Details

```
'-value' field cannot be specified with '-observe'/'-control'
field of 'force_ta' constraint
```

```
One of '-value'/'-observe' /'-control' fields must be specified
with 'force_ta' constraint
```

Severity

Fatal

SGDC_gatingcell01

Existence check for '-name' field of constraint 'gatingcell'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the `-name` field of the `gatingcell` constraint does not exist as a design module in the current design.

Severity

Fatal

SGDC_gatingcell02

Existence check for '-clkinTerm' field of constraint 'gatingcell'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the `-clkinTerm` field of the `gatingcell` constraint does not exist as a port in the current design.

Severity

Fatal

SGDC_gatingcell03

'Existence check for '-enTerm' field of constraint 'gatingcell'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the `-enTerm` field of the `gatingcell` constraint does not exist as a port in the current design.

Severity

Fatal

SGDC_gatingcell04

Existence check for '-clkoutTerm' field of constraint 'gatingcell'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the `-clkoutTerm` field of the `gatingcell` constraint does not exist as a port in the current design.

Severity

Fatal

SGDC_gatingcell05

Value check for '-enValue' field of constraint 'gatingcell'

Language

Verilog, VHDL

Rule Description

A violation is reported if the argument specified with the `-enValue` field of the `gatingcell` constraint is invalid. Valid values are 0 and 1.

Severity

Fatal

SGDC_initForBist01

Existence check for constraint 'initForBist'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_initForBist02

Value check for constraint 'initializeForBist'

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_keeper01

Existence check for constraint 'keeper'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE in the current design.

Severity

Fatal

SGDC_keeper02

Existence check for constraint 'keeper'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-pin' field does not exist as a PORT of a SUB-MODULE in the current design.

Severity

Fatal

SGDC_keeper03

Value check for constraint 'keeper'

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_libgroup01

Existence check for library name specified through -libname field

Language

Verilog, VHDL

Rule Description

Library name specified through the `-libname` field of the `define_library_group` constraint must be present in one of the gateslib/sglib.

Message details

```
Invalid library name '<lib-name>' specified in the '-libname'
field of 'define_library_group' constraint command
```

Severity

Fatal

SGDC_libgroup02

Existence check for library group name specified through -name field

Language

Verilog, VHDL

Rule Description

Library group name specified through the `-name` field of the `use_library_group` constraint must be already defined through the `define_library_group` constraint.

Message details

```
Invalid library group name '<lib-group-name>' specified in the
'-name' field of 'use_library_group' constraint command
```

Severity

Fatal

SGDC_libgroup03

No library group specified for the design

Language

Verilog, VHDL

Rule Description

The `use_library_group` constraint has not been specified for the top design units.

Message details

```
'use_library_group' constraint missing for top design unit
'<du-name>'
```

Severity

Fatal

SGDC_memorytype01

Existence check for constraint 'memorytype'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE in the current design.

Severity

Fatal

SGDC_memoryforce01

Existence check for constraint 'memoryforce'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_memoryforce02

Value check for constraint 'memoryforce'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_memoryreadpin01

Existence check for constraint 'memoryreadpin'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-memname' field does not exist as a SUBMODULE in the current design.

Severity

Fatal

SGDC_memoryreadpin02

Existence check for constraint 'memoryreadpin'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-readport' field does not exist as a PORT of the SUB-MODULE specified in the '-memname' field.

Severity

Fatal

SGDC_memoryreadpin03

Value check for constraint 'memoryreadpin'

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_memorywritedisable01

Existence check for constraint 'memorywritedisable'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_memorywritedisable02

Value check for constraint 'memorywritedisable'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_memorywritepin01

Existence check for constraint 'memorywritepin'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-memname' field does not exist as a SUBMODULE in the current design.

Severity

Fatal

SGDC_memorywritepin02

Existence check for constraint 'memorywritepin'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-writeport' field does not exist as a PORT of the SUB-MODULE specified in the '-memname' field.

Severity

Fatal

SGDC_memorywritepin03

Value check for constraint 'memorywritepin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_memorywritepin04

Uniqueness check for constraint 'memorywritepin'

Language

Verilog, VHDL

Rule Description

A violation is reported if there is more than one constraint with the same combination of values specified in the fields '-memname' and '-writeport'.

Severity

Fatal

SGDC_memory3s01

Existence check for constraint 'memory3s'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-memname' field does not exist as a SUBMODULE in the current design.

Severity

Fatal

SGDC_memory3s02

Existence check for constraint 'memory3s'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-enableport' field does not exist as a PORT in the module provided in '-memname' field.

Severity

Fatal

SGDC_memory3s03

Value check for constraint 'memory3s'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_moduleByPass01

Existence check for constraint 'moduleByPass'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE in the current design.

Severity

Fatal

SGDC_moduleByPass02

Existence check for constraint 'moduleByPass'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-bpin' field does not exist as a PORT of the module specified in the '-name' field.

Severity

Fatal

SGDC_moduleByPass03

Existence check for constraint 'moduleByPass'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-iport' field does not exist as a PORT of the module specified in the '-name' field.

Severity

Fatal

SGDC_moduleByPass04

Existence check for constraint 'moduleByPass'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-oport' field does not exist as a PORT of the module specified in the '-name' field.

Severity

Fatal

SGDC_moduleByPass05

Value check for constraint 'moduleByPass'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the value specified using the '-value' field is not a valid NUMERIC.

Severity

Fatal

SGDC_nofault01

Existence check for constraint 'nofault'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE or INSTANCE in the current design.

Severity

Fatal

SGDC_noScan01

Existence check for constraint 'noScan'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE or INSTANCE or NET or PORT in the module given in the '-du' field.

Severity

Fatal

SGDC_noScan02

Existence check for constraint 'noScan'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-du' field does not exist as a SUB-MODULE in the current design.

Severity

Fatal

SGDC_scan01

Existence check for constraint 'scan'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE or INSTANCE or NET or PORT in the module given in the '-du' field.

Severity

Fatal

SGDC_scan02

Existence check for constraint 'scan'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-du' field does not exist as a SUB-MODULE in the current design.

Severity

Fatal

SGDC_pullDown01

Existence check for constraint 'pullDown'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE in the current design.

Severity

Fatal

SGDC_pullDown02

Existence check for constraint 'pullDown'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-pin' field does not exist as a PORT of a SUB-MODULE in the current design.

Severity

Fatal

SGDC_pullDown03

Value check for constraint 'pullDown'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_pullUp01

Existence check for constraint 'pullUp'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE in the current design.

Severity

Fatal

SGDC_pullUp02

Existence check for constraint 'pullUp'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-pin' field does not exist as a PORT of a SUB-MODULE in the current design.

Severity

Fatal

SGDC_pullUp03

Value check for constraint 'pullUp'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_allowedPath01

Value check for -top field of constraint 'allowedPath'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the value specified using the '-top' field is not one of (yes | no).

Severity

Fatal

SGDC_allowedPath02

Value check for -tie field of constraint 'allowedPath'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the value specified using the '-tie' field is not one of (yes | no).

Severity

Fatal

SGDC_allowedPath03

Value check for -tie field of constraint 'allowedPath'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the value specified using the '-domain' field is not one of (public | private).

Severity

Fatal

SGDC_require_path01

Existence check for constraint 'require_path'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-from' field does not exist as a NET or a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_require_path02

Existence check for constraint 'require_path'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-to' field does not exist as a 'module.port' in the current design.

Severity

Fatal

SGDC_require_path03

Value check for constraint 'require_path'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-tag' field of 'require_path' command contains undefined tag name.

Severity

Fatal

SGDC_require_value01

Existence check for constraint 'require_value'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a NET or a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_require_value02

Value check for '-value' field of constraint 'require_value'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_require_value03

Value check for constraint 'require_value'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-tag' field of 'require_value' command contains undefined tag name.

Severity

Fatal

SGDC_reset01

Existence check for constraint 'reset'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a NET in the current design.

Severity

Fatal

SGDC_reset02

Uniqueness check for constraint 'reset'.

Language

Verilog, VHDL

Rule Description

A violation is reported if there is more than one constraint with the same combination of values specified in the fields '-name' and '-async'.

Severity

Fatal

SGDC_reset03

Uniqueness check for constraint 'reset'.

Language

Verilog, VHDL

Rule Description

A violation is reported if there is more than one constraint with the same combination of values specified in the fields '-name' and '-sync'.

Severity

Fatal

SGDC_reset04

Value check for constraint 'reset'

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_reset_pin01

Existence check for constraint 'reset_pin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a 'module.port' in the current design.

Severity

Fatal

SGDC_reset_pin02

Value check for '-value' field of constraint 'reset_pin'.

Language

Verilog, VHDL

A violation is reported if the event specified using the '-value' field is not a value from the pre-defined set = (0,1).

Severity

Fatal

SGDC_scanchain01

Existence check for constraint 'scan_chain'.

Language

Verilog, VHDL

A violation is reported if what is specified with the '-scanin' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_scanchain02

Existence check for constraint 'scan_chain'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-scanout' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_scanchain03

Existence check for constraint 'scan_chain'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-module' field does not exist as a module instantiated in the current design.

Severity

Fatal

SGDC_scanenable01

Existence check for constraint 'scanenable'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_scanin01

Existence check for constraint 'scanin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_scanin02

Optional Connection Point (OCP) existence check for constraint 'scanin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the OCP specified using the '-connPoint' field does not exist as a PORT or a HIER_TERMINAL or a NET in the design.

Severity

Fatal

SGDC_scanin03

Value check for constraint 'scanin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_scanin04

Value check for '-connVal' field of constraint 'scanin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-connVal' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_scanout01

Existence check for constraint 'scanout'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_scanout02

Optional Connection Point(OCP) existence check for constraint 'scanout'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the OCP specified using the '-connPoint' field does not exist as a PORT or a HIER_TERMINAL or a NET in the design.

Severity

Fatal

SGDC_scanout03

Value check for constraint 'scanout'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_scanout04

Value check for '-connVal' field of constraint 'scanout'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-connVal' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

SGDC_scanratio01

Value check for '-value' field of constraint 'scanratio'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid NUMERICAL value.

Severity

Fatal

SGDC_scanwrap01

Existence check for constraint 'scanwrap'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a SUB-MODULE or INSTANCE in the module given in the '-du' field.

Severity

Fatal

SGDC_scanwrap02

Value check for '-value' field of constraint 'scanwrap'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid NUMERICAL value.

Severity

Fatal

SGDC_scanwrap03

Existence check for constraint 'scan'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-du' field does not exist as a SUB-MODULE in the current design.

Severity

Fatal

SGDC_scanwrap04

Sanity check if '-name' is not specified along with -du in constraint 'scanwrap'.

Language

Verilog, VHDL

Rule Description

A violation is reported and scanwrap constraint is ignored if '-name' field is not specified along with '-du' field in the 'scanwrap' constraint.

Message Details

```
Warning: Constraint 'scanwrap -du <Design Unit>' is incomplete
without '-name' option and will be ignored.
```

Severity

Warning

SGDC_set01

Existence check for constraint 'set'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a port or a net in the current design.

Severity

Fatal

SGDC_set02

Value check for '-value' field of constraint 'set'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT.

Severity

Fatal

SGDC_set_pin01

Existence check for constraint 'set_pin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a 'module.port' in the current design.

Severity

Fatal

SGDC_set_pin02

Value check for '-value' field of constraint 'set_pin'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid NUMERICAL value.

Severity

Fatal

SGDC_sgdc_import01

Reports if less than or more than two values are specified after '-import'

Language

Verilog, VHDL

Rule Description

A violation is reported if less than or more than two values are specified after the '-import' field of SGDC command, 'SGDC'.

Message details

```
Invalid 'sgdc' specification. '-import' field takes a block
name followed by block-level SGDC file name.
```

Severity

Fatal

SGDC_sgdc_import02

Reports if a block-level SGDC file is specified twice for a block-level module

Language

Verilog, VHDL

Rule Description

A violation is reported if a block-level SGDC file is specified twice for a block-level module.

Message details

```
Block level SGDC file '<file>' has already been imported for
block '<block>'
```

Severity

Fatal

SGDC_sgdc_import03

Reports the generated file path

Language

Verilog, VHDL

Rule Description

Reports the path of migrated SGDC file generated corresponding to a block-level SGDC file.

Message details

```
Output SGDC file '<file>' has been generated corresponding to
'sgdc' command
```

Severity

Info

SGDC_sgdc_import04

Reports fatal when SpyGlass fails to infer path of imported SGDC files

Language

Verilog, VHDL

Rule Description

Reports fatal when SpyGlass fails to infer path of imported SGDC files, which are generated in generation step of hierarchical migration of block level SGDC files to chip level.

For validation, specify block level SGDC file(s) and the imported SGDC file(s) explicitly by using the `read_file -type sgdc` *<file>* command in the project file.

The solution is to comment all 'SGDC' commands in chip-level SGDC files and specify imported files explicitly by using the `read_file -type sgdc` *<file>* command.

Message details

```
SpyGlass could not infer output file(s) path of generation
step. Please comment SGDC command(s) 'sgdc' and specify block
level SGDC file(s) and imported SGDC file(s) using 'sgdc'
option
```

Severity

Fatal

SGDC_sgdc_import05

Reports if block-level SGDC file is not found in validation step

Language

Verilog, VHDL

Rule Description

A violation is reported if block-level SGDC file is not found in the validation step. SpyGlass does not search for imported version of such block-level file and this command will stand as ignored.

The solution is to specify imported file explicitly by using the `read_file -type sgdc <file>` command in the project file.

Message details

```
Block-level file '<file>' does not exist. Specify block level
file and its imported version explicitly using sgdc command
```

Severity

Fatal

SGDC_sgdc_import06

Reports warning when SpyGlass fails to infer path of imported SGDC file

Language

Verilog, VHDL

Rule Description

A warning is reported when SpyGlass fails to infer path of imported SGDC file, which is generated in generation step of hierarchical migration of block level SGDC file to chip level. SpyGlass does not read this block-level file for validation purpose. If this file is required for validation then provide this block level file explicitly by using the `read_file -type sgdc` <*file*> command in the project file.

The solution is to comment the reported 'SGDC' command in chip-level SGDC files and specify imported files explicitly by using the `read_file -type sgdc` <*file*> command in the project file.

Message details

```
SpyGlass could not infer output file path of generation step
for this 'sgdc' command. Please comment SGDC command(s) 'sgdc'
and specify imported SGDC files using 'sgdc' option
```

Severity

Error

SGDC_sgdc_import07

Reports when no valid 'SGDC' commands are specified in chip-level SGDC file

Language

Verilog, VHDL

Rule Description

A violation is reported if no valid 'SGDC' commands are specified in chip-level SGDC file(s). This may happen either when no 'SGDC' commands are present in chip-level SGDC file(s) or when the block-level SGDC file(s) does not exist.

SpyGlass continues the analysis assuming that you have specified block-level SGDC file(s) and imported files explicitly by using the `read_file -type sgdc` <*file*> command in the project file.

Message details

```
No valid 'sgdc' command(s) specified. SpyGlass will continue
analysis assuming block level file(s) and imported file(s) are
specified explicitly using 'sgdc' command
```

Severity

Warning

SGDC_shiftmode01

Existence check for constraint 'shiftmode'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT in the current design.

Severity

Fatal

SGDC_shiftmode02

Optional Connection Point(OCP) existence check for constraint 'shiftmode'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the OCP specified using the '-connPoint' field does not exist as a PORT or a HIER_TERMINAL in the design.

Severity

Fatal

SGDC_shiftmode03

Value check for constraint 'shiftmode'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_shiftmode04

Value check for '-connVal' field of constraint 'shiftmode'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-connVal' field is not a valid EVENT (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_testmode01

Existence check for constraint 'test_mode'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT or a HIER_TERMINAL or a NET in the current design.

Severity

Fatal

SGDC_testmode02

Value check for constraint 'test_mode'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid event (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_testmode03

Scanshift/capture cannot be applied again to the same test_mode constraint.

Language

Verilog, VHDL

Rule Description

For a `test_mode` constraint applied on a particular field, scanshift and (or) capture fields cannot be applied to it again within the same current_design.

LIMITATION: This rule does not flag for conflicting testmodes that is, capture and invertInCapture, if testmode signal name cannot be unambiguously determined as in the following examples:

```
current_design test
  test_mode -name test.n1 -value 0 -capture
  test_mode -name n1 -value 0 -invertInCapture
```

Here, net n1 & test.n1 are same. However, this situation is not handled currently.

```
current_design test
  test_mode -name n1[8:3] -value 0 -capture
  test_mode -name n1[5:4] -value 0 -invertInCapture
```

While some testmode nets (n1[5:4]) are overlapping in two cases, the rule does not flag for this case.

Message Details

```
'test_mode' constraint already specified for object '<object>'
at file '<file>', line '<line>'
```

```
'test_mode' constraint for object '<object>' has conflicting
modes: '-capture' at file '<file1>', line '<line1>', and
'-invertInCapture' at file '<file2>', line '<line2>'
```

Severity

Fatal

SGDC_testpoint01

Existence check for constraint 'testpoint'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT or a HIER_TERMINAL or a NET in the current design.

Severity

Fatal

SGDC_testpoint02

Value check for constraint 'testpoint'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-type' field is not one of (control | observe | full).

Severity

Fatal

SGDC_testpoint03

Existence check for constraint 'testpoint'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT or a HIER_TERMINAL or a NET in the current design.

Severity

Fatal

SGDC_set_case_analysis01

Existence check for the set_case_analysis constraint

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT or a HIER_TERMINAL or a NET in the current design.

Severity

Fatal

SGDC_set_case_analysis02

Value check for the set_case_analysis constraint

Language

Verilog, VHDL

Rule Description

A violation is reported if the event specified using the '-value' field is not a valid event (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_set_case_analysis_LC

Reports logic contention issues due to set_case_analysis specifications in SGDC files.

Language

Verilog, VHDL

Rule Description

A violation is reported if a net is assigned some value using `set_case_analysis`, and that net assumes a different value during simulation.

NOTE: *In such cases, assigned value is taken into consideration for simulation, and simulated value is ignored.*

This rule is checked during flat netlist creation. If SpyGlass run does not pass through flat netlist creation stage, this rule does not report violation.

For each violation, this rule traverses backwards from the point of contention, and highlights the paths from this point to the points in fan-in cone where some other `set_case_analysis` values have been specified in the current simulation cycle. This rule also displays all non-x values (for the current simulation cycle) in the schematic for the highlighted path.

Message Details

```
Net '<net>' forced to set_case_analysis value '<value>' [in
cycle '<cycle>'] whereas propagated simulation value is
'<propagated-value>'
```

Severity

Warning

SGDC_block01

Existence check for constraint 'block'.

Language

Verilog, VHDL

Rule Description

A violation is reported if there is no instantiation of what is specified with the '-name' field of 'block' constraint. '-name' could either be a <module-name> or an <entity.architecture> specification or it could be a top-level instance.

Severity

Fatal

SGDC_syncclock01

Existence check for constraint 'syncclock'

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT or a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_syncclock02

Existence check for constraint 'syncclock'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-domain' field of 'syncclock' command is not alphanumeric

Severity

Fatal

SGDC_clockgating01

Existence check for constraint 'clockgating'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-name' field does not exist as a PORT or a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_clockgating02

Existence check for constraint 'clockgating'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-pin' field does not exist as a PORT or a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_clockgating03

Value check for constraint 'clockgating'.

Language

Verilog, VHDL

Rule Description

A violation is reported if the value specified using the '-value' field is not a valid event (that is, not made up exclusively from 0/1/X/Z).

Severity

Fatal

SGDC_domain_override01

Existence check for constraint 'domain_override'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-clock_name' field does not exist as a PORT or NET or a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_domain_override02

Existence check for constraint 'domain_override'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-du_list' field does not exist as a module in the current design.

Severity

Fatal

SGDC_domain_override03

Existence check for constraint 'domain_override'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-block_list' field does not exist in the current design.

Severity

Fatal

SGDC_domain_override04

Existence check for constraint 'domain_override'.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the '-node_list' field does not exist as a PORT or NET or a HIER_TERMINAL in the current design.

Severity

Fatal

SGDC_define_rule_group01

Value check for constraint 'define_rule_group'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-name' field of 'define_rule_group' command is not alphanumeric.

Severity

Fatal

SGDC_define_rule_group03

Value check for constraint 'define_rule_group'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-rules' field of 'define_rule_group' command contains invalid rule/group/rule_group name(s).

Severity

Fatal

SGDC_define_runflow01

Value check for constraint 'define_runflow'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-name' field of 'define_rule_group' command is not alphanumeric.

NOTE: *The SGDC_define_runflow01 rule is switched off by default.*

Severity

Fatal

SGDC_define_runflow02

Value check for constraint 'define_runflow'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-name' field of 'define_runflow' command is not unique.

NOTE: *The SGDC_define_runflow02 rule is switched off by default.*

Severity

Fatal

SGDC_define_runflow03

Value check for constraint 'define_runflow'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-tag' field of 'define_runflow' command contains undefined tag name.

NOTE: *The SGDC_define_runflow03 rule is switched off by default.*

Severity

Fatal

SGDC_define_runflow04

Value check for constraint 'define_runflow'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-rulelist' field of 'define_runflow' command contains invalid rule/group/rule_group name(s).

NOTE: *The SGDC_define_runflow04 rule is switched off by default.*

Severity

Fatal

SGDC_define_runflow05

Value check for constraint 'define_runflow'.

Language

Verilog, VHDL

Rule Description

A violation is reported that '-rulelist' field of 'define_runflow' command contains rules which don't support RUNFLOW mechanism.

NOTE: *The SGDC_define_runflow05 rule is switched off by default.*

Severity

Fatal

SGDC_voltagedomain01

Existence check of the instance specified through the voltagedomain constraint.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the -instname option does not exist as an instance in the design. Use the constraint field -modname to specify the top-level module as a voltage domain.

Here are the reasons:

- Instance does not exist in the design
- Instance name has not been specified hierarchically with respect to the top
- Multiple instances if specified are not space separated
- Hierarchical Instance name contains escape symbols and has not been specified in double quotes

Severity

Fatal

SGDC_voltagedomain02

Existence check of the modname specified through the voltagedomain constraint.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the -modname option does not exist as an instance in the design. Use the constraint field -modname to specify the top-level module as a voltage domain.

Reasons

- Module does not exist in the design.
- Module name contains escaped symbols and has not been specified in double quotes.

Severity

Fatal

SGDC_voltagedomain03

Existence check for the port name specified through the voltagedomain constraint.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the -portname option does not exist as a top-level port in the design.

Reasons

Port does not exist as a top-level port in the design

Severity

Fatal

SGDC_voltagedomain04

Existence check for isolation signal specified through -isosig field of voltagedomain constraint.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the -isosig option does not exist as a VALID NET in the design.

Reasons

- Net(s) specified does not exist in the design
- Isolation signal has not been hierarchically specified from the top design
- Multiple Isolation signals if specified are not space separated
- Hierarchical Signal name contains escaped symbols has not been specified in double quotes

Severity

Fatal

SGDC_voltagedomain05

Incorrect -isoval specification in voltagedomain constraint.

Language

Verilog, VHDL

Rule Description

For voltagedomain constraint -isoval should be specified with value 1 or 0.

Reasons

Under the power down condition isolation signal can either take a value of '1' or '0'. Any other value for the isolation signal is invalid.

Severity

Fatal

SGDC_voltagedomain06

Flags if a voltagedomain constraint is defined with a zero or negative ON value.

Language

Verilog, VHDL

Rule Description

This rule checks if the ON value associated with the voltage/power domain is a non-zero positive value.

Reason

The voltage domain value, specified with '-value' (The ON value in case of power domains) must be greater than zero. Positive floating-point values as well as integers are accepted.

Severity

Fatal

SGDC_voltagedomain07

Flags if a power domain has a non-zero OFF value.

Language

Verilog, VHDL

Rule Description

This rule verifies that when a Power Domain is defined, then the OFF voltage has a value of 0.This rule will also violate if more than one OFF values have been specified for the power domain.

Reason

The off value for a power domain, specified with field '-value' of the voltagedomain constraint must be '0'. It can be specified as an integer ('0') or as a floating point value ('0.0') ONLY.

Severity

Fatal

SGDC_voltagedomain08

Existence check of the clock signal specified through voltagedomain constraint.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the -clkdomain option does not exist as a signal in the design.

Reason

- Clock Net(s) specified does not exist in the design
- Clock signal has not been hierarchically specified from the top design
- Multiple clock signals if specified are not space separated
- Hierarchical Signal name contains escaped symbols has not been specified in double quotes

Severity

Fatal

SGDC_powerdomainoutputs01

Existence check of the signal names specified through the powerdomainoutputs constraint

Language

Verilog, VHDL

Rule Description

A violation is reported if the signal specified with the -value option in the powerdomainoutputs constraint does not exist as a signal in the design.

Reason

- Net(s) specified does not exist in the design
- Signal has not been hierarchically specified from the top design
- Multiple signals if specified are not space separated
- Hierarchical Signal name contains escaped symbols has not been specified in double quotes

Severity

Fatal

SGDC_powerdomainoutputs02

Incorrect -value specification in powerdomainoutputs constraint.

Language

Verilog, VHDL

Rule Description

When -value is specified for a powerdomainoutputs constraint, then the signal values should be specified with value 0, 1, or 2.

Reason

Valid signal values for signals specified with 'powerdomainoutputs' constraints can only be:

'0' - To isolate to steady state value of '0'

'1' - To isolate to steady state value of '1'

'2' - To retain value before isolation

Severity

Fatal

SGDC_powerswitch01

Existence check of the enable net specified through powerswitch constraint.

Language

Verilog, VHDL

Rule Description

A violation is reported if what is specified with the -enable option does not exist as a signal in the design.

Reasons

- Net(s) specified does not exist in the design
- Enable signal has not been hierarchically specified from the top design
- Multiple Enable signals if specified are not space separated
- Hierarchical Signal name contains escaped symbols has not been specified in double quotes

Severity

Fatal

SGDC_supply01

Flags if a supply constraint is defined with a negative value.

Language

Verilog, VHDL

Rule Description

This rule checks if the value associated with the supply constraint is a zero or positive value.

Reason

The supply constraint is used to specify the value of the power and ground rails.

For power rails, the '-value' field of the 'supply' constraint should be positive and for ground rails, the '-value' field of the 'supply' constraint should be zero.

Integer and floating-point values are accepted.

Severity

Fatal

SGDC_waive01

waive command must have at least one of the mandatory fields

Language

Verilog, VHDL

Rule Description

This rule flags SGDC commands without at least one of the required fields.

For example, while using the "waive" command, you must supply the file list (using the -file field), the design unit list (using the -du field), or both.

Also, you must supply the message to be waived (using the -msg field), or the severity-label for which you want to waive messages (using the -severity field), or both. In case, any of these conditions is not met, the SGDC_waive01 rule flags a message.

Message Details

```
At least one of the mandatory fields must be specified in SGDC
command '<command>'
```

Severity

Fatal

SGDC_waive02

Both file name and line number must be specified with -file_line field

Language

Verilog, VHDL

Rule Description

Field '-file_line' of SGDC command 'waive', expects file name and line number pairs for values. Line number should be a positive integer greater than zero. For example:

```
waive -file_line myFile.v 10
```

Message Details

```
Missing file name or line number in '-file_line' field
```

Severity

Fatal

SGDC_waive03

File name, start line number, and end line number must be specified with -file_lineblock field

Language

Verilog, VHDL

Rule Description

Field '-file_lineblock' of SGDC command 'waive', expects file name, start line number, and end line number triplets for values.

Start and end line numbers should be positive integers greater than zero. Also, start line number should be less than or equal to end line number.

For example:

```
waive -file_lineblock myFile.v 10 20
```

Message Details

```
Missing file name, start line number, or end line number in '-file_lineblock' field
```

Severity

Fatal

SGDC_waive04

File names with spaces are not supported

Language

Verilog, VHDL

Rule Description

Space in file name is not supported

Message Details

```
File names with spaces are not supported (file '<file>'
specified with '<field>' field)
```

Severity

Fatal

SGDC_waive05

Line numbers must be non-zero positive integers

Language

Verilog, VHDL

Rule Description

Line numbers for field '-file_line' and '-file_lineblock' should be non-zero positive integers.

Message Details

```
Invalid line number (<num>) specified with '<field>' field.
Specify a non-zero positive integer
```

Severity

Fatal

SGDC_waive06

Start line number cannot be greater than end line number for -file_lineblock field

Language

Verilog, VHDL

Rule Description

Start line number should be less than or equal to end line number for field '-file_lineblock'

Message Details

```
Start line number (<start-line-num>) should be less than or
equal to end line number (<end-line-num>) for field '-
file_lineblock'
```

Severity

Fatal

SGDC_waive07

Rule / Group / Product names with spaces are not supported

Language

Verilog, VHDL

Rule Description

Space in rule, group, or product name is not supported

Message Details

```
Rule/Group/Product names with spaces are not supported (name
'<name>' specified with '<field>' field)
```

Severity

Fatal

SGDC_waive08

Only uppercase keywords can be specified with -rule and -except fields

Language

Verilog, VHDL

Rule Description

Rule and except keywords are to be specified in upper-case only.

Message Details

```
Keyword '<Keyword>' specified for '<field>' field contains
lowercase letters. Only uppercase letters are supported
```

Severity

Fatal

SGDC_waive09

Severity class/label names with spaces are not supported

Language

Verilog, VHDL

Rule Description

Space in severity label or severity class is not supported

Message Details

```
Severity class/label names with spaces are not supported (name
'<name>' specified with '-severity' field)
```

Severity

Fatal

SGDC_waive10

Both start and end weight values must be specified with -weight_range field

Language

Verilog, VHDL

Rule Description

Field '-weight_range' of SGDC command 'waive', expects a pair of integers that define the weight range that has to be waived.

The start value should be greater than or equal to the end value. Also, both start and end values should be non-zero positive integers.

For example:

```
waive -weight_range 10 20
```

Message Details

```
Missing start weight number or end weight number in '-
weight_range' field
```

Severity

Fatal

SGDC_waive11

Weight values must be non-zero positive integers

Language

Verilog, VHDL

Rule Description

Weight for fields '-weight' and '-weight_range' should be non-zero positive integer.

Message Details

```
Invalid value (<value>) specified with '<field>' field. Specify
a non-zero positive integer
```

Severity

Fatal

SGDC_waive12

Start weight value cannot be greater than end weight value for -weight_range field

Language

Verilog, VHDL

Rule Description

Start weight value should be less than or equal to end weight value for field '-weight_range'

Message Details

```
Start weight value (<start-weight-value>) should be less than
or equal to end weight value (<end-weight-value>) for field '-
weight_range'
```

Severity

Fatal

SGDC_waive13

Design unit name specified with -du or -ip fields must be a valid HDL name.

Language

Verilog, VHDL

Rule Description

Reports invalid design unit names. For example, space other than ending delimiter in escaped Verilog design unit name.

Message Details

```
Design unit name '<name>' specified in '<field>' field is an
invalid name
```

Severity

Fatal

SGDC_waive21

Same name/keyword should not be specified with -rule and -except fields.

Language

Verilog, VHDL

Rule Description

A rule/group/product/keyword specified both in -rule and -except. For example:

```
waive
  -rule Rule1 Rule2 Rule3 Group1 Group2
  -except Rule3 Group3
```

Here, 'Rule3' is specified both in '-rule' and '-except' arguments.

```
waive
  -rule Rule1 Rule2 Rule3 Group1 Group2
  -except Group2
```

Here, 'Group2' is specified both in '-rule' and '-except' arguments.

Message Details

```
Same name '<name>' specified with both '-rule' and '-except'
fields
```

Severity

Warning

SGDC_waive22

FATAL/DATA severity class/label messages should not be specified with -severity field.

Language

Verilog, VHDL

Rule Description

FATAL/DATA severity messages are not waived through 'waive' commands. Hence, the fatal/data severity labels and the fatal/data class specified in `waive` command is simply ignored.

Message Details

```
<FATAL | DATA> specified in '-severity' field is ignored
```

Severity

Warning

SGDC_waive23

Rules of FATAL severity class/label should not be specified with -rule field

Language

Verilog, VHDL

Rule Description

Violations of rule belonging to severity class, FATAL, cannot be waived by using the `waive` command.

This rule reports violation if you specify the waiver command as `waive -rule R`, where `R` is a rule of the FATAL severity. Here, `R` could be a rule list, a rule group, or simply a rule.

Following points cover different cases in which this rule reports violation:

- When you specify the command, `waive -rule R`, where `R` is a rule of the FATAL severity. In this case, the SGDC_waive23 rule flags the following message:

  ```
  Rule messages of severity class 'FATAL' cannot be waived (FATAL rule 'R' specified with '-rule' field)
  ```

- When you specify the command, `waive -rule R`, where `R` is a multi-message rule and each message of this rule is of the FATAL severity.

NOTE: *If a multi-message rule has different severities in which all the labels are not of the FATAL severity, SpyGlass does not report the SGDC_waive23 warning message. However, if the actual severity is reported as FATAL when this rule (*`R`*) is run, SpyGlass does not waive the messages of the* `R` *rule.*

- When you specify a list of rules in the waive command, and some rules in this list are of the FATAL severity.

  For example, if you specify the command, `waive -rule R1,R2,R2`, and if `R1` and `R2` rules are of the FATAL severity, SpyGlass reports the SGDC_waive23 warning message separately for the `R1` and `R2` rules.

- When you specify the command, `waive -rule ruleGroup`, and if `ruleGroup` contains few rules of the FATAL severity.

For example, consider that `ruleGroup` contains the rules, `R1`, `R2`, and `R3`, and the `R1` and `R2` rules are of the FATAL severity. In this case, SpyGlass would report the following message

```
Rule messages of severity class 'FATAL' cannot be waived.
GroupName in this waiver command contains few rule(s)
('R1,R2') which are of severity 'FATAL'
```

Message Details

The following message appears if a rule of the FATAL severity is waived by using the `waive` command:

```
Rule messages of severity class 'FATAL' cannot be waived (FATAL
rule '<rule>' specified with '-rule' field)
```

In case if a rule contains multiple messages, SpyGlass flags the above warning only if all the messages of that rule are of the FATAL severity.

The following message appears if a rule group being waived, by using the waive command, contains rule(s) of the FATAL severity:

```
Rule messages of severity class 'FATAL' cannot be
waived. GroupName in this waiver command contains few
rule(s) ('<rule-list>') which are of severity 'FATAL'
```

Severity

Warning

SGDC_waive24

Design units specified with -du or -ip fields should exist in the design

Language

Verilog, VHDL

Rule Description

The rule reports a violation if a design unit name specified in 'waive' command, is not found in the design hierarchy. It may happen either because the design unit was not at all present in the design, or because of elaboration error it could not become part of the design hierarchy.

Message Details

```
Design unit name '<name>' specified with '<field>' field was
not found in design hierarchy and hence ignored
```

Severity

Warning

SGDC_waive25

Specified regular expression does not match any design unit names

Language

Verilog, VHDL

Rule Description

Regular expression specified in 'waive' command does not match to any design unit name in the design.

Message Details

```
Regular expression '<expression>' specified in '<field>' field
does not match to any design unit names and hence ignored
```

Severity

Warning

SGDC_waive26

Specified files should exist in the current setup

Language

Verilog, VHDL

Rule Description

File name specified in 'waive' command could not be found in SpyGlass run setup. The look up domain includes -

- RTL source files
- Gates library files
- Ovl files
- Psl files
- SGDC files
- SDC files

Message Details

```
File '<file>' specified in '<field>' field is either not
present in the current source setup or is a pre-compiled file
```

Severity

Warning

SGDC_waive27

Specified regular expression does not match any files in the current setup

Language

Verilog, VHDL

Rule Description

File name specified as regular expression in 'waive' command did not match any file name in SpyGlass run setup. The search domain includes:

- RTL source files
- Gates library files
- Ovl files
- Psl files
- SGDC files
- SDC files

Message Details

```
Regular expression '<expression>' specified in '<field>' field
either did not match any source file names or matches
precompiled file names
```

Severity

Warning

SGDC_waive28

Rule / Group / Product specified with -rule or -except fields should be available in the current setup

Language

Verilog, VHDL

Rule Description

Rule or group name specified in SGDC command 'waive' is not a registered name in current SpyGlass run.

NOTE: *It may happen that the reported rule/group name belongs to a product that actually is not part of the current run.*

Message Details

```
Rule/Group/Product '<name>' specified in '<field>' field either
does not exist or is not selected in the current run
```

Severity

Warning

SGDC_waive29

Out-of-range line number specified with -file_line or -file_lineblock fields

Language

Verilog, VHDL

Rule Description

Line number specified in SGDC command 'waive', for fields '-file_line' or '-file_lineblock', is out of the file range.

Message Details

```
Line number '<num1>' is out-of-range for all files '<files>'
specified with '<field>' field. File '<file>' has maximum
'<num2>' lines
```

Severity

Warning

SGDC_waive30

Severity of rule specified with -rule field is not matching severities specified with -severity field

Language

Verilog, VHDL

Rule Description

Rule's severity does not match to severity specified in SGDC command.

Message Details

```
Severity class/label of rule '<rule>' specified with '-rule'
field does not match any of the severity classes/labels
specified with '-severity' field
```

Severity

Warning

SGDC_waive31

Reports if top module name is specified in -ip field of SGDC command ‘waive’

Language

Verilog, VHDL

Rule Description

Reports if top module name is specified in `-ip` field of the `waive` SGDC command. This will waive all violations on that top module hierarchy if none of the `-rule`, `-severity`, `-msg`, and `-weight` fields are specified.

Message Details

```
Top design unit name/expression '<name/expression>' specified
with '-ip' field will waive all messages for the hierarchy
under matching top modules '<modules>'
```

Suggestion

Use the `set_option top <du-name>` command in the project file to specify hierarchy for which you want to do the analysis

Severity

Info

SGDC_waive32

Sanity Check on Severity Label and Class

Language

Verilog, VHDL

Rule Description

Gives a message if severity label or class specified in the waive constraint is not the registered one.

Message Details

```
'<severity label/class>' specified with '-severity' is not a
registered label/class
```

Severity

Warning

SGDC_waive33

Rule with DATA severity should not be specified with -rule or -except field

Language

Verilog, VHDL

Rule Description

Data severity messages are not waived through the 'waive' command.

Message Details

```
Rule <rule> specified in <field> field is ignored as it is
registered with DATA severity
```

Severity

Warning

SGDC_waive34

Gives info message specifying the file where migrated waiver commands are generated corresponding to a particular block and block waiver file.

Language

Verilog, VHDL

Rule Description

Gives info message specifying the file where migrated waiver commands are generated corresponding to a particular block and block waiver file.

Message Details

```
Refer to file '<file>' for migrated waive commands
corresponding to block '<block>' and block waiver file
'<waiver-file>'
```

Severity

Info

SGDC_waive35

Reports in waive -import command specifies an ip name which is not present in current design.

Language

Verilog, VHDL

Rule Description

Reports if the `waive -import` command specifies the name of an IP that is not present in the current design or is a black box. Also reports if a module or a parent in it's hierarchy fails to synthesize correctly. Such commands are ignored.

Message Details

```
This import command is ignored as the ip name ('<name>') is not
present in current design
```

Severity

Fatal

SGDC_waive36

Reports when block level waiver file does not exist

Language

Verilog, VHDL

Rule Description

Reports when block level waiver file specified in the `-import` field of the `waive` command does not exist.

Message Details

```
Block level waiver file '<waiver-file>' does not exist.
Ignoring this command
```

Severity

Warning

SGDC_waive38

Gives warning message when -file is used along with -file_line/-file_lineblock.

Language

Verilog, VHDL

Rule Description

Reports when the `-file` argument of the `waive` constraint is used along with `-file_line/-file_lineblock` argument.

Such cases may result in waiving of unintended rule violations if the same file is specified in the `-file` and `-file_line/-file_lineblock` arguments.

Message Details

```
Usage of -file along with -file_line/-file_lineblock is not
recommended and would be deprecated in a future Spyglass
release
```

Severity

Warning

SGDC_watchpoint01

Existence check for '-name' field of constraint 'watchpoint'

Rule Description

A violation is reported if the argument specified with the '-name' field of the watchpoint constraint does not exist as a port or net in the current design.

Message Details

```
<net-name>'[Port + Net] not found on/within module '<du-name>'
```

Severity

Fatal

ReportUnmigratedWaivers

Reports if migrated waiver file contains non/incompletely migrated commands

Language

Verilog, VHDL

Rule Description

Reports a warning message if Migrated Waiver file contains non/incompletely migrated commands.

A separate section in the Migrated Waiver file contains these commands. Before every command in this section, reason(s) of migration failure is/are printed.

Message Details

```
Few of the waive commands in block-level waiver file could not be migrated successfully
```

Severity

Warning

Basic HDL Built-In Rules

Overview

The basic HDL built-In rules performs various checks on the user-specified HDL source files.

AutoGenerateSglib

Reports the status of sglib generation via 'enable_gateslib_autocompile' for the technology libraries specified with 'gateslib' option.

Language

Verilog, VHDL

Rule Description

The AutoGenerateSglib rule reports the status of .sglib file generation for technology libraries when the `set_option enable_gateslib_autocompile yes` command is specified in a project file.

These technology libraries are specified by using the `read_file -type gateslib` *<lib-name>* command in a project file.

Rule Exceptions

This rule does not report a violation when library compilation fails due to fatal violations.

Message Details

The following message of Error severity appears if errors are found during library compilation:

```
Sglib '<sglib-file>' have been auto-generated successfully but with errors
```

The following message of Info severity appears if the library is compiled without errors:

```
Sglib '<sglib-file>' have been auto-generated successfully
```

Severity

Error | Info

DetectTopDesignUnits

Identify the top-level design units in user design.

Language

Verilog, VHDL

Rule Description

The DetectTopDesignUnits rule identifies all top-level design units that are being processed by SpyGlass.

NOTE: *Unlike other basic rules, you cannot switch off the DetectTopDesignUnits rule; this rule will always run.*

NOTE: *The DetectTopDesignUnits rule is a high-profile rule and is reported first in all SpyGlass standard reports.*

Quite often, it is observed that SpyGlass is processing design units that are not part of design hierarchy that you would be interested in. This causes wasted CPU time and memory, and in fact may result in design appearing bigger than what it actually is required to be.

You should carefully review the top-level design unit list that the DetectTopDesignUnits rule reports. If there is a significant design component that is not relevant to your analysis, use the `set_option top` *<du-name>* command in the project file to select the design hierarchy that is relevant.

Message Details

The following message appears when a top-level design unit *<du-name>* of *<type>* is encountered:

`<type> <du-name> is a top level design unit`

Where *<type>* can be `Module` (for Verilog) or `Architecture` or `Configuration` (for VHDL).

Severity

Info

InferBlackBox

This rule is used to infer black box module interface from black box instances in given design.

Language

Verilog

Rule Description

The InferBlackBox rule infers module interfaces with port direction by looking at all black box instances in the design.

Using this approach, black box port direction can be set as defined in this file, and not same `input` (or `inout`) direction for all ports in all black box modules.

The InferBlackBox rule works on synthesized netlist. Therefore, only post-synthesis rules would get the inferred port directions.

NOTE: *Please note the following points for the InferBlackBox rule:*

- If the InferBlackBox rule successfully infers the interface for any of the black boxes in design, SpyGlass appends the string, `module interface inferred`, in the message of the *ErrorAnalyzeBBox* rule.
- The InferBlackBox rule reports only one violation irrespective of the number of black boxes in the design.

Message Details

The following message appears when the interface for a module *<module-name>* is inferred:

```
Interfaces for black box modules have been inferred. Please
refer to AnalyzeBBox violations for more details
```

Severity

Info

InferBlackBoxRTL

This rule is used to infer black box module interface from black box instances in given design.

Language

Verilog

Rule Description

The InferBlackBoxRTL rule infers module interfaces with port direction as best as it can be determined by traversing all black box instances in the design.

Using this approach, black box port direction can be set as defined in this file, and not same `input` (or `inout`) direction for all ports in all black box modules.

The InferBlackBoxRTL rule works on RTL object model. Therefore, even the RTL based rules would get the inferred port directions. This rule is CPU time intensive as compared to the InferBlackBox rule. Therefore, it is disabled by default. You can enable it by using the `set_option inferblackbox_rtl yes` command in the project file.

NOTE: *Please note the following points for the InferBlackBoxRTL rule:*

- If the InferBlackBoxRTL rule successfully infers the interface for any of the black boxes in design, SpyGlass appends the string, `module interface inferred`, in the message of the *ErrorAnalyzeBBox* rule.
- The InferBlackBoxRTL rule reports only one violation irrespective of the number of black boxes in the design.

Message Details

The following message appears when the interface for a module *<module-name>* is inferred:

```
Interfaces for black box modules have been inferred. Please refer to AnalyzeBBox violations for more details
```

Severity

Info

IgnoredLibCells

Reports ignored library cell definition from sglib if same is present in HDL

Language

Verilog, VHDL

Rule Description

The IgnoredLibCells rule reports the ignored library cell definition from sglib, if the same definition is present in HDL description.

This rule generates the spyglass_reports/SpyGlass/ignored_libcells.rpt report that contains sglib and HDL backref information.

Message Details

The following message appears to indicate ignored library cell definition from sglib if the same definition is present in HDL:

```
'<count>' library cells (for e.g. '<cell-name>') have
definition both in HDL and.sglib. The .sglib definition has
been ignored. File '<file-name>' contains the complete list of
such cells.
```

Severity

Warning

AnalyzeBBox

Analyze all black boxes present in the design

The AnalyzeBBox rule reports all modules whose interface or definition is missing in current SpyGlass analysis, or is not synthesizable, and hence would impact QOR from SpyGlass analysis.

Messages generated by this rule are of different severities depending on its impact on current analysis. Therefore, this parent rules contains multiple rules, *InfoAnalyzeBBox*, *WarnAnalyzeBBox*, *ErrorAnalyzeBBox* and *FatalAnalyzeBBox* rules.

The InfoAnalyzeBBox rule reports the following scenarios:

- Modules which have been stopped using the `set_option stop` *<module-name>* command
- Modules for which 'assume_path' constraint has been specified.

This WarnAnalyzeBBox rule reports the following scenarios:

- Library Modules whose functional view is missing in the given .sglib files
- Modules whose definition is empty or masked completely by pragmas

The ErrorAnalyzeBBox rule reports the following scenarios:

- Modules whose definition is missing
- DesignWare components used but `set_option dw yes` project file command is not specified
- Modules which could not be synthesized due to synthesis errors

The FatalAnalyzeBBox rule reports the messages generated by above WarnAnalyzeBBox and ErrorAnalyzeBBox rules if the `set_option nobb yes` project file command is specified.

NOTE: *AnalyzeBBox is a superset of existing DetectBlackBoxes and DetectLibInstances rules. Hence, these rules have been made obsolete in 390.*

Suggested Fix

Some of the possible ways to fix these issues are:

- Supply missing RTL file or library file or precompile dump, or
- If design unit has synthesis errors, then correct them or supply synthesizable definition, or

- Change some of your commands, such as `set_option pragma <values>` or `set_option define <list-of-macros>`, to make right definition visible

InfoAnalyzeBBox

Reports black boxes in the design with Info severity

Language

Verilog, VHDL

Rule Description

The InfoAnalyzeBBox rule reports all those design units which are stopped using the stop/stopfile/stopdir project file commands or black boxes for which 'assume_path' constraint has been specified.

For further details, please refer to rule *AnalyzeBBox*.

Severity

Info

WarnAnalyzeBBox

Reports black boxes in the design with Warn severity.

Language

Verilog, VHDL

Rule Description

The WarnAnalyzeBBox rule reports those library modules in the design for which functional view was not found in any of the given .sglib files. It also reports those modules which have an empty definition or definition is masked due to the use of pragmas.

For further details, please refer to rule *AnalyzeBBox*.

Severity

Warning

ErrorAnalyzeBBox

Reports black boxes in the design with Error severity.

Language

Verilog, VHDL

Rule Description

The ErrorAnalyzeBBox rule reports all those modules in the design whose definition was not found or those modules which could not be synthesized.

Further, DesignWare components are also reported if `set_option dw yes` project file command is not specified, as these remain black boxes without this switch.

For further details, please refer to rule *AnalyzeBBox*.

Violation Details

```
Design Unit '<du-name>' has no definition; black box behavior assumed
```

The following message appears if the interface for a black box is inferred successfully:

```
Design Unit '<du-name>' has no definition; black box behavior assumed and module interface inferred
```

Severity

Error

FatalAnalyzeBBox

Reports black boxes in the design with Fatal severity.

Language

Verilog, VHDL

Rule Description

The FatalAnalyzeBBox rule reports all black boxes in the design. This rule is enabled when the `set_option nobb yes` project file command is specified, and aborts the run if any black box is found in the design.

For further details, please refer to rule *AnalyzeBBox*.

Severity

Fatal

checkDupCell

Multiple definition of a technology cell in library files

Language

Verilog, VHDL

Rule Description

The checkDupCell rule reports multiple definitions of a cell in the library files.

Multiple occurrences of a technology cell may cause SpyGlass to behave in unexpected manner. SpyGlass searches for a .lib cell description in the same order in which these library files are specified. Hence, SpyGlass may use a definition that is different from the one you intend to use. You should verify that these duplicate definitions of .lib cells are not unintentional and in case of duplicate definitions, the first definition is the correct one.

NOTE: *The checkDupCell rule is switched off by default.*

Violation Details

The following violation message appears when a duplicate technology cell <*cell-name*> is encountered when the same cell has already been encountered in library <*lib-name*> (at line <*num*> in library file <*file-name*>):

```
Cell '<cell-name>' is already defined in library '<lib-name>' (file '<file-name>', line <num>)
```

Severity

Info

ZeroSizeFile

Report files having zero size

Language

Verilog, VHDL

Rule Description

The ZeroSizeFile rule reports the zero-size files specified to SpyGlass.

The ZeroSizeFile rule checks the VHDL/Verilog source files, Library files, and SpyGlass Design Constraints files.

Message Details

The following message appears when a file *<file-name>* with zero size is encountered:

```
Input <file-name> is empty
```

Severity

Warning

ReportDuplicateLibrary

Reports duplicate libraries coming from sglibs which have been ignored.

Language

Verilog, VHDL

Rule Description

The ReportDuplicateLibrary rule reports duplicate library names (and not file name).

If you specify multiple libraries of the same name, SpyGlass considers the first specified library and ignores the rest.

Message Details

The following message appears to indicate the ignored duplicate library <*lib-name*> coming from a particular sglib source <*sglib-source1*>, as that library has already been picked from the sglib source <*sglib-source2*>:

```
Library '<lib-name>' coming from '<sglib-source1>' has been ignored as it has already appeared in '<sglib-source2>'.
```

Severity

Info

ReportUDP

Reports UDP instances

The ReportUDP rule runs the *ReportUnsynthesizedDU* rule.

ReportUnsynthesizedDU

Reports design units which have not been synthesized by SpyGlass.

Language

Verilog, VHDL

Rule Description

The ReportUnsynthesizedDU rule flags design units that have not been synthesized by SpyGlass.

A design unit may not be synthesized due to any of the following reasons:

- The design unit contains non-synthesizable constructs.

  Some examples of un-synthesizable constructs are `real` declaration in Verilog, memory size exceeding threshold in Verilog, primitive use in Verilog, use of `` `active `` in VHDL, port mismatch while passing actuals to generics/parameters, and so on.

- Design unit contains nets/variables whose number of bits exceeds the specified threshold value. The `set_option mthresh` *<num>* project file command can be used to increase the threshold value. By default, the value is 4096 bits.
- Verilog Primitive is unsynthesizable.
- The design unit is in the `set_option stop` *<du-list>* list.
- The design unit is a Verilog library module and SpyGlass was invoked with the `set_option ignorelibs yes` command in the project file.
- The design unit is a cell-defined module (in Verilog) and SpyGlass was invoked with the `no_synth_celldefine` command.

The ReportUnsynthesizedDU rule also reports the reason why the design unit was not synthesized.

NOTE: *The ReportUnsynthesizedDU rule is a high-profile rule and is reported first in all SpyGlass standard reports.*

An un-synthesized design unit will be treated as black box and there will not be any post-synthesis rule-checking on these modules.

Message Details

The following message appears for an un-synthesizable module *<module-name>*:

```
Design unit '<module-name>' (elaborated name '<module-elab_name>') was not synthesized (Reason: <reason>)
```

Where *<module-elab-name>* is module's name after design elaboration and *<reason>* can be one of the following values:

- Celldefine module and ignore_celldefine specified
- Celldefine module and no_synth_celldefine specified
- Contains memory > threshold limit
- Contains unsynthesizable constructs
- Error found during synthesis
- Stop applied on design unit using stop option
- Verilog library module and ignorelibs specified
- Verilog Primitive is unsynthesizable

Severity

Warning

Notes

You can use the *<reason>* string in the ReportUnsynthesizedDU rule message to create SpyGlass `waive` constraints to waive the messages corresponding to specific category of unsynthesized design units.

For example, you want to waive all messages of the ReportUnsynthesizedDU rule for `` `celldefine `` modules when the `ignore_celldefine` option is used. Create a `waive` constraint as follows:

```
waive
  -ignore -regexp -file ".*" -rule ReportUnsynthesizedDU \
  -msg "Celldefine module and ignore_celldefine specified"
```

An un-synthesized design unit will be treated as black box and there will not be any post-synthesis rule-checking on this design unit. SpyGlass

reports messages with NotChecked severity label to inform you about design units that have been skipped for rule-checking.

The <*reason*> strings also come as part of the messages with NotChecked severity label that are generated for each rule that work on the synthesized design unit and for each design unit that was not synthesized. You can also apply waiver on these messages by appropriately specifying the <*reason*> string as part of the waive constraint. For example, an equivalent waive constraint for the above scenario is as follows:

```
waive -ignore -regexp -file ".*" -severity NotChecked \
  -msg "Celldefine module and ignore_celldefine specified"
```

PrecompileLibCheck

Checks precompiled libraries

The PrecompileLibCheck rule runs *PrecompileLibCheck01*, *PrecompileLibCheck02*, *PrecompileLibCheck03*, and *PrecompileLibCheck04* rules.

PrecompileLibCheck01

Precompiled Library path specified by user does not exist

Language

Verilog, VHDL

Rule Description

The PrecompileLibCheck01 rule is helpful when a precompiled library is specified by the user. If the specified precompiled library does not exist, this rule will show violation.

Violation Details

The following message appears when a precompiled library <*lib-name*> specified in the design does not exist in the specified path <*lib-path*>:

```
Physical path '<lib-path>' for logical library '<lib-name>'
doesn't exist
```

Severity

Warning

PrecompileLibCheck02

Precompiled Library path specified by user is empty

Language

Verilog, VHDL

Rule Description

The PrecompileLibCheck02 rule is helpful when a precompiled library is specified by the user. If the specified precompiled library is empty, this rule will show violation.

Violation Details

The following message appears when a precompiled library <*lib-name*> saved in the specified path <*lib-path*> is empty:

```
Precompiled library '<lib-name>' (Physical path: <lib-path>) is empty
```

Severity

Warning

PrecompileLibCheck03

Precompiled Library path specified by user does not contain precompiled data

Rule Language

Verilog, VHDL

Description

The PrecompileLibCheck03 rule is helpful when a precompiled library is specified by user. If the specified precompiled libraries do not contain precompiled data, the rule will show violation.

Violation Details

The following message appears when a precompiled library <*lib-name*> saved in the specified path <*lib-path*> does not contain precompiled data:

```
Physical path '<lib-path>' for logical library '<lib-name>'
doesn't contain precompile data
```

Severity

Warning

PrecompileLibCheck04

Precompiled Library path specified by user is not compatible with current version

Language

Verilog, VHDL

Rule Description

The PrecompileLibCheck04 rule is helpful when a precompiled library is specified by user. If the specified library contains precompiled dump generated by using SpyGlass 3.7.0 or earlier versions, this rule will show violation.

Violation Details

The following message appears when the specified precompiled library <*lib-name*> saved in the specified path <*lib-path*> contains precompiled data generated using SpyGlass 3.7.0 or earlier versions:

```
Precompiled library <lib-name> (Physical path: <lib-path>) is not compatible with current SpyGlass version
```

Severity

Warning

ReportStopSummary

Provides information about the stop specification given by the user

Language

Verilog, VHDL

Rule Description

The ReportStopSummary rule generates a report containing the stop specifications given to SpyGlass analysis.

The report is generated only if one of the following project file commands is specified for the analysis of the design:

- `set_option stop <du-name>`
- `set_option stopfile <file-name>`
- `set_option stopdir <directory-name>`

The report shows the design units and design files that have been skipped for SpyGlass analysis and the reason why these were skipped. In addition, the report also shows the stop specifications that were specified, but not applied because of varying reasons.

Message Details

The following message appears when the stop specifications are given during the design analysis and a report `<report-name>` is generated:

```
Design option(s) stop/stopfile/stopdir specified. Please see
<report-name> to know the design units which have been stopped
```

Severity

Info

SortVHDLFiles

Sort and dump VHDL Files dependency data

Language

VHDL

Rule Description

The SortVHDLFiles rule sorts the VHDL design files that are being analyzed using SpyGlass. It also reports the following information regarding VHDL files:

- VHDL file dependencies
- Hierarchy of VHDL files
- Sorted order of VHDL files

Message Details

The following message appears when the VHDL files are sorted by SpyGlass:

```
VHDL files have been sorted by SpyGlass
```

The following message appears when the VHDL files are sorted by SpyGlass and a report is generated and saved in a file <*file-name*>:

```
VHDL files have been sorted by SpyGlass; Please refer file
'<file-name>' for summary report
```

Severity

Info

ElabSummary

Generates the summary of elaborated design units

Description

The *ElabSummary* rule generates the *elab_summary* report that shows summary data for elaborated design units.

Language

Verilog, VHDL

Messages and Suggested Fix

The following message appears when the elaboration summary data is generated and saved in the file *<file-name>*:

```
[INFO] Please refer file '<file-name>' for elab summary report
```

Consequences of Not Fixing

Not applicable

How to Debug and Fix

Not applicable

Example Code and/or Schematic

Following is the sample *elab_summary* report generated by the *ElabSummary* rule:

```
============================================================
Elaborated Name: 'WORK_bottom_2_3'
 (RTL Design Unit name: 'bottom', test.v:1)
  Parameters (Total: 2)
  A = 2 (default: 0, overridden with 'param' option value: 2)
  B = 3 (default: 1, overridden with 'param' option value: 3)
  Instances (Total: 2): top._top_bottom,
                        top.m1.m2._mid_level2_bottom
```

```
============================================================
Elaborated Name: 'WORK_bottom_6_7'
  (RTL Design Unit name: 'bottom', test.v:1)
  Parameters (Total: 2)
  A = 6 (default: 0, overridden with 'param' option value: 2)
  B = 7 (default: 1, overridden with 'param' option value: 3)
  Instances (Total: 1): top.m1._mid_level1_bottom

============================================================
Elaborated Name: 'top'
    (RTL Design Unit name: 'top', test.v:33)
     Parameters (Total: 0)
============================================================
```

Default Severity Label

Info

Reports and/or Related Files

This rule generates the *elab_summary* report that is used to identify RTL design units for parameterized modules whose elaborated name may be different from RTL name.

The report shows the following information:

- Names of elaborated design units
- Original and updated parameter values for elaborated design units
- Names of RTL design units
- Total number of instances of each design unit

  By default, only five instances are shown. To configure this number, use the following project-file command:

  ```
  set_option report_max_inst non_negative_count <inst-count>
  ```

The report is saved in the following location:

/spyglass_reports/SpyGlass/elab_summary.rpt

The sample report is shown in *Example Code and/or Schematic*.

ReportSglibSummary

Reports path of the generated summary report on .sglib files

Language

Verilog, VHDL

Rule Description

The ReportSglibSummary rule reports the path of the report, named debug_sglib, with summary details of the .sglib files. These details include the inferred functionality for each gate that was successfully synthesized by the SpyGlass Library Compiler when the corresponding .sglib file was created.

NOTE: *This rule is run only if you specify the* `set_option enable_sglib_debug yes` *command in the project file.*

Violation Details

The following message appears when the details of the .sglib files are saved in the debug_sglib report:

```
Please refer file '<file-name>' for .sglib summary report
```

Severity

Info

ReportSglibVersionSummary

Reports path of the generated version summary report of .sglib files

Language

Verilog, VHDL

Rule Description

The ReportSglibVersionSummary rule reports the path of the version summary report, named sglib_version_summary, of .sglib files. Report contains version information of a .sglib file along with the enhancements made in subsequent releases of SpyGlass Library compiler.

NOTE: *This rule is run only if you specify the* `set_option enable_sglib_debug yes` *command in the project file.*

Message Details

The following message appears when the report is generated and saved in the sglib_version_summary.rpt file:

```
Please refer file '<file-name>' for Library Compiler version info
```

where, *<file-name>* refers to the complete path of the sglib_version_summary.rpt report.

Severity

Info

ReportMissingLibCell

Reports missing lib cell corresponding to a macro

Language

Verilog, VHDL

Rule Description

The ReportMissingLibCell rule reports physical library macros for which corresponding library cell definition could not be found in the specified gates or sglib libraries. Such macros are ignored for rule-checking.

Violation Details

The following message appears when a macro *<macro-name>* is encountered for which the library cell definition is not found:

```
Macro '<macro-name>' ignored as corresponding cell definition was not found
```

Severity

Warning

ReportMissingPowerorGroundPins

Reports missing power or ground information corresponding to a lib cell

Language

Verilog, VHDL

Rule Description

The ReportMissingPowerorGroundPins rule reports library cells without power and ground information. Such cells are processed without the power and ground information.

Power and ground information can be specified either through physical library or by using the `pgcell` SGDC command.

Violation Details

The following message appears when a library cell *<cell-name>* without power or ground pin information is found:

`Power/Ground information is not specified for cell '<cell-name>'`

Severity

Warning

ReportUnusedMacroPin

Reports if a signal pin in macro is not present in lib cell

Language

Verilog, VHDL

Rule Description

The ReportUnusedMacroPin rule reports if a signal pin in macro is not present in corresponding library cell. Such signal pins are ignored.

Violation Details

The following message appears when a signal pin *<pin-name>* of a macro *<macro-name>* is not found in the file *<file-name>* at line number *<num>*:

```
Signal pin '<pin-name>' of macro '<macro-name>' not found in cell (File: '<file-name>', Line: '<num>')
```

Severity

Warning

ReportMissingMacroPin

Reports if a lib cell pin is not present in macro

Language

Verilog, VHDL

Rule Description

The ReportMissingMacroPin rule reports those lib cell pins that are not present as signal pin in the corresponding macro.

Violation Details

The following message appears when a pin *<pin-name>* of a macro *<macro-name>*, specified in a file *<file-name>* at a line *<num>* is not a signal pin:

```
Pin '<pin-name>' of cell is not a signal pin in macro '<macro-name>' (File: '<file-name>', Line: '<num>')
```

Severity

Warning

ReportDuplicateMacro

Reports if a macro is present in plib that is also present in the lef file

Language

Verilog, VHDL

Rule Description

The ReportDuplicateMacro rule reports duplicate macros that are present in both .lef and .plib files.

Violation Details

The following message appears when the definition of a macro <*macro-name*> given in the .plib file is ignored. Instead, the definition is taken from a .lef file <*file-name*> from line <*num*>:

```
Macro '<macro-name>' of plib is ignored. Picking the definition
from lef file (File: '<file-name>', Line: '<num>')
```

Severity

Warning

ReportObsoletePragmas

Reports if RTL SGDC pragmas are specified in RTL

Language

Verilog, VHDL

Rule Description

The ReportObsoletePragmas reports when RTL SGDC pragmas are specified in RTL.

The RTL SGDC pragmas, such as //spyglass sysclock, //spyglass resetmode, //spyglass testmode, //spyglass testclock, and //spyglass blackbox have been deprecated. You need to write the corresponding SGDC commands and use the `read_file -type sgdc` *<file-name>* command in the project file to specify the SGDC file.

SGDC corresponding to RTL SGDC pragmas can be copied from SpyGlass generated file spyglass_spysch/constraint/pragma2Constraint.sgdc.

Violation Details

The following message appears when RTL SGDC pragmas are specified in RTL:

```
RTL(SGDC) pragmas have been deprecated. You can copy SGDC file
'<file-name>' generated for these pragmas, and pass it with
'sgdc' option.
```

Severity

Warning

ReportUngroup

Reports if one or more modules were ungrouped during synthesis in the current run

Language

Verilog, VHDL

Rule Description

The ReportUngroup rule reports a violation, if one or more modules were ungrouped during synthesis in the current run.

Violation Details

This rule reports the following message:

```
One or more modules ungrouped during synthesis in current run
```

Severity

Info

Synthesis Built-In Rules

Overview

Synthesis built-in rules are the synthesis and elaboration built-in messages that appear while synthesizing a Verilog or VHDL design.

The following synthesis built-in rules are switched off by default:

- *SYNTH_5115*
- *SYNTH_5116*
- *SYNTH_5117*
- *SYNTH_5130*
- *SYNTH_5132*

Please note the following points:

- The language applicable for some synthesis rules is mentioned as Verilog+VHDL in the rules spreadsheet shipped with this release. However, some rules may only be applicable for one language. In such cases, ignore the language specification as it has no impact on functionality.
- Synthesis messages reported during synthesis process

- If a library contains cells with more than eight inputs, these cells are ignored for technology mapping and the following message appears on the screen out:

  ```
  Nodes with more than 8 fanins will node be processed.
  ```

SYNTH_5014

Size of the memory is modified by the handlememory command.

Language

Verilog, VHDL

Rule Description

Size of the memory is modified by the `set_option handlememory 1` project file command.

The following constructs are treated as memory:

For Verilog, any variable or signal is declared as memory. For example,

```
reg [7:0] 2D_Mem [255:0];
```

```
reg [7:0] MultiDim_Mem [3:0][255:0];
```

For VHDL, any variable or signal is declared as array of std_logic, user_defined_bit_type, or user_defined_type (e.g., record) and is assigned in the clocked process.

For example,

```
memory_type
```

```
type memory_type is an array (255 down to 0) of
std_logic_vector (7 down to 0);
```

```
signal mem : memory_type;
```

```
user_bit
```

```
type user_bit is (high, low);   -- user_bit is user defined
type.
```

```
type user_byte is array (7 down to 0) of user_bit;
```

```
type user_memory is array (255 down to 0) of user_byte;
```

```
user_record
```

```
type user_record is
     record
     ...
```

```
     end record;
type record_byte is array (7 downto 0) of user_record;
type record_memory is array (255 downto 0) of record_byte;
```

For VHDL, the following constructs are not treated as memory:

Any port declared as memory (array of array) type. For example,

```
type memory_type is array (255 down to 0) of
std_logic_vector(7 downto 0);
port(p1:in memory_type);
```

Any member variable of record. For example,

```
type memory_type is array(255 downto 0) of std_logic_vector(7
downto 0);
type user_record is
     record
     mem : memory_type; -- member variable as memory_type;
     end record;
```

Message Details

```
Memory analysis enabled for <name> (in design unit:
<design-unit>)
```

Severity

SynthesisInfo

SYNTH_5026

You cannot pass output port to the function calls $signed and $unsigned.

Language

Verilog

Message Details

```
Output Port '<port>' is passed to the System Function Call
'<function-call>'
```

Severity

SynthesisWarning

SYNTH_5027

Module / Instance name is too long, hence truncated to 80 characters

Language

Verilog, VHDL

Message Details

```
Module/Instance name '<name>' is too long, hence truncated to
80 characters
```

Severity

SynthesisWarning

SYNTH_5028

Parallel drivers will be removed.

Language

Verilog, VHDL

Rule Description

In case we have two assignments in a module like this

```
assign x = a
assign x = a
```

only one buffer will be created.

Message Details

`Removing Parallel Driver for the output signal '<signal>'`

Severity

SynthesisWarning

SYNTH_5029

An Entity is having problem. Its instances are being ignored.

Language

Verilog, VHDL

Rule Description

Consider the following example:

```
================================
entity myNOT  is
    generic (pa:real);
        port ( A :in bit ;Z:out bit );
    end myNOT;

    architecture rtl of myNOT is
    begin
        Z <= not (A);
    end rtl;
================================
entity top  is
        port ( A :in bit ;Z:out bit );
    end top;

    architecture rtl of top is
component myNOT  is
        port ( A :in bit ;Z:out bit );
end component;

    begin
       I1 : myNOT port map (A,Z);
    end rtl;
================================
```

In the above example, the `I1` instance of `myNOT` has an elaboration error due to generic '`pa`' of type '`real`'. Due to this error, the `I1` instance has been ignored for further rule-checking.

Message Details

```
Instance '<instance-name>' of '<module-name>' having elab error
has been ignored
```

Severity

SynthesisWarning

SYNTH_5030

Other backend tools may not support defparam inside Verilog generate.

Language

Verilog

Message Details

`Other backend tools may not support defparam inside Verilog generate`

Severity

SynthesisWarning

SYNTH_5031

Other backend tools may not support type conversion on formal

Language

Verilog, VHDL

Rule Description

User-defined function calls on formals are not supported for synthesis. Type conversions on formal associations may not be supported by some backend tools.

Message Details

```
Other backend tools may not support type conversion on formal
<formal>
```

Severity

SynthesisWarning

SYNTH_5032

Hanging user instance removed during optimization

Language

Verilog, VHDL

Rule Description

SpyGlass reports this synthesis warning to indicate that hanging user instance has been removed during optimization.

To suppress this behavior, specify the following command in the TCL switch file:

```
syn_set_option remove_hanging_user_instances false
```

SpyGlass reports this rule message only for hanging user instances having at least one output or inout pin. As a result, no message is reported for instances of assertion-related modules, although such modules themselves would still be removed.

Message Details

```
Hanging user instance '<inst>' has been removed during
optimization
```

Severity

SynthesisWarning

SYNTH_5033

Multiply driven net is converted to wired-or.

Language

Verilog, VHDL

Rule Description

It is ensured that all multiply driven nets are forced to wired-or.

Message Details

`Multiple drivers on net <net>, type changed to wired-or`

Severity

SynthesisWarning

SYNTH_5034

Avoid comparison with 'don't care' or 'tristate'

Description

The *SYNTH_5034* rule reports a violation if a comparison expression contains don't care or tristate (that is `x` or `z`).

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[WARNING] `Comparison with don't care or tristate will be always false`

Consequences of Not Fixing

If you do not fix this violation, you may get undesired results as comparison with `x` or `z` is always false in `case` statements.

How to Debug and Fix

Update the reported lines by removing `x` and `z` from comparison expressions.

Example Code and/or Schematic

Consider the following example:

```
module b (q, clk, ci);
  output [3:0] q;
  input clk;
  input [3:0] ci;
  reg [3:0] q;
  always @ (posedge clk) begin
    case (ci)
      4'b000x: q = 4'h2;
```

```
        4'b000z: q = 4'h7;
        4'b0001: q = 4'h9;
      endcase
    end
endmodule
```

In the above example, the *SYNTH_5034* rule reports a violation for the `4'b000x` and `4'b000z` comparisons because comparison with `x` and `z` is always false in `case` statements.

Default Severity Label

SynthesisWarning

SYNTH_5035

Repeated case item detected

Description

The *SYNTH_5035* rule reports a violation for a duplicate expression in a `case` statement.

Language

Verilog

Messages and Suggested Fix

The *SYNTH_5035* rule reports the following message:

[WARNING] Case <case> has already been covered. Now ignoring it

Consequences of Not Fixing

If you do not fix this violation, the duplicate expression in the `case` statement is ignored from SpyGlass analysis.

How To Debug and Fix

To fix this violation, remove the duplicate expression from the `case` statement.

Example Code and/or Schematic

Consider the following example:

```
module b (q, clk, ci);
  output [3:0] q;
  input clk;
  input [3:0] ci;
  reg [3:0] q;
  always @ (posedge clk) begin
    case (ci)
      4'b0001: q = 4'h2;
```

```
      4'b0001: q = 4'h2;
    endcase
  end
endmodule
```

In the above example, the *SYNTH_5035* reports a violation for the duplicate assignment, `4'b0001: q = 4'h2`, because this condition is already covered in the previous line.

Severity

SynthesisWarning

SYNTH_5036

Non-blocking assignment in a sub program is treated as blocking for synthesis.

Rule Description

The *SYNTH_5036* rule reports a violation when a non-blocking assignment statement is placed inside a function.

Language

Verilog

Messages and Suggested Fix

The SYNTH_5036 rule reports the following message:

```
[WARNING] Non-blocking assignment "<assignment>" in subprogram
will be treated as blocking for synthesis
```

Consequences of Not Fixing

If you do not fix this violation, the non-blocking assignments are considered as blocking during synthesis.

How to Debug and Fix

To fix this violation, remove all the non-blocking assignment statements from functions.

Example Code and/or Schematic

Consider the following example:

```
module test(in1, out1);
  input [7:0]in1;
   [7:0]out1;
  assign out1 = func1(in1);
  function [7:0]func1;
    input [7:0]in1;
    reg [7:0]reg1;
    begin
```

```
      reg1 <= in1;
    end
  endfunction
endmodule
```

For the above example, the *SYNTH_5036* rule reports a violation as the non-blocking statement, `reg1 <= in1`, is written inside a function body.

Default Severity Label

SynthesisWarning

SYNTH_5037

Statement unreachable

Description

The *SYNTH_5037* rule reports a violation when a branch condition cannot be met.

Language

Verilog

Messages and Suggested Fix

The *SYNTH_5037* rule reports the following message:

```
[WARNING] Statement unreachable (Branch condition "<condition>" impossible to meet)
```

Consequences of Not Fixing

If you do not fix this violation, the reported statement is not considered during SpyGlass analysis.

How to Debug and Fix

To fix this violation, remove the reported statement or modify it so that the condition in that statement can be met.

Example Code and/or Schematic

Consider the following example:

```
module test(in1, out1);
  input in1;
  output reg [3:0]out1;
  always
    begin
    case(in1)
      0 : out1 = 4'h3;
      1 : out1 = 4'h2;
      2 : out1 = 4'h3;
```

```
    endcase
  end
endmodule
```

For the above example, the *SYNTH_5037* rule reports a violation for the following comparison:

```
2 : out1 = 4'h3
```

In this example, `in1` of width `1` can have values `[0-1]`. However, in the above comparison, the value 2 is specified, which is not reachable.

Default Severity Label

SynthesisWarning

SYNTH_5038

Statement unreachable. False path detection. Branch condition cannot be true.

Language

Verilog, VHDL

Rule Description

NOTE: *The SYNTH_5038 rule is switched off by default.*

Message Details

```
Statement unreachable (Branch condition impossible to meet)
```

Severity

SynthesisWarning

SYNTH_5039

Statement unreachable. False path detection. Branch condition cannot be true.

Language

Verilog, VHDL

Rule Description

NOTE: *The SYNTH_5039 rule is switched off by default.*

Message Details

```
Statement unreachable (Branch condition impossible to meet)
```

Severity

SynthesisWarning

SYNTH_5041

Mismatch in the LHS and RHS widths of a VHDL assignment statement

Rule Description

The *SYNTH_5041* rule reports a violation if LHS and RHS widths in a VHDL assignment statement are not equal.

Language

VHDL

Messages and Suggested Fix

The *SYNTH_5041* rule reports the following message:

```
[WARNING] The LHS width (<LHS-width>) does not match the RHS
width (<RHS-width>) of the assignment statement
```

Consequences of Not Fixing

If you do not fix this violation, incorrect LHS-RHS assignments may occur.

How to Debug and Fix

To fix this violation, modify the reported statement such that LHS and RHS widths are same.

Example Code and/or Schematic

Consider the following example:

```
entity ent is
port(
in_vector : in signed(3 downto 0) ;
out_vector : out signed (4 downto 0)
);
end;
architecture rtl of ent is
begin
A : process (in_vector)
begin
out_vector <= in_vector;
```

```
end process A;
end rtl
```

For the above example, the *SYNTH_5041* rule reports a violation for the following assignment because the width of `out_vector` is 5 and `in_vector` is 4:

```
out_vector <= in_vector
```

Default Severity Label

SynthesisWarning

SYNTH_5042

Binary operator specified in 'For-Loop' is not allowed

Language

Verilog, VHDL

Rule Description

Only following Binary operators ('+', '-', '*', '/', '%', '=', '==', '!=', * '&&', '||', '<', '>', '<=', '>=', '>>', '<<', '&', '|', and '^') are allowed in 'For-Loop'

Message Details

`This Binary Operator (<Operator>) Not Handled in For Loop`

Severity

SynthesisWarning

SYNTH_5043

Right operand for mod should be a power of 2

Language

Verilog, VHDL

Message Details

```
Right operand for mod (<mod>) should be a power of 2. Others
not supported
```

Severity

SynthesisWarning

SYNTH_5044

This SystemVerilog assertion construct is not supported for synthesis.

Language

Verilog

Rule Description

This SystemVerilog assertion construct is not supported for synthesis.

Message Details

```
The 'bind' construct is not supported. It will be ignored for
synthesis
```

Severity

SynthesisWarning

SYNTH_5046

Null range warning inside a sub-program

Description

This rule reports a violation when there is a null range declaration or use inside a sub-program. Analyzer throws warning for null range statements outside sub-program.

Language

VHDL

Messages and Suggested Fix

The *SYNTH_5046* rule reports the following message:

[SYNTHESISWARNING] Range <range> is null

Example Code and/or Schematic

The *SYNTH_5046* rule reports a violation in the following case:

```
library IEEE;
use IEEE.std_logic_1164.all;

package mypkg is
    type float_u is array (integer range <>) of std_logic;
end mypkg;

use WORK.mypkg.all;

entity test is
end test;

architecture rtl of test is
signal siga  :  float_u( 0 downto -1);
signal sigb  :  float_u( 1 downto 0);

function get_value (arg: float_u) return float_u is
variable man : float_u(arg'high-1 downto 0);    -- null
range warning
```

```
begin
    for i in 0 to arg'high-1 loop   -- null range warning
        man(i) := arg(i);
    end loop;
    return man;
end get_value;

begin
    sigb <= get_value(siga);
end rtl;
```

Severity

SynthesisWarning

SYNTH_5049

Expression is not valid for synthesis.

Language

Verilog, VHDL

Message Details

```
Expression (<expression>) is not valid for synthesis
```

Severity

SynthesisWarning

SYNTH_5054

Nets tied to supply0 or supply1 will be ignored for synthesis.

Language

Verilog

Rule Description

During synthesis nets tied to supply0 or supply1 are not considered.

Message Details

```
Assignment to supply0/supply1 net will be ignored in synthesis
```

Severity

SynthesisWarning

SYNTH_5055

Instance connection with empty port will be ignored in synthesis.

Language

Verilog, VHDL

Message Details

```
Instance connection with empty port will be ignored in
synthesis
```

Severity

SynthesisWarning

SYNTH_5057

Open input port for instance will be connected to '0' for synthesis.

Language

Verilog, VHDL

Message Details

```
Input port is open for the instance <instance>, connecting to
"0" for synthesis
```

Severity

SynthesisWarning

SYNTH_5058

Operator '===' not allowed. Hence treating it as '==' for synthesis.

Description

The *SYNTH_5058* rule reports a violation if the === operator is encountered in code.

Language

Verilog

Messages and Suggested Fix

The *SYNTH_5058* rule reports the following message:

[WARNING] Operator (===) encountered. Treating as (==) for synthesis

Consequences of Not Fixing

If you do not fix this violation, the === operator is considered as == for synthesis.

How to Debug and Fix

To fix this violation, replace the === operator with ==.

Example Code and/or Schematic

Consider the following example:

```
module test(in1, in2, out1);
  input in1, in2;
  output reg out1;
  always
  begin
    if(in1 === 1'b0)
    out1 = in2;
  end
endmodule
```

For the above example, the *SYNTH_5058* rule reports a violation for the following comparison:

```
in1 === 1'b0
```

Default Severity Label

SynthesisWarning

SYNTH_5059

Operator '!==' not allowed. Hence treating it as '!=' for synthesis.

Description

The *SYNTH_5059* rule reports a violation when the `!==` operator is encountered in code.

Language

Verilog

Messages and Suggested Fix

The *SYNTH_5059* rule reports the following message:

[WARNING] `Operator (!==) encountered. Treating as (!=) for synthesis`

Consequences of Not Fixing

If you do not fix this violation, the `!==` operator is considered as `!=` during synthesis.

How to Debug and Fix

To fix this violation, replace the `!==` operator with `!=`.

Example Code and/or Schematic

Consider the following example:

```
module test(in1, in2, out1);
  in1, in2;
  output reg out1;
  always
  begin
    if(in1 !== 1'b0)
      out1 = in2;
  end
endmodule
```

For the above example, the *SYNTH_5059* rule reports a violation for the following comparison:

```
in1 !== 1'b0
```

Default Severity Label

SynthesisWarning

SYNTH_5061

Bit/Part select is not allowed for scalar.

Language

Verilog, VHDL

Rule Description

Since, bit-select/part-select are valid only on vectors and not on scalars, the presence of bit-select/part-select on scalars are ignored for synthesis.

Message Details

```
Bit/Part select on scalar (<name>) is not meaningful. Will be
ignored for synthesis
```

Severity

SynthesisWarning

SYNTH_5063

Non-synthesizable statements are ignored for synthesis.

Language

Verilog, VHDL

Message Details

```
Statement (<statement>) is not synthesizable. Ignoring for
synthesis
```

Severity

SynthesisWarning

SYNTH_5064

Non-synthesizable statements are ignored for synthesis.

Description

The *SYNTH_5064* rule reports a violation when the `assert`, `assume`, `expect`, or `cover` statement is not synthesizable.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5064* rule reports the following message:

[WARNING] <assert | assume | expect | cover> statements are not synthesizable. Ignoring for synthesis

Consequences of Not Fixing

If you do not fix this violation, the statement containing the `assert`/`assume`/`cover`/`expect` keyword are ignored for synthesis.

How to Debug and Fix

To fix this violation, remove the `assert`/`assume`/`cover`/`expect` statement from code.

Example Code and/or Schematic

Consider the following example:

```
module test(input clk);
  int w1;
  int w2;
  int w3;
  int w4;
  sequence s1(a,b,c,d);
    @(posedge clk) (a==0) ##1 (a==1) ##1 (a==2) ##1 (a==3)
    ##1 (b==1) ##12 (c==1) ##16 (d==1);
  endsequence
```

```
  property p1;
    s1(w1,w2,w3,w4);
  endproperty
  assert property (p1) $display("PASSED\n");
  else $display("FAILED\n");
endmodule
```

In the above example, the *SYNTH_5064* rule reports a violation for the `assert` statement.

Default Severity Label

SynthesisWarning

SYNTH_5065

Expressions that cannot be evaluated statically are ignored for synthesis.

Language

Verilog, VHDL

Message Details

```
Expressions with "NULL" (<RTL-expression>) cannot be evaluated
statically hence ignoring for synthesis
```

Severity

SynthesisWarning

SYNTH_5066

Some sequential elements have been removed during optimization

Language

Verilog, VHDL

Rule Description

This rule reports a violation to indicate that some sequential elements have been removed during optimization.

Such optimization can be done for a constant flip-flop, feedback flip-flop, identical flip-flop, interdependent flip-flop, hanging flip-flop, and some conditions of a latch.

Following is the example of sequential element removal because of identical flip-flop condition:

```
module temp (d, out1, out2, clk);
  input d, clk;
  output out1, out2;
  reg out1,out2;
  always @ (posedge clk)
    out1 = d;
  always @ (posedge clk)
    out2 = d;
endmodule
```

In above example, `out1` and `out2` are the outputs of identical flip-flops. Therefore, one of them is replaced by a buffer.

Message Details

```
Some sequential elements have been removed during optimization.
Details are present in '<file-name>'
```

Severity

SynthesisWarning

SYNTH_5067

Only static values are allowed for IF | FOR Generate index.

Language

Verilog, VHDL

Message Details

```
<IF | FOR> - Generate index (<index>) cannot be evaluated
statically hence ignoring for synthesis
```

Severity

SynthesisWarning

SYNTH_5070

Value cannot be expressed in base.

Language

Verilog, VHDL

Rule Description

Base specification should be one of these: 2, 8, 10, 16.

Message Details

`Value <value> cannot be expressed in base <base>`

Severity

SynthesisWarning

SYNTH_5071

Reading or writing on a signal/shared variable declared inside packages may lead to potential mismatch between Simulation and synthesis

Language

VHDL

Description

Consider the following example:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
PACKAGE global IS
SIGNAL sig1: std_logic := 'H';
SIGNAL sig2: std_logic := 'H';
END global;
PACKAGE BODY global IS
END global;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE work.global.all;
ENTITY test IS
PORT (
d : IN std_logic;
q : OUT std_logic);
END test ;
ARCHITECTURE rtl OF test IS
BEGIN
sig1 <= d;
q <= sig2;
END rtl;
```

In the above example, `sig1` and `sig2` are shared signals because they are declared in packages. Therefore, reading or writing them can lead to simulation mismatch.

Message Details

```
Reading or writing on a signal / shared variable '<variable>'
declared inside packages may lead to potential mismatch between
Simulation and synthesis
```

Severity

SynthesisWarning

SYNTH_5095

Declaration of type VHFILEDECL is non-synthesizable

Language

VHDL

Rule Description

Only declarations of type VHVARIABLE, VHSIGNAL, VHCONSTANT are allowed.

Message Details

`Declaration (<declaration>) is not synthesizable`

Severity

SynthesisWarning

SYNTH_5100

Single character expression should not be 'X','U','W','-' and 'Z'.

Language

VHDL

Rule Description

Case item expression in case of a single character literal being equal to 'X','U','W','-' and 'Z' are not allowed.

Message Details

```
Comparisons to '<expression>' are always assumed to be false,
may cause a simulation mismatch
```

Severity

SynthesisWarning

SYNTH_5101

File Declaration is not supported for synthesis.

Language

Verilog, VHDL

Message Details

File Declaration (<declaration>) is not supported for synthesis

Severity

SynthesisWarning

SYNTH_5104

Unable to calculate index value.

Language

Verilog, VHDL

Message Details

`Could not calculate value for index`

Severity

SynthesisWarning

SYNTH_5106

Could not evaluate left value.

Language

Verilog, VHDL

Message Details

```
Could not evaluate left value
```

Severity

SynthesisWarning

SYNTH_5107

Could not evaluate right value.

Language

Verilog, VHDL

Message Details

`Could not evaluate right value`

Severity

SynthesisWarning

SYNTH_5110

Could not find do node.

Language

Verilog, VHDL

Rule Description

SpyGlass generates this language warning to indicate that SpyGlass software has encountered an internal error. Please contact SpyGlass Support (*spyglass_support@synopsys.com*) for advice on how to proceed.

Message Details

```
Could not find do node
```

Severity

SynthesisWarning

SYNTH_5111

Dimension mismatch for arguments of STD_MATCH, evaluating as FALSE.

Language

VHDL

Rule Description

Dimension mismatch for arguments of `STD_MATCH` evaluating as `FALSE`. For example, `STD_MATCH(var(1 downto 0),"1-0")` returns `FALSE` because of dimension mismatch `(2!=3)`.

Message Details

```
Dimension ('<value1>' and '<value2>') mismatch for arguments
('<argument1>' and '<argument2>') of function STD_MATCH,
evaluating as FALSE
```

Severity

SynthesisWarning

SYNTH_5113

Message number should be valid.

Language

Verilog, VHDL

Rule Description

SpyGlass generates this language warning to indicate that SpyGlass software has encountered an internal error. Please contact SpyGlass Support (*spyglass_support@synopsys.com*) for advice on how to proceed.

Message Details

`Invalid message number <num>`

Severity

SynthesisWarning

SYNTH_5115

The full_case directive is used in a case statement, but all cases are not covered.

Description

The *SYNTH_5115* rule reports a violation when you specify the `full_case` directive but do not cover all the cases in a `case` statement.

NOTE: *The SYNTH_5115 rule sometimes report false violations. Therefore, it is switched off by default.*

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5115* rule reports the following message:

[WARNING] `Used full case directive but not all cases are covered`

Consequences of Not Fixing

If you do not fix this violation, sequential logic (latches) may get generated, which may not be desirable.

How to Debug and Fix

To fix this violation, cover all the cases in a `case` statement.

Example Code and/or Schematic

Consider the following example:

```
module test(out1, clk, in1, in2, in3);
  output [1:0] out1;
  input clk;

  input [1:0] in1, in2, in3;
  reg[1:0] out1;
```

```
  always @(posedge clk)
  begin
    case (in1) // synopsys full_case
      2'b00 : out1 <= in2;
      2'b11 : out1 <= in3;
    endcase
  end
endmodule
```

For the above example, the *SYNTH_5115* rule reports a violation because all the cases for `in1` are not covered. Other possible values of `in1` are `2'b01` and `2'b10`.

Default Severity Label

SynthesisWarning

SYNTH_5116

Used parallel case directive but items may overlap.

Language

Verilog, VHDL

Rule Description

NOTE: *The SYNTH_5116 rule sometimes raises false alarms. Hence, it is switched off by default.*

Message Details

```
Used parallel case directive but items may overlap
```

Severity

SynthesisWarning

Suggested Fix

To overcome the SYNTH_5116 messages, please ensure that your design is clean with respect to the W398 rule of the SpyGlass lint solution.

SYNTH_5117

Read before write may cause difference in behavior between simulation & synthesis.

Language

Verilog, VHDL

Rule Description

NOTE: *The SYNTH_5117 rule is switched off by default.*

Message Details

```
Read before write (<variable/signal-name>) may cause difference
in behavior between simulation & synthesis
```

Severity

SynthesisWarning

SYNTH_5118

Function may not return a correct value.

Language

Verilog

Rule Description

This message is flagged for Verilog test cases in which there are function descriptions that have the return variable not being assigned in one or all of the control path(s), or even if the variable is assigned values in all the paths, there is at least a path in which the assignment uses an uninitialized variable. This might lead to junk value being returned. The following test case illustrates the issue (func is not assigned in all paths):

```
module top (sel, din, out);
input  [1:0] sel, din;
output [1:0] out;

assign out = func(din, sel);
function [1:0] func;
input    [1:0] fin, fsel;
reg      [1:0] tmp1;
begin
    if (fsel[0])
         tmp1 = fin;
    func = tmp1;
end
endfunction
endmodule
```

Message Details

```
Function (<function>) may not return a correct value as
'<return-variable>' is either unassigned or assigned some
uninitialized variable in some path(s)
```

Severity

SynthesisWarning

SYNTH_5119

Multiple architectures found for entity.

Language

VHDL

Message Details

```
Multiple architectures found for entity ( <entity> ).
Architecture <architecture> will be dumped as
<entity-name>_<architecture-name> in the output Verilog file
```

Severity

SynthesisWarning

SYNTH_5121

Left value of a range in a downto statement must be greater than the right value

Description

The *SYNTH_5121* rule reports a violation if the left index is less than the right index in a `downto` statement.

Language

VHDL

Messages and Suggested Fix

The *SYNTH_5121* rule reports the following message:

[WARNING] `Left value of range "<range>" must be greater than right value "<value>" for downto direction`

Consequences of Not Fixing

If you do not fix this violation, there can be a difference in assignment from input to output.

For example, if you write a simple architecture of assigning an input directly to an output, the following happens:

```
out1[3]=in1[0]; (when input is 0 down to 3)
out1[3]=in1[3]; (when input is 3 downto 0)
```

How to Debug and Fix

To fix this violation, update the left index so that it is greater than the right index.

Example Code and/or Schematic

Consider the following example:

```
library ieee;
  use ieee.std_logic_1164.all;
  entity test is
```

```
port(in1 : in std_logic_vector(0 downto 3);
out1 : out std_logic_vector(3 downto 0));
end test;
architecture rtl of test is
begin
end rtl;
```

In the above example, the left index of `in1` is less than the right index specified in the downto statement. Therefore, the *SYNTH_5121* rule reports a violation.

Default Severity Label

SynthesisWarning

SYNTH_5122

Right value of a range must be greater than the left value in the 'to' statement.

Description

The *SYNTH_5122* rule reports a violation when the right index is less than the left index in the `to` statement.

Language

VHDL

Messages and Suggested Fix

The *SYNTH_5122* rule reports the following message:

[WARNING] Right value of range "<range>" must be greater than left value "<value>" for to direction

Consequences of Not Fixing

If you do not fix this violation, there can be a difference in assignment from input to output.

For example, `in1[0]` gets assigned to `out1[0]` when ideally `in1[3]` should get assigned to `out1[0]`.

How to Debug and Fix

To fix this violation, make the right index greater that the left index.

Example Code and/or Schematic

Consider the following example:

```
library ieee;
  use ieee.std_logic_1164.all;
  entity test is
  port(in1 : in std_logic_vector(3 to 0);
  out1 : out std_logic_vector(3 downto 0));
```

```
end test;
architecture rtl of test is
begin
end rtl;
```

For `in1` in the above example, the right index is less than the left index for the `to` direction. Therefore, the *SYNTH_5122* rule reports a violation.

Severity

SynthesisWarning

SYNTH_5125

The declaration in this scope is not supported for synthesis.

Language

Verilog, VHDL

Message Details

`The declaration in this scope is not supported for synthesis`

Severity

SynthesisWarning

SYNTH_5126

Multiply driven net should have resolution function.

Language

Verilog, VHDL

Message Details

`Multiple driver on a net with no resolution function defined`

Severity

SynthesisWarning

SYNTH_5128

System functions within implicit state machines ignored for synthesis.

Language

Verilog, VHDL

Message Details

```
System functions within implicit state machines ignored for
synthesis
```

Severity

SynthesisWarning

SYNTH_5130

For proper synchronization between simulation and synthesis index and array size should match.

Language

Verilog, VHDL

Rule Description

NOTE: *The SYNTH_5130 rule is switched off by default.*

Message Details

```
Potential simulation-synthesis mismatch if index <index>
exceeds size '<msb>:<lsb>' of array <signal>
```

Severity

SynthesisWarning

SYNTH_5131

Unrecognized option name should be either 'named', 'positional', 'as is';- unrecognized option is taken as 'as is'.

Language

Verilog, VHDL

Message Details

```
Unrecognized option name '<name>' should be either 'named',
'positional', 'as is';- unrecognized option is taken as 'as is'
```

Severity

SynthesisWarning

SYNTH_5132

Output port is being read.

Language

Verilog, VHDL

Rule Description

This warning is issued when you try to read the value of output port as in the following Verilog example:

```
module sub2(a,b,y);
  input a;
  output b,y;

  assign y = a|b;
endmodule
```

NOTE: *Some users find the SYNTH_5132 warning to be useful while for other users, this warning might not be of much importance. Hence, this rule is switched off by default.*

Message Details

```
Output port '<port>' is being read
```

Severity

SynthesisWarning

SYNTH_5133

Ignore assignment of values to input port.

Description

The *SYNTH_5133* rule reports a violation when you make a continuous assignment to an input port irrespective of whether it is a simple assignment of a constant/signal or a complex expression.

Synthesis still infers the logic as per the assignment made in RTL.

Language

Verilog, VHDL

Message Details

The *SYNTH_5133* rule reports the following message:

```
[WARNING] Input port '<port>' is being continuously driven
```

Consequences of Not Fixing

If you do not fix this violation, the assignments to the reported port is ignored during synthesis.

How to Debug and Fix

To fix this violation, remove the assignment to the reported input port.

Example Code and/or Schematic

Consider the following example:

```
module test(in1, in2, in3);
  input in1, in2, in3;
  assign in1 = 1'b1;
  assign in3 = in1 & in2;
endmodule
```

In the above example, the rule reports violations for the following

assignments.

```
assign in1 = 1'b1;
assign in3 = in1 & in2;
```

Default Severity Label

SynthesisWarning

SYNTH_5134

Open input port will be connected to '0' for synthesis.

Language

Verilog, VHDL

Message Details

`Input port '<port>' is open, connecting to '0' for synthesis`

Severity

SynthesisWarning

SYNTH_5135

Synopsys attribute 'infer_mux' (not supported) is associated incorrectly.

Language

Verilog, VHDL

Message Details

```
Synopsys attribute 'infer_mux' (not supported) is
'<declaration-string>' associated incorrectly
```

Severity

SynthesisWarning

SYNTH_5136

Alias to a file type will be ignored for synthesis.

Language

Verilog, VHDL

Message Details

`Alias (<alias>) to a file type will be ignored for synthesis`

Severity

SynthesisWarning

SYNTH_5140

Wait on statement will be ignored for synthesis.

Language

VHDL

Message Details

```
Wait on statement will be ignored for synthesis
```

Severity

SynthesisWarning

SYNTH_5141

All signals affecting the output of a block should be present in sensitivity list.

Language

Verilog, VHDL

Rule Description

This warning message appears when not all the signals that are read appear in the sensitivity list of the block.

Consider the following examples:

```
//Verilog Example
...
always @(in1)
  myout = in1 & in2;
...

--VHDL Example
entity e is
  port(in1, in2: in bit; myout: out bit);
end;

architecture a of e is
begin
  process(in1) begin
    myout <= in1 and in2;
  end process;
end;
```

This example results in an AND gate whose inputs are `in1` and `in2`, and whose output is `myout`. The output of the AND gate will potentially change whenever input signals `in1` or `in2` changes. However, in simulation of the original description, `myout` will change only when `in1` changes, because `in1` is the only signal to appear in the sensitivity list of the block.

Solution: Add `in2` to the sensitivity list.

Message Details

```
Though not in the sensitivity list, "<signal>" is being
referred to inside this always block. Potential simulation
synthesis mismatch
```

Severity

SynthesisWarning

SYNTH_5142

Specify block is ignored for synthesis

Language

Verilog

Message Details

`Specify block is ignored for synthesis`

Severity

SynthesisWarning

SYNTH_5143

Initial block is ignored for synthesis.

Language

Verilog

Message Details

```
Initial block is ignored for synthesis
```

Severity

SynthesisWarning

SYNTH_5144

NULL ranges are not allowed in RHS | LHS.

Language

Verilog, VHDL

Message Details

`Null Range on <RHS | LHS>.`

Severity

SynthesisWarning

SYNTH_5146

Avoid Out of range write.

Language

Verilog, VHDL

Message Details

```
Out of range write possible. Simulation and synthesis behavior
may not match
```

Severity

SynthesisWarning

SYNTH_5148

Ranges for Typecasting are inconsistent. Can cause simulation error

Language

Verilog, VHDL

Message Details

```
Ranges for Typecasting are inconsistent. Can cause simulation
error
```

Severity

SynthesisWarning

SYNTH_5154

Variable assignments in clocked if construct are ignored when the variable is also being used outside the clocked-if construct's boundary.

Language

Verilog, VHDL

Rule Description

The assignment to variable `tmp` in the following example is ignored as variables cannot store values and thus should not cross the clocked-if construct's boundary:

```
...
process(reset_a, clk)
  variable tmp : std_logic;
  begin
    tmp := '0';
    if (reset_a = '1') then
      di_data <= '0';
    elsif (clk'event and clk = '1') then
      di_data <= '1';
      tmp := '1'; -- ignored
    end if;
end process;
...
```

Such models may result in simulation mismatch between pre-synthesis and post-synthesis stages.

Message Details

```
Value of variable '<variable>' is ignored in the else path of
clocked if, may cause simulation-synthesis mismatch
```

Severity

SynthesisWarning

SYNTH_5155

Defparam scope variable has a generate block as a starting scope and this scope is not the first generate block scope.

Language

Verilog

Rule Description

SpyGlass reports this warning to indicate a partial scope `defparam` statement.

Consider the following code snippet:

```
...
  generate if(P1 == 4)
   begin:B2
    for(i=0;i<=P1-1;i++)
    begin : B3
      case(0)
          1'b1: flop U1(in[i], clk, out1[i]);
           default:
                begin :D1
                   flop U2(in[i], clk, out1[i]);
                end
      endcase
    end
    defparam B3[1].D1.U2.XYZ = 5;
   end
 endgenerate.
...
```

In the above example, the `defparam` statement `defparam B3[1].D1.U2.XYZ = 5` represents the partial scope `defparam` statement.

This is because the scope variable `B3[1].D1.U2.XYZ` has the starting `generate` block scope of `B3`, while the `generate` block top scope is `B2`.

Even if the `generate` block scope represented by `B2` was un-named, this

would still be considered as a partial scope:

Message Details

```
Partial Scope (<scope>) is not supported as defparam scope
variable. Ignoring DEFPARAM for synthesis
```

Severity

SynthesisWarning

SYNTH_5158

Module name is same as that of a built-in primitive name.

Description

The *SYNTH_5158* rule reports a violation if a module name matches with a built-in primitive name.

Language

Verilog, VHDL

Message Details

The *SYNTH_5158* rule reports the following message:

[WARNING] Module name <name1> is same as that of built in primitive name, so changing the module name to <name2>

Consequences of Not Fixing

If you do not fix this violation, SpyGlass changes the name of the reported module, and consider that name during synthesis.

The naming convention is as follows:

- _m<*unique-number*> for a module
- _n<*unique-number*> for a net

How to Debug and Fix

To fix this violation, assign a different name to the reported module such that the name does not match with a built-in primitive name.

Example Code and/or Schematic

Consider the following example:

```
module RTL_NOT(in1, out1);
  input [3:0]in1;
  output [3:0]out1;
  not(out1, in1);
```

```
endmodule
```

For the above example, this rule reports a violation because `RTL_NOT` is a primitive name.

In this case, SpyGlass changes the name of the module to `rtlc_RTL_NOT_M1`.

Default Severity Label

SynthesisWarning

SYNTH_5159

VHDL 'assert' or 'report' statements are used

Description

The *SYNTH_5159* rule reports a violation when the `assert` or `report` statements are used in a VHDL code.

Language

VHDL

Messages and Suggested Fix

The *SYNTH_5159* rule reports the following violation:

[WARNING] <assert | report> statements are not supported for synthesis

Consequences of Not Fixing

If you do not fix this violation, the `assert` or `report` statements are ignored during SpyGlass analysis.

How to Debug and Fix

To fix this violation, remove the `assert` or `report` statements from the VHDL code.

Example Code and/or Schematic

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
end test;
architecture rtl of test is
signal tmp1 : integer range 0 to 10;
begin
process(tmp1)
```

```
begin
if (tmp1 < 5) then
assert false report "Stack overflow" severity error;
end if;
end process;
end rtl;
```

For the above example, the *SYNTH_5159* rule reports a violation for the `assert` statement.

Default Severity Label

SynthesisWarning

SYNTH_5162

Synthesis will ignore those Verilog generate loop statements, where assignment and condition statement refers to different genvar.

Language

Verilog

Rule Description

Synthesis ignores those Verilog generate loop statements where assignment and condition statement refers to different genvar because then number of iterations cannot be determined.

Message Details

```
Ignoring generate loop. Genvar assignments and genvar condition
in a generate loop should refer to the same genvar
```

Severity

SynthesisWarning

SYNTH_5163

The 'unaffected' assignment is ignored.

Description

The *SYNTH_5163* rule reports a violation when the `unaffected` assignment is used in code.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5163* rule reports the following message:

```
[WARNING] Identifier 'unaffected' is ignored for synthesis
```

Consequences of Not Fixing

If you do not fix this violation, the `unaffected` assignment is ignored during SpyGlass analysis.

How to Debug and Fix

To fix this violation, remove the `unaffected` assignment from code.

Example Code and/or Schematic

Consider the following example:

```
entity test is
  end test;
architecture rtl of test is
  signal s1 : integer;
  begin
    s1 <= unaffected;
  end rtl;
```

For the above example, the *SYNTH_5163* rule reports a violation for the

unaffected identifier.

Default Severity Label

SynthesisWarning

SYNTH_5164

Ignoring defparam for synthesis

Description

The *SYNTH_5164* rule reports a violation when the `defparam` identifier appears in code.

Language

Verilog

Messages and Suggested Fix

The *SYNTH_5164* rule reports the following message:

[WARNING] Component (<component-name>) not found. Ignoring DEFPARAM (<DEFPARAM-name>) for synthesis

Where, <*component-name*> refers to the instance or parameter that the `defparam` identifier is trying to change.

Consequences of Not Fixing

If you do not fix this violation, the reported `defparam` identifier is ignored during synthesis.

How to Debug and Fix

To fix this violation, remove the reported `defparam` identifier from code.

Example Code and/or Schematic

Consider the following example:

```
module top (in1, in2);
  input [3:0] in1;
  output [3:0] in2;
  wire top;
  do_assign name (in1, in2);
  defparam name.m = 4;
```

```
  endmodule
module do_assign (in1, in2);
  parameter n = 2;
  input [n-1:0] in1;
  output [n-1:0] in2;
  assign in2 = in1;
endmodule
```

For the above example, the *SYNTH_5164* rule reports a violation for `defparam` identifier that is changing the value of `m`, which is not present in the module.

Default Severity Label

SynthesisWarning

SYNTH_5165

Trying to shift by too many bits. Ignoring shift operation for synthesis.

Language

Verilog, VHDL

Message Details

```
Trying to shift by too many bits (<bits>). Ignoring shift
operation for synthesis
```

Severity

SynthesisWarning

SYNTH_5166

Non-synthesizable statement ignored.

Language

Verilog, VHDL

Message Details

(<statement>) Statement is not synthesizable. Ignoring it

Severity

SynthesisWarning

SYNTH_5167

Repetition multiplier in a concatenation expression <expression> has negative value

Language

Verilog

Rule Description

In Verilog 2001, the *SYNTH_5167* rule reports a violation for a concat expression with a negative replication constant.

Therefore, the following Verilog code is not valid and this rule reports a violation in this case:

```
assign out3 = {-1{2'b11}};
```

Message Details

```
Repetition multiplier in a concatenation expression
<expression> has negative value
```

Severity

SynthesisWarning

SYNTH_5168

DEFPARAM for nested named blocks parameters/SV Interface parameters/Compilation unit parameters/SV Package parameters are not supported. Ignoring DEFPARAM for synthesis.

Language

Verilog

Message Details

The SYNTH_5168 rule reports the following message:

```
[WARNING] DEFPARAM for nested named blocks parameters/SV
Interface parameters/Compilation unit parameters/SV Package
parameters are not supported. Ignoring DEFPARAM for synthesis
```

Consequences of Not Fixing

If you do not fix this violation, the reported defparam identifier is ignored during synthesis.

How to Debug and Fix

To fix this violation, remove the reported defparam identifier from code.

Example Code and/or Schematic

Consider the following example:

Example1

Consider the following code:

```
interface intf ();
    parameter WIDTH = 4;
endinterface


module top (intf P1);
    defparam P1.WIDTH = 8;
endmodule
```

In this example, the rule reports the following warning message:

```
DEFPARAM for SV Interface parameters are not supported.
Ignoring DEFPARAM for synthesis
```

Example2

Consider the following code:

```
parameter P = 5;
module top;
    defparam P = 3;
endmodule
```

In this example, the rule reports the following warning message:

```
DEFPARAM for Compilation unit parameters are not supported.
Ignoring DEFPARAM for synthesis
```

Example3

Consider the following code:

```
package myPack1;
    parameter p1 = 11;
endpackage


module test;
    import myPack1::p1;
    defparam p1 = 55;
endmodule
```

In this example, the rule reports the following warning message:

```
DEFPARAM for SV Package parameters are not supported. Ignoring
DEFPARAM for synthesis
```

Severity

SynthesisWarning

SYNTH_5169

Out of bound bit select of a parameter, replacing with 'x'.

Language

Verilog, VHDL

Message Details

```
Out of bound bit select of a parameter, replacing with 'x'
```

Severity

SynthesisWarning

SYNTH_5170

Repetition multiplier in a concatenation expression <expression> has zero value

Language

Verilog

Rule Description

In Verilog 2001, the *SYNTH_5170* rule reports a violation for a concat expression with a zero replication constant.

Therefore, the following Verilog code is not valid and this rule reports a violation in this case:

```
assign out2 = {0{1'b1}}
```

Message Details

```
Repetition multiplier in a concatenation expression
<expression> has zero value
```

Severity

SynthesisWarning

SYNTH_5171

non-synthesizable usage.

Language

Verilog, VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that SpyGlass software has encountered an internal error. Please contact SpyGlass Support (*spyglass_support@synopsys.com*) for advice on how to proceed.

Message Details

```
This usage of <name> is not synthesizable
```

Severity

SynthesisError

SYNTH_5172

Exits because of Memory Allocation Failed.

Language

Verilog, VHDL

Message Details

`Out of memory.`

Severity

SynthesisError

SYNTH_5173

Design unit used earlier as a black box

Language

Verilog, VHDL

Message Details

```
Design unit <design-unit> used earlier as a black box (file:
<file>)
```

Severity

SynthesisError

SYNTH_5174

Design contains a Verilog module and a VHDL entity with same name.

Language

Verilog, VHDL

Message Details

```
Design contains a Verilog module and a VHDL entity with same
name '<name>'
```

Severity

SynthesisError

SYNTH_5175

Predefined attribute is non synthesizable

Description

The *SYNTH_5175* rule reports a violation when a VHDL predefined attribute is used in the code.

Language

VHDL

Messages and Suggested Fix

The *SYNTH_5175* rule reports the following message:

[ERROR] Predefined Attribute <attribute> is non-synthesizable

Consequences of Not Fixing

If you do not fix this violation, the reported attribute is ignored from synthesis.

How to Debug and Fix

To fix this violation, remove and/or replace the predefined VHDL attribute from code.

Example Code and/or Schematic

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
port(in1, in2 : in std_logic;
out1 : out std_logic);
end test;
architecture rtl of test is
begin
process(in1, in2)
```

```
begin
if(in1'active) then
out1 <= in2;
end if;
end process;
end rtl;
```

For the above example, the *SYNTH_5175* rule reports a violation for the usage of the VHDL predefined attribute `active`.

Default Severity Label

SynthesisError

SYNTH_5176

If function returns nothing it is flagged.

Language

Verilog, VHDL

Rule Description

Any function is expected to return at least one value. If this is not the case, it is syntactically wrong. But if it does not return any value under some conditions, it will be synthesized issuing a warning.

Message Details

`Function (<function>) may not return a value`

Severity

SynthesisError

SYNTH_5178

For module defined in target library, specparam value for PRIM_FUNC is NULL

Language

Verilog, VHDL

Message Details

`For module <module> defined in target library, specparam value for PRIM_FUNC is NULL.`

Severity

SynthesisError

SYNTH_5179

Port name in specparam not found in the module

Language

Verilog

Message Details

`Port name <name> in specparam not found in the module <module>.`

Severity

SynthesisError

SYNTH_5180

specparam value for Port not found in cell

Language

Verilog

Message Details

```
specparam value <value> for Port <port> not found in cell
<cell>.
```

Severity

SynthesisError

SYNTH_5181

specparam value for Port is NULL.

Language

Verilog

Message Details

`specparam value for Port <port> is NULL.`

Severity

SynthesisError

SYNTH_5182

Port name in specparam is NULL.

Language

Verilog

Message Details

`Port name in specparam is NULL.`

Severity

SynthesisError

SYNTH_5183

specparam for all the ports in the module is not defined.

Language

Verilog

Message Details

```
specparam for all the ports in the module '<module>' is not
defined.
```

Severity

SynthesisError

SYNTH_5184

could not open target library for opt_level > 0.

Language

Verilog, VHDL

Rule Description

SpyGlass generates this synthesis error to indicate that SpyGlass software has encountered an internal error. Please contact SpyGlass Support (*spyglass_support@synopsys.com*) for advice on how to proceed.

Message Details

```
could not open target library <library> for opt_level > 0.
```

Severity

SynthesisError

SYNTH_5185

Equivalent Module not found in the target library, so could not map the module.

Language

Verilog, VHDL

Message Details

```
Equivalent Module for <name> not found in the target library,
so could not map the module.
```

Severity

SynthesisError

SYNTH_5186

Invalid condition of if-statement in asynchronous implicit style sequential state machine. Not supported.

Language

Verilog, VHDL

Message Details

```
Invalid condition of "if" statement in asynchronous implicit
style sequential state machine. Not supported
```

Severity

SynthesisError

SYNTH_5187

Invalid event control statement in asynchronous implicit style sequential state machine.

Language

Verilog, VHDL

Message Details

```
Invalid event control statement in asynchronous implicit style
sequential state machine
```

Severity

SynthesisError

SYNTH_5188

Invalid placement of event control statement inside an asynchronous implicit-style always block.

Description

The *SYNTH_5188* rule reports a violation when an event control statement is placed inside an asynchronous implicit-style `always` block.

Language

Verilog

Messages and Suggested Fix

The *SYNTH_5188* rule reports the following message:

[ERROR] `Invalid placement of event control statement inside asynchronous implicit style always block. Not supported`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, remove the event control statement from the `always` block.

Example Code and/or Schematic

Consider the following example:

```
module test(in1, clk, rst, out1);
  input in1, clk, rst;
  output reg out1;

  always@(posedge clk, negedge rst)
    begin
```

```
      if(!rst)
        out1 <= @(posedge clk) 1'b1;
      else
        out1 <= 1'b0;
    end
endmodule
```

For the above example, the *SYNTH_5188* rule reports a violation for the `out1 <= @(posedge clk) 1'b1` statement.

Default Severity Label

SynthesisError

SYNTH_5189

If statement does not conform to the implicit style of modeling.

Language

Verilog, VHDL

Message Details

`If statement does not conform to the implicit style of modeling`

Severity

SynthesisError

SYNTH_5190

Only LOGICAL 'OR', 'NOT', 'AND' and '==' operators are allowed in conditions of 'if' statements in asynchronous part of an 'if-else' or conditional operator ladder, inside sequential always block.

Language

Verilog

Rule Description

NOTE: *The SYNTH_5190 error is suppressed, and SYNTH_149 is flagged in place of this error. If SYNTH_5190 is flagged then please contact SpyGlass Support (spyglass_support@synopsys.com).*

Message Details

```
Only LOGICAL 'OR', 'NOT', 'AND' and '==' operators are allowed
in conditions of 'if' statements in asynchronous part of an
'if-else' or conditional operator ladder, inside sequential
always block
```

Severity

SynthesisError

SYNTH_5191

Invalid condition of if statement inside asynchronous sequential always block.

Language

Verilog, VHDL

Message Details

```
Invalid condition of if statement inside asynchronous
sequential always block
```

Severity

SynthesisError

SYNTH_5192

Signal edge in the condition of an if statement does not match with the signal edge in the sensitivity list of an always block.

Description

The *SYNTH_5192* rule reports a violation if the signal edge used in the condition of an `if` statement does not match with the signal edge used in the sensitivity list of an `always` block.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Signal edge of "<name>" used in condition of if statement does not match that specified in the sensitivity list of always block`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Update the edge of the signal either in the `if` condition or in the sensitivity list of the `always` block so that both the edges match.

Example Code and/or Schematic

Consider the following example:

```
module test(clock, reset, in1, out1);
  input clock, reset, in1;
  output reg out1;
  always @(posedge clock or negedge reset)
```

```
  begin
  if(reset)
    out1 = 1'b0;
  else
    out1 = in1;
  end
endmodule
```

In the above example, the `reset` signal is defined to be negative edge sensitive. However, this signal is checked for positive value in the `if` condition. This is incorrect. Either the `reset` signal should be made positive edge sensitive, or the `if` condition should check for `~reset`.

Default Severity Label

SynthesisError

SYNTH_5193

Any signal or variable or constant remains unconstrained after elaboration, error is flagged.

Language

Verilog, VHDL

Rule Description

Any signal or variable or constant remains unconstrained after elaboration, this error message is flagged

If a signal or variable or constant is unconstrained and it is known at analysis time only, then the error message comes at analysis time.

Message Details

```
Type of "<name>" is unconstrained after elaboration
```

Severity

SynthesisError

SYNTH_5194

Multiple Configuration is not supported yet.

Language

VHDL

Message Details

```
Multiple Configuration is not supported yet.
```

Severity

SynthesisError

SYNTH_5195

No actual corresponding to formal is found.

Description

The *SYNTH_5195* rule reports a violation if the same escaped names are not used in Verilog and VHDL code.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5195* rule reports the following message:

[ERROR] No actual corresponding to formal <formal> is found.

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

To fix this violation, specify the same escaped names in the Verilog and VHDL files.

Example Code and/or Schematic

Consider the following Verilog module and VHDL entity:

Verilog Module:

```
module _Dut(Bert, BERT);
  parameter IP = 1;
  input Bert;   // Both have uppercase chars
  input BERT;
  wire [IP:0] testReg;
  assign testReg = 2;
endmodule
```

VHDL entity:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity tb is
end entity tb;

architecture arch of tb is
    component \_Dut\
        generic ( IP : integer);
        port (\Bert\ : in std_logic;
              \BERT\ : in std_logic);
    end component \_Dut\;

    signal \Bert\ : std_logic := '0';--ignoring this sig in
                                     -- port conn
    signal \BERT\ : std_logic := '1';--Consider this value in
                                     -- port conn


begin
    i1_dut : \_Dut\
        generic map (IP => 0)
        port map (\Bert\ => \Bert\, \BERT\ => \BERT\);
end architecture arch;
```

In the above example, an escaped name is used for the `bert` port. However, in the Verilog file, the escaped name is not used. So during port mapping, `\Bert` and `\BERT` are not available in the `_Dut` module. Therefore, the *SYNTH_5195* rule violation appears.

To fix this violation, use an escaped name for `Bert` and `BERT` in the `_Dut` module.

Default Severity Label

SynthesisError

SYNTH_5196

Could not elaborate design units.

Language

Verilog, VHDL

Message Details

`Could not elaborate design units.`

Severity

SynthesisError

SYNTH_5197

Bit/Part select is not allowed for scalar.

Language

Verilog, VHDL

Rule Description

Bit-select/part-select is valid only on vectors and not on scalars so, the presence of bit/part select on scalars are not synthesizable.

Message Details

`Bit/Part select on scalar (<scalar>) is not synthesizable`

Severity

SynthesisError

SYNTH_5198

Left | Right range should be static.

Description

The *SYNTH_5198* rule reports a violation to indicate a non static left or right range.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] <Left | Right> range (<range>) is not static

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass checking is done.

How to Debug and Fix

Modify the reported code to specify a static range.

Example Code and/or Schematic

Consider the following Verilog entity:

```
module top (in1, out1);
parameter P1 = 'hz;
input [P1:0] in1;
output logic [P1:0] out1;
always
out1 = ~in1;
endmodule
```

In the above example, the *SYNTH_5198* rule reports a violation because the value of P1 is unknown. Due to unknown value of P1, the range for the

in1 input becomes non-static.

Default Severity Label

SynthesisError

SYNTH_5199

Port remains unconstrained even after elaboration

Description

The *SYNTH_5199* rule reports a violation if a port remains unconstrained even after elaboration.

NOTE: *The ports declared in a top module cannot be constrained.*

Language

VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] Port <port> remains unconstrained after elaboration and hence not synthesizable

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Constraint the reported port after elaboration.

Example Code and/or Schematic

Consider the following example:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity test is
port(
in_port : in std_logic_vector;
out_port : out std_logic_vector
);
```

```
end entity test;
architecture behavior of test is
begin
out_port <= in_port;
end architecture behavior;
```

In the above example, the `in_port` and `out_port` ports are declared as `std_logic_vector`, and therefore, are unconstrained.

If the `icache_flush` entity is instantiated at another place in a design and the signals connected to these ports have a defined size, the port sizes can be determined from the instance port after elaboration. However, in this example, the port sizes are undefined even after elaboration because the ports are declared in a top-level module.

Default Severity Label

SynthesisError

SYNTH_5200

Chandle Data Type is not synthesizable.

Description

The *SYNTH_5200* rule reports a violation on usage of chandle data types.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] Chandle Data Type is not synthesizable

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the usage of ‘chandle’ data-type and use another data-types instead, which are synthesizable.

Example Code and/or Schematic

Consider the following example:

```
interface intf6(input wire w0);
wire w1[1:0];
reg w2;
chandle w7;
endinterface

module test(intf6 I1);
wire out1;
assign out1 = I1.w1[1] && I1.w2;
endmodule
```

In the above example chandle data-type is used in interface.

Default Severity Label

SynthesisError

SYNTH_5201

Not a Valid Case Item Expression Having Unknowns.

Language

Verilog, VHDL

Message Details

```
Not a Valid Case Item Expression Having Unknowns.
```

Severity

SynthesisError

SYNTH_5202

If min/typ/max values for a delay are present then module becomes unsynthesizable.

Language

Verilog, VHDL

Message Details

```
Module not synthesizable due to presence of delay
```

Severity

SynthesisError

SYNTH_5205

Bad character in binary / octal / hex string.

Language

Verilog, VHDL

Rule Description

The allowed characters are as follows:

Type	Allowed Characters	Handling of
Binary	0, 1, ?, z, Z, x, X	Any z/Z/x/X/? character in the string is converted to '0'
Octal	0, 1, 2, 3, 4, 5, 6, 7, z, Z, x, X	Any z/Z/x/X character in the string is converted to '000'
Hex	0, 1, 2, 3, 4, 5, 6, 7, z, Z, x, X, A, B, C, D, E, F, a, b, c, d, e, f	Any z/Z/x/X character in the string is converted to '0000'

Message Details

```
Bad character <character> in <string> string
```

Severity

SynthesisError

SYNTH_5209

The parameter which is being used have undefined value due to some error, One example could be division by zero error

Language

Verilog, VHDL

Message Details

```
Value of Parameter '<parameter>' is undefined
```

Severity

SynthesisError

SYNTH_5210

For an array of interface instantiation if we are passing some signals in the port of the interface then we need to perform some port mapping that will map each instance of the interface array with the corresponding variable

Language

Verilog, VHDL

Message Details

```
Array of Interface Instances with ports is not currently
supported
```

Severity

SynthesisError

SYNTH_5211

Currently we are not supporting hierarchical references for synthesis.

Language

Verilog, VHDL

Message Details

```
Hierarchical reference '<reference>' is not supported for
synthesis
```

Severity

SynthesisError

SYNTH_5212

Expansion unsuccessful

Language

Verilog, VHDL

Message Details

`<name> Expansion unsuccessful`

Severity

SynthesisError

SYNTH_5213

Invalid target library specified.

Language

Verilog, VHDL

Rule Description

Cannot expand macros for specified library as the specified target library is invalid. If no target library specified the expansion is done to generic cells "RTL_".

Message Details

```
Cannot expand macros for specified library
```

Severity

SynthesisError

SYNTH_5214

The MSB or LSB expected to lie between the range 2147483647:2147483646

Description

The *SYNTH_5214* rule reports a violation if MSB or LSB lies outside the range of 2147483647:214748364.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

```
[SynthesisError] MSB or LSB '<name>[<num1>:<num2>]'is lying outside the range <-2147483647 to 2147483646>
```

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Specify MSB and LSB within the range of 2147483647:214748364.

Example Code and/pr Schematic

Verilog Example

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
port (
data : in std_logic_vector(0 to 5);
sum : out std_logic_vector(0 to 5));
```

```
end test;
architecture rtl of test is
signal sum_v : std_logic_vector(2147483642 to 2147483647);
begin
sum_v <= data;
sum <=sum_v;
end rtl;
```

In the above example, the *SYNTH_5214* rule reports a violation because the `sum_v` signal is declared as *`std_logic_vector`* (2147483642 to 2147483647).

However, the value 2147483647 is not within the supported range, which is 2147483647:2147483646.

VHDL Example

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
generic (gen1:integer := 1);
port (
data : in std_logic_vector(0 to 5);
sum : out std_logic_vector(0 to 5));
end test;
architecture rtl of test is
type t1 is array (integer range<>) of std_logic;
signal sum_v : std_logic_vector(gen1-5 to gen1);
begin
sum_v <= data;
sum <=sum_v;
end rtl;
library ieee;
use ieee.std_logic_1164.all;
entity top is
port (
data : in std_logic_vector(0 to 5);
```

```
sum : out std_logic_vector(0 to 5));
end top;
architecture rtl of top is
component test is
generic (gen1:integer := 1);
port (
data : in std_logic_vector(0 to 5);
sum : out std_logic_vector(0 to 5));
end component;
begin
I1:test generic map (-2147483643) port map (data,sum);
end rtl;
```

In the above example, the *SYNTH_5214* rule reports a violation because the range of the `sum_v` signal for the `I1` instance (where `gen1` is passed as `2147483643`) will become `-2147483648(gen1-5)` to `-2147483643(gen1)`. However, this range is outside the supported range, which is `2147483647:2147483646`.

Default Severity Label

SynthesisError

SYNTH_5216

Only input, output and inout port directions allowed.

Language

Verilog, VHDL

Rule Description

Port directions other than input, output, and inout are not allowed. Hence no synthesis is done.

Message Details

```
Port direction is neither 'input' nor 'output' or 'inout'
```

Severity

SynthesisError

SYNTH_5217

Any task or function should have equal number of actual arguments as formal

Language

Verilog, VHDL

Message Details

```
Mismatch in number of actual and formal arguments for Task/
Function '<name>'
```

Severity

SynthesisError

SYNTH_5218

Could not open library file

Language

Verilog, VHDL

Rule Description

Could not open library file.

Message Details

`Could not open library file <file>`

Severity

SynthesisError

SYNTH_5219

Error in parsing the library file.

Language

Verilog, VHDL

Message Details

`Error in parsing the library file <file>`

Severity

SynthesisError

SYNTH_5220

Could not break loops for module.

Language

Verilog, VHDL

Rule Description

Combinational loops in the design are detected. For loop breaking it creates pseudo primary-output/primary input pairs. If the attempt to break the loop fails no synthesis is done.

Message Details

`Could not break loops for module <module>.`

Severity

SynthesisError

SYNTH_5221

Invalid net name being accessed

Description

The *SYNTH_5221* rule reports a violation if an invalid net name is accessed while using the generate-block name referencing or while using hierarchical names.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] Invalid Net Name (<net-name>) being accessed

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the reported net from the code.

Example Code and/or Schematic

Example 1

Consider the following example:

```
module test(input [3:0] in, output [3:0] out);
    parameter num_req = 4;
    wire [3:0] out;
genvar i;
    generate
        for (i = 0; i < 4; i = i + 1)
           begin : vargen1
```

```
                    wire compres1;
                    assign vargen1[i+1].compres1 = in[i];
            end
    endgenerate
endmodule
```

In the above example, an invalid net name is generated from the out-of-bound use of the *for-generate* index while referencing a generate-block name.

The *SYNTH_5221* rule reports a violation in this case because `in[i]` is getting assigned to `vargen1[i+1].compres1`, but '`i+1`' goes out of range in the last iteration.

Default Severity Label

SynthesisError

SYNTH_5222

Only Port directions 'input' and 'output' expected.

Language

Verilog, VHDL

Rule Description

Only Port directions 'input' and 'output' expected.

Message Details

`Unknown port type (<type>) in buffer`

Severity

SynthesisError

SYNTH_5223

Width of signals should not exceed 4000000.

Description

The *SYNTH_5223* rule reports a violation when the width of signals in a declaration is greater than 4000000.

Language

Verilog, VHDL

Messages and Suggested Fix

[SynthesisError] Width of '<declaration>' greater than '4000000' not supported for declaration

Consequences of Not Fixing

If the width of signals is greater than 4000000 (even if the used range is less), the vector net for the entire width is created which can cause a lot of memory/time consumption.

How to Debug and Fix

Modify the RTL to remove such large buses.

Example Code and/or Schematic

Consider the following example in VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
entity RAM is
generic
(
address_width: INTEGER := 30;
data_width: INTEGER := 30
);
```

```
port
(
CLOCK : in  STD_LOGIC;
DATA: in  STD_LOGIC_VECTOR(data_width - 1 downto 0);
WRITE_ADDRESS : in  STD_LOGIC_VECTOR(address_width - 1 downto
0);
READ_ADDRESS : in  STD_LOGIC_VECTOR(address_width - 1 downto
0);
WE : in  STD_LOGIC;
Q : out STD_LOGIC_VECTOR(data_width - 1 downto 0)
);
end RAM;

architecture RTL of RAM is
type ram is array(0 to 2 ** address_width - 1) of
STD_LOGIC_VECTOR(data_width - 1 downto 0);

signal RAM_BLOCK : ram;
begin
process (CLOCK)
begin
if (CLOCK'EVENT and CLOCK = '1') then
if (WE = '1') then
RAM_BLOCK(CONV_INTEGER(UNSIGNED(WRITE_ADDRESS))) <= DATA;
end if;
Q <= RAM_BLOCK(CONV_INTEGER(UNSIGNED(READ_ADDRESS)));
end if;
end process;
end RTL;
```

In the above example, the *SYNTH_5223* rule reports a violation because the width of the RAM_BLOCK signal is greater than 4000000. Signals of such widths are not supported by SpyGlass even if the used range is less.

Default Severity Label

SynthesisError

SYNTH_5224

Task/Function of a module cannot be accessed through its instance.

Language

Verilog, VHDL

Message Details

```
Access of Task/Function of module '<module>' through its
instance is not synthesizable
```

Severity

SynthesisError

SYNTH_5225

Parse failed

Language

Verilog, VHDL

Message Details

`Parse failed`

Severity

SynthesisError

SYNTH_5226

Synthesis failed due to the mentioned error

Description

The *SYNTH_5226* rule reports the SpyGlass error, such as *ELAB_633*, due to which synthesis failed.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] `Error <SpyGlass-generated-error> found during synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Check the error name reported in the violation message. Based on the reported error, fix it accordingly.

For example, if the error reported in the violation message is *ELAB_633*, fix this error by removing the infinite loop from the code.

Example Code and/or Schematic

Not applicable

Default Severity Label

SynthesisError

SYNTH_5227

Right operand of rem should be static.

Language

Verilog, VHDL

Message Details

```
Non-static right operand in rem not supported for synthesis
```

Severity

SynthesisError

SYNTH_5228

X or Z in case item expression not valid.

Language

Verilog, VHDL

Message Details

`X or Z in case item expression not valid`

Severity

SynthesisError

SYNTH_5229

Unknown values in case item expression.

Language

Verilog, VHDL

Message Details

`Unknown values in case item expression.`

Severity

SynthesisError

SYNTH_5230

Number of iterations in <for | while> -loop exceeds max.

Language

Verilog, VHDL

Rule Description

The SYNTH_5230 rule flags an error when the number of iterations in the `for` or `while` loop exceeds the maximum limit.

NOTE: *By default, the maximum limit is set to 2048. You can change this limit by using the* `set_option sgsyn_loop_limit` *<value> command in the project file.*

Message Details

```
Number of iterations in <for | while>-loop exceeds max.
allowable limit (<limit>).
```

Severity

SynthesisError

SYNTH_5231

Unary operator not valid.

Language

Verilog, VHDL

Message Details

`Unary operator (<operator>) not valid.`

Severity

SynthesisError

SYNTH_5232

Nested interfaces are not supported for synthesis.

Description

The *SYNTH_5232* rule reports a violation if SpyGlass encounters a nested interface.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Currently nested interface '<interface>' is not supported for synthesis

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove nested interfaces from the code.

Example Code and/or Schematic

Consider the following example in Verilog:

```
interface I1;
logic [3:0] sum;
modport m2 (output sum);
interface I2;
logic [3:0] address;
modport m1 (input address);
endinterface
```

```
endinterface
module test (I1 a );
endmodule
```

In the above example, the *SYNTH_5232* rule reports a violation because the `I2` interface is defined in the `I1` interface. Nested interfaces are not supported for synthesis.

Default Severity Label

SynthesisError

SYNTH_5233

For loop not unrollable.

Language

Verilog, VHDL

Message Details

`For loop not unrollable`

Severity

SynthesisError

SYNTH_5234

Range not computable

Description

The *SYNTH_5234* rule reports a violation if the range in an expression cannot be computed by SpyGlass.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] <range> Range not computable

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Modify the code such that the expression range becomes computable.

Example Code and/or Schematic

Consider the following example in Verilog:

```
module test(din,address);
input  [15:0] din;
input  [3:0] address;
typedef reg [7:0] mybyte [15:0];
typedef struct {
bit a;
mybyte array;
reg [2:0] b ;
}strc;
strc strc1;
```

```
always@(*)
strc1.array[address][address:5] <= din;
endmodule
```

In the above example, the *SYNTH_5234* rule reports a violation because `address` defined as input is non-static. Therefore, the `[address:5]` range in the expression `strc1.array[address][address:5]` is not computable.

Default Severity Label

SynthesisError

SYNTH_5235

Division by zero is not allowed.

Language

Verilog, VHDL

Message Details

```
Division by zero is illegal
```

Severity

SynthesisError

SYNTH_5236

Unknown module in debug.

Language

Verilog, VHDL

Message Details

`Unknown <module> module in debug <debug>`

Severity

SynthesisError

SYNTH_5237

Trying to access element of interface which is not a part of modport passed as a port in this module.

Language

Verilog, VHDL

Rule description

SpyGlass flags this synthesis error in a situation like mentioned below:

Consider a scenario in which an interface is instantiated using a modport and through this instantiated interface's instance, one tries to access an interface element that is not a part of the modport using which the interface is instantiated. In such a situation, SpyGlass flags this synthesis error, since one can only access the interface elements that are specified in the modport. To come out of this situation, either add that element in the modport or instantiate the interface without using any modport so that all the interface elements can be accessed with this instance.

Message Details

```
Trying to access element '<element>' of interface which is not
a part of modport passed in the module
```

Severity

SynthesisError

SYNTH_5238

Export declaration of task and function inside modport are not synthesizable.

Message Details

```
Export declaration of task and function inside modport are not
synthesizable
```

Severity

SynthesisError

SYNTH_5240

This rule has been deprecated

Use the *SYNTH_5338* and *SYNTH_5344* rules instead of the *SYNTH_5240* rule.

SYNTH_5241

There is a path in the sequential m/c with no real state in between.

Language

Verilog, VHDL

Message Details

```
There is a path in the sequential m/c with no real state in
between.
```

Severity

SynthesisError

SYNTH_5242

Cannot resolve hierarchical reference

Language

Verilog, VHDL

Message Details

`Cannot resolve hierarchical reference '<reference>'`

Severity

SynthesisError

SYNTH_5243

Synthesis does not support implicit ref port while processing interface port.

Language

Verilog, VHDL

Message Details

```
Synthesis does not support implicit ref port ‘<reference-port>’
while processing interface port '<interface-port>' at line
<line> in file '<file>'.
```

Severity

SynthesisError

SYNTH_5245

Synthesis failed for the module since it has unsupported SystemVerilog constructs

Language

Verilog

Message Details

```
Synthesis failed for the module since it has unsupported
SystemVerilog constructs <constructs>
```

Severity

SynthesisError

SYNTH_5246

Accessing index outside range.

Description

The *SYNTH_5246* rule reports a violation when an expression contains an out-of-range index.

Cases when SYNTH_5246 Cause a Module in Getting Synthesized

The *SYNTH_5246* rule may sometimes cause a module in getting synthesized. Such cases occur if in the RTL, there is a part-select operation in which the direction of the part-select is not in sync with the declared direction of the net.

To check if the module in which the error was reported is synthesized, perform any of the following checks:

- The module name is not present in the list of unsynthesized modules.
- The error message does not indicate the file name or line number.

  If the error results in a black box, the violation will be reported with the file name / line number of the erroneous region in RTL.

Both the above conditions hold true for a synthesized module.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Accessing index outside range

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Modify the code such that the index is within its range.

Example Code and/or Schematic

Consider the following example in VHDL:

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY test is
PORT (
in1   : in std_logic_vector ( 3 downto 0 );
out1 : out std_logic_vector (3 downto 0 )
);
END test;
ARCHITECTURE rtl of test IS
component comp1 is
generic (
SIZE : integer := 2
) ;
port (
a   : in std_logic_vector ( SIZE -1 downto 0 );
y   : out std_logic_vector ( SIZE -1 downto 0 )
);
end component;
begin
I1 : comp1
GENERIC MAP(
SIZE    => 3
)
PORT MAP (
a(31 downto 30)   =>in1(3 downto 2),
a(29)   =>in1(0),
y  => out1(2 downto 0)
);
end rtl;
```

In the above example, the *SYNTH_5246* rule reports a violation because

access to `a(31 downto 30)` and `a(29)` for the input port `a(2 downto 0)[SIZE=3 for instance I1]` is outside the range.

Default Severity Label

SynthesisError

SYNTH_5247

Float parameter used in a non delay type expression.

Description

The *SYNTH_5247* rule reports a violation if a float-type parameter is used in a non delay type expression.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

```
[SynthesisError] Float parameter used in non delay type expression '<expression>'
```

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Do not use the float-type parameter in a non delay type expression.

Example Code and/or Schematic

Consider the following example in Verilog:

```
module mod1 (out1,in1);
output [9:0] out1;
input  [7:0] in1;
parameter p1 = 1.2e3;
assign out1 = in1 + p1;
endmodule
```

In the above example, the *SYNTH_5247* rule reports a violation because

the float parameter `p1` is used in the addition operator `(in1+p1)`, that is a non-delay expression.

Default Severity Label

SynthesisError

SYNTH_5248

Direction of access of a signal is inconsistent with the direction of declaration of signal.

Language

Verilog, VHDL

Rule Description

NOTE: *This rule will be deprecated in a future SpyGlass release. The functionality of this rule is covered by the STX_VE_491 the rule.*

This error is issued when you try to access a signal in Verilog in reverse direction as in the following example:

```
module test(...);
...
  output [1:0] a;
  assign a[0:1] = 2'b01;
...
endmodule
```

Message Details

```
Part Select direction [<num1>:<num2>] is inconsistent with
direction [<num3>:<num4>] of symbol '<symbol>'
```

Severity

SynthesisError

SYNTH_5249

Static range expected.

Description

The *SYNTH_5249* rule reports a violation if the left and right range values of vector nets during declaration cannot be evaluated statically.

Such cases occur if a non static range is used in an expression.

Language

VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] <range> range is not static

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Update the reported range to make it static.

Example Code and/or Schematic

Consider the following example in VHDL:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ent1 is
port( in1 : in std_logic_vector(7 downto 0);
in2 : in natural range 0 to 15;
out1: out std_logic_vector(15 downto 0));
end ent1;
architecture rtl of ent1 is
```

```
function func (arg1 : std_logic_vector; arg2 : natural)
return std_logic_vector is
variable result : std_logic_vector(arg2-1 downto 0) :=
(others => '0');
begin
result(0) := arg1(0);
end func;
begin
process(in1, in2)
begin
out1 <= (others => '0');
out1(in2-1 downto 0) <= func(in1, in2);
end process;
end rtl;
```

In the above example, the *SYNTH_5249* rule reports a violation because the non static input `in2` is passed to `arg2` of the `func` function. This results in a non-static range for the `result` variable for which a static value is expected.

Default Severity Label

SynthesisError

SYNTH_5250

Task/Function contains unsupported SystemVerilog constructs

Description

The *SYNTH_5250* rule reports a violation if a task/function contains a SystemVerilog construct that is not supported by SpyGlass.

Language

Verilog, VHDL

Messages and Suggested Fix

[SynthesisError] This Task/Function contains unsupported SystemVerilog Constructs <constructs>

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove the unsupported SystemVerilog construct from the task/function.

Example Code and/or Schematic

Consider the following example in Verilog:

```
package pack;
function automatic bit signed [7:0] func1 (input bit signed
 [7:0] in1,input in2);
union tagged packed {
bit signed [7:0] bit1;
logic signed [7:0] logic1;
} U1;
case (in2)
1'b1: func1 = U1.bit1;
```

```
1'b0: func1 = U1.logic1;
endcase
endfunction
endpackage
module mod1 (input bit signed [7:0] in1,input bit in2,output
 bit signed [7:0] out1);
always @(in1)
out1=pack::func1(in1,in2);
endmodule
```

In the above example, the *SYNTH_5250* rule reports a violation because of the presence of the unsupported SystemVerilog function tagged union inside the `func1` function.

Default Severity Label

SynthesisError

SYNTH_5251

Illegal range select.

Description

The *SYNTH_5251* rule reports a violation if an out-of-range value is specified in an expression.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5251* rule reports the following message:

[ERROR] Illegal range select. Selection range [<num1>:<num2>] for "<name>" is out of range [<num3>:<num4>] in expression: "<expression>"

Consequences of Not Fixing

If you do not fix this violation, the expression containing out-of-range values are ignored from SpyGlass analysis.

How to Debug and Fix

To fix this violation, specify a correct range in an expression.

Example Code and/or Schematic

Consider the following example:

```
module test(in1, out1);
input [4:0]in1;
output [3:0]out1;
reg [3:0]out1;
always
out1 =in1[5:2];
endmodule
```

In the above example, the part-select range `[5:2]` for the `in1` port is out

of the declared range `[4:0]` of `in1`.

Default Severity Label

SynthesisError

SYNTH_5252

'time' declarations inside SystemVerilog constructs are not supported by synthesis

Description

The *SYNTH_5252* rule reports a violation when `time` declarations are specified inside SystemVerilog constructs, such as `struct`, `union`, `typedef`, and `enum`.

Language

Verilog

Messages and Suggested Fix

The *SYNTH_5252* rule reports the following message:

[ERROR] `TIME declarations ('<declarations>') inside SV Constructs are not supported by synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, remove `time` declarations from SystemVerilog constructs.

Example Code and/or Schematic

Consider the following example:

```
module test;
  typedef struct {
  int a;
  time b;
  } data_t;
```

```
  data_t data_array [23:0];
endmodule
```

In the above example, the *SYNTH_5252* rule reports a violation for the `time` declaration inside the `typedef` SystemVerilog construct.

Default Severity Label

SynthesisError

SYNTH_5253

REAL type parameters are not supported by synthesis

Description

The *SYNTH_5253* rule reports a violation if real-type parameters are used in code.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] REAL type parameters ('<parameters>') are not supported by synthesis

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove the reported real-type parameter from the code.

Example Code and/or Schematic

Consider the following example in Verilog;

```
module top (input bit [15:0] in1,output bit[15:0] out1);
parameter shortreal p = 2.5;
m1 I1 ((shortint'(p)),in1,out1);
endmodule
#######################################################

module m1 ( input int in1, input logic[15:0] in2, output
logic[15:0] out1);
```

```
assign     out1 = in1 & in2;
endmodule
#######################################################
```

In the above example, the *SYNTH_5253* rule reports a violation because the `p` parameter of the type shortreal is passed in the instantiation of the `m1` module.

Default Severity Label

SynthesisError

SYNTH_5254

'real' declarations inside SystemVerilog constructs are not supported by synthesis

Description

The *SYNTH_5254* rule reports a violation when, `real` declarations are specified inside SystemVerilog constructs, such as `struct`, `union`, `typedef`, and `enum`

Language

Verilog

Messages and Suggested Fix

The *SYNTH_5254* rule reports the following message:

[ERROR] `REAL declarations inside SV Constructs are not supported for synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, remove `real` declarations from SystemVerilog constructs.

Example Code and/or Schematic

Consider the following example:

```
module test;
  typedef struct {
  int a;
  real b;
  } str1;
```

```
  str1 data_array [3:0];
endmodule
```

In the above example, the *SYNTH_5254* rule reports a violation for the `real` declaration inside the `typedef` SystemVerilog construct.

Default Severity Label

SynthesisError

SYNTH_5255

Out-of-range bit-select operations

Description

The *SYNTH_5255* rule reports a violation for the bit-select operations that are out-of-range.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] Illegal bit select. Index <index> is out of range [<low>]:[<high>] in expression: "<expression>"

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Modify the code such that the reported index is within its specified range.

Example Code and/or Schematic

Example 1

Consider the following module:

```
module test (in1, out1);
    input [3:0] in1;
    output out1;
    assign out1 = in1[4];
endmodule
```

In the above example, `in1[4]` is out-of-range as `in1` is declared within the range of `[3:0]`.

Example 2

Consider the following example:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity test is
port (
    in1: in std_logic_vector(3 downto 0);
    out1: out std_logic
);
end test;
architecture rtl of test is
begin
    out1 <= in1(4);
end rtl;
```

In the above example, `in1[4]` is out-of-range as `in1` is declared within the range of `[3:0]`.

Default Severity Label

SynthesisError

SYNTH_5256

Signal used in the condition of an if statement must be compared to '1' or '0' only

Description

The *SYNTH_5256* rule reports a violation if a signal in an `if` condition is compare with a value other than 1 or 0.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5256* rule reports the following message:

[ERROR] Signal edge of "<variable/signal-name>" used in condition of if statement may have comparison to '1' or '0' only.

Consequences of Not Fixing

If you do not fix this violation, the condition containing the reported signal is ignored during synthesis.

How to Debug and Fix

To fix this violation, compare the signal with either 1 or 0 in an `if` condition.

Example Code and/or Schematic

Consider the following example:

```
module test(reset,clk,in1,out1);
  parameter p = 3;
  input reset,clk,in1;
  output out1;
  reg out1;
  always @(posedge clk or negedge reset)
```

```
  begin
    if(reset == p)
      out1 <= 1'b0;
    else
      out1 <= in1;
  end
endmodule
```

For the above example, the *SYNTH_5256* rule reports a violation because `reset` is compared with `p` in the `if` statement `if(reset == p)`.

Default Severity Label

SynthesisError

SYNTH_5257

Part select on a vector is out of range

Description

The *SYNTH_5257* rule reports a violation if the part select on a vector is out of range.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Part Select [<num1>:<num2>] on a Vector <name>[<num3>:<num4>] is out of range

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Modify the reported part select to specify the correct range.

Example Code and/or Schematic

Consider the following example in Verilog:

```
module test(in1, out1);
  input [4:0]in1;
  output [3:0]out1;
  assign out1 =in1[5:2];
endmodule
```

In the above example, the *SYNTH_5257* rule reports a violation because the part-select `[9:6]` for the `in1` input is outside the range of `[7:0]`.

Default Severity Label

SynthesisError

SYNTH_5258

Bit Select on a Vector is out of range.

Language

Verilog, VHDL

Message Details

```
Bit Select [<bit-select>] on a Vector <name>[<num1>:<num2>] is
out of range.
```

Severity

SynthesisError

SYNTH_5259

A particular bit of Part select operation is out of range

Language

Verilog, VHDL

Message Details

The <num>th bit of the Part select [<num1>:<num2>] on a Vector [<num3>:<num4>] is out of range

Severity

SynthesisError

SYNTH_5260

Usage of a SystemVerilog string in the design.

Description

The *SYNTH_5260* rule reports this violation to indicate the usage of a SystemVerilog `string` in a design.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `STRING data type not supported for synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove the usage of `string` from the design.

Example Code and/or Schematic

Consider the following example in Verilog:

```
module test1 (input in1,output out1);
  assign out1 = func1(in1);
  function string func1 (input in1);
  func1 = in1;
  endfunction
endmodule
```

In the above example, the *SYNTH_5260* rule reports a violation because the return type of the `func1` function is `string`.

Default Severity Label

SynthesisError

SYNTH_5261

Implicit state machines not supported.

Language

Verilog, VHDL

Message Details

```
Implicit state machines not supported
```

Severity

SynthesisError

SYNTH_5262

Module instance name not specified.

Language

Verilog, VHDL

Message Details

`Module instance name not specified`

Severity

SynthesisError

SYNTH_5264

Net type 'TIME' is not supported

Description

The SYNTH_5264 rule reports a violation to indicate the usage of net with type as time construct. Such constructs are not synthesizable by SpyGlass.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Net type 'TIME' is not supported

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the time constructs from the code.

Example Code and/or Schematic

The following example shows the scenario in which this rule reports a violation:

```
module test(out1);
output time  out1[1:0] ;
wire time w1[1:0];
assign w1[1:0] = '{'1,'1};
assign out1[0] = w1[0];
endmodule
```

In the above example, the *SYNTH_5264* rule reports a violation because of the usage of the net `w1` with its type as time.

Default Severity Label

SynthesisError

SYNTH_5266

File specified recursively.

Language

Verilog, VHDL

Message Details

`Recursion in file specification`

Severity

SynthesisError

SYNTH_5267

Design unit not found.

Language

Verilog, VHDL

Message Details

```
Design unit <design-unit> not found
```

Severity

SynthesisError

SYNTH_5271

'wait' statement without a condition is not supported for synthesis.

Language

VHDL

Message Details

```
'wait' statement without a condition is not supported for
synthesis
```

Severity

SynthesisError

SYNTH_5272

Master module not defined.

Language

Verilog, VHDL

Message Details

```
Master module <module> not defined.
```

Severity

SynthesisError

SYNTH_5273

Number of bits for net/variable exceeds the mthresh value

Description

The *SYNTH_5273* rule reports a violation if the number of bits of a net/variable/array exceeds the specified number (by default, 4096) of bits.

Reducing the Number of Bits

Specify the `set_option handlememory 1` project-file command to reduce the number of bits, which might not have any detrimental effect in most cases of rule-checking (at least for regular structures).

If you want to synthesize all those bits, increase the limit by using the `set_option mthresh <num>` project-file command.

Rule Exceptions

The *SYNTH_5273* rule does not report a violation for single-dimensional arrays or vector declarations even if they are exceeding the threshold value.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] Number of bits (<num>) for net/variable '<name>' is greater than mthresh value <value>. Set the 'mthresh' option to a value greater than the indicated number of bits

Consequences of Not Fixing

SpyGlass does not synthesize the modules containing such net/variable/array. This is done to prevent a very high count of a regular structure.

How to Debug and Fix

If the *SYNTH_5273* violation appears for the variable that is declared but not used in the design, do either of the following to remove this violation:

- Increase the *mthresh* value to what is suggested in the violation message.
- Remove the declaration.

Example Code and/or Schematic

Example 1

```
reg [5:0] arr [3:0][3:0][63:0]
```

In this example, the total number of bits is 6144 (4x4x64x6). As 6144 is greater than the default threshold value (4096 bits), the design unit will not be synthesizable.

Example 2

```
typedef struct packed {
  int a; // 32 bits
  reg [15:0] r [3:0][15:0]; // 1024 bits (4x16x16)
  logic [1:0][5:0] arrLogic [255:0]; // 3072 bits
                                     // (256x2x6)
} St;
St data; // data is variable
```

In this example, the total number of bits is 4128 (32 + 1024 + 3072). As 4128 is greater than the default threshold value (4096 bits), the design unit will not be synthesizable.

Example 3

```
type Full_dport_range is range 0 to 63;
type Node_id_type is range 0 to 11 type Dport_conn_array is
array (Full_dport_range) of std_logic_vector (7 downto 0);

type Sys_conn_array is array (Node_id_type) of
```

```
Dport_conn_array;

signal S : Sys_conn_array
```

In this example, the total number of bits in signal, `S`, are 6144 (64x8x12). As 6144 is greater than the default threshold value (4096 bits), the design unit will not be synthesizable.

Example 4

```
type Sys_info_record is record
  conn1: Sys_conn_array; // 6144 bits
  conn2: Dport_conn_array; // 512 bits
end record;

variable var : Sys_info_record
```

In this example, the total number of bits in variable, `var`, is 6656 (6144 + 512). As 6656 is greater than the default threshold value (4096 bits), the design unit will not be synthesizable.

NOTE: *In the above examples, you can make a design unit synthesizable by specifying the threshold value greater than equal to the number of bits of net/variable, but at the cost of an increase in the memory and time of synthesis.*

Default Severity Label

SynthesisError

SYNTH_5274

Port not found in the master.

Description

The *SYNTH_5274* rule reports a violation if an instance contains a port that does not exist in the instance master.

This violation appears only with the escaped ports names in a design when the port name is not the exact match with what is available in the master.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5274* rule reports the following message:

```
[ERROR] Port '<port>' not found in the master.
```

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

To fix this violation, specify a port that exists in the instance master.

Example Code and/or Schematic

Consider the following example:

```
module top (in, out);
  input in;
  output out;
  test inst(.in(in), .out(out));
endmodule

module test(out);
```

```
  output out;
endmodule
```

For the above example, the *SYNTH_5274* rule reports a violation because there is no `in` port in the master (`test`) of the `top` module.

Default Severity Label

SynthesisError

SYNTH_5275

Event control list missing at the beginning of the always block is unsynthesizable

Description

The *SYNTH_5275* rule reports a violation if an event-control list is missing at the beginning of an `always` block.

Language

Verilog

Messages and Suggested Fix

```
@(event_control_list) missing at the beginning of the always block. Not synthesizable
```

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Place an event-control statement at the beginning of an `always` block.

Example Code and/or Schematic

Consider the following example in Verilog:

```
module test(g_rst, rst,in1,out1);
input g_rst, rst,in1;
output out1;
reg out1;
always
begin
if (g_rst)
out1 <= 0;
```

```
@(rst or in1)
if (rst==0)
out1 <=0;
else
out1 <=in1;
end
endmodule
```

In the above example, the *SYNTH_5275* rule reports a violation because the event-control statement `@(rst or in1)` is used inside the `always` block and not at the beginning of the `always` block.

Default Severity Label

SynthesisError

SYNTH_5276

Multiple event_control_lists with level sensitive signals within an always block is not synthesizable.

Description

The *SYNTH_5276* rule reports a violation if an `always` block contains multiple event-control statements.

Language

Verilog

Messages and Suggested Fix

[SynthesisError] `Multiple event_control_lists with level sensitive signals within an always block is not synthesizable`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Update the code to have only one event-control statement in an `always` block. Multiple event-control statements in an `always` block are not synthesizable.

Example Code and/or Schematic

Consider the following Verilog example:

```
module test(rst,clk,in1, in2, a, b, out1);
input rst,clk,in1, in2, a, b;
output out1;
reg out1;
always
begin
```

```
@(clk or rst or in1)
if (rst==0)
out1 <=0;
else
out1 <=in1;
@(a or b or in2)
out1 <= in2;
end
endmodule
```

In the above example, the *SYNTH_5276* rule reports a violation because of the presence of multiple event-control statements `@(clk or rst or in1)` and `@(a or b or in2)` in the `always` block.

Default Severity Label

SynthesisError

SYNTH_5277

Clocked element inside VHDL functions is not supported for synthesis.

Description

The *SYNTH_5277* rule reports a violation when the constructs, such as `rising_edge(clk)`, `falling_edge(clk)`, and `clk'EVENT` are used inside VHDL functions.

Such constructs are not allowed in VHDL functions because VHDL functions can represent only combinational logic.

Language

VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Clocked Elements inside VHDL Function are not supported for synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, remove the timing control statements from functions.

Example Code and/or Schematic

Example 1

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;
entity test1 is
```

```
port(clk , d :in std_logic;
q : out std_logic);
end test1;
architecture rtl of test1 is
function func1(signal d,clk : in std_logic)
return std_logic is
begin
if( clk'event and clk='1') then
return d;
end if;
end func1;
begin
q <= func1(d,clk);
end rtl;
```

For the above example, the *SYNTH_5277* rule reports a violation because `clk'event` is used inside the `myfunc` function.

Example 2

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;
entity test2 is
port(clk, d, rst :in std_logic;
q : out std_logic);
end test2;
architecture rtl of test2 is
function myfunc(signal d,clk,rst : in std_logic)
return std_logic is
variable var : std_logic;
begin
if(rst='1') then
var := '0';
elsif (rising_edge (clk)) then
var := d;
end if;
```

```
return var;
end function;
begin
q <= myfunc(d,clk,rst);
end rtl;
```

For the above example, the *SYNTH_5277* rule reports a violation due to usage of because `rising_edge(clk)` is used inside the `func` function.

Default Severity Label

SynthesisError

SYNTH_5278

Only If statements are handled inside asynchronous reset/clear always block.

Language

Verilog, VHDL

Message Details

```
Only If statements are handled inside asynchronous reset/clear
always block
```

Severity

SynthesisError

SYNTH_5279

Expression type not supported for TRAN terminal.

Language

Verilog, VHDL

Message Details

`Expression type not supported for TRAN terminal.`

Severity

SynthesisError

SYNTH_5280

'<TypeDef | Array | Enum>' of '<TIME | REAL | EVENT | REALTIME | SHORTREAL | SV STRING | CHANDLE>' is not synthesizable

Description

The *SYNTH_5280* rule reports a violation if the SV constructs, such as time, SV-string, event, real, or short-real is used in association with the SV constructs, such as typedef, structure, union, array, or enum.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] `'<TypeDef | Array | Enum>' of '<TIME | REAL | EVENT | REALTIME | SHORTREAL | SV STRING | CHANDLE>' is not synthesizable`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Do not use the SV constructs, such as time, SV-string, event, real, or short-real is used in association with the SV constructs, such as typedef, structure, union, array, or enum

Example Code and/or Schematic

This rule reports a violation in the following example because the event, real, and time constructs are used with the typedef constructs:

```
typedef event myevent;
typedef real myreal;
typedef time mytime;
```

```
module top6718;
 myevent i;
endmodule

module top6712;
 myreal i;
endmodule

module top6709;
 mytime i;
endmodule
```

Default Severity Label

SynthesisError

SYNTH_5281

Width greater than '4000000' is not supported for declaration.

Description

The *SYNTH_5281* rule reports a violation if the width in a declaration exceeds *4000000*.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5281* rule reports the following message:

[ERROR] `Width '<name>[<num1>:<num2>]' greater than '4000000' is not supported for declaration.`

Consequences of Not Fixing

If you do not fix this violation, the declaration with a width greater than 4000000 is ignored during synthesis.

How to Debug and Fix

To fix this violation, specify a width less than 4000000 in a declaration.

Example Code and/or Schematic

Consider the following example:

```
module test();
  bit [0:499] [0:99] tmp1 [0:9] [0:9];
endmodule
```

For the above example, the *SYNTH_5281* rule reports a violation for `tmp1` as the size of `tmp1` is 5000000.

Default Severity Label

SynthesisError

SYNTH_5282

Non synthesizable construct: event declaration.

Language

Verilog, VHDL

Message Details

`Non synthesizable construct: event declaration`

Severity

SynthesisError

SYNTH_5283

Non synthesizable construct. Floating point declaration.

Description

The *SYNTH_5283* rule reports a violation if a design contains a non synthesizable construct that SpyGlass cannot synthesize.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Non synthesizable construct : floating point <num> declaration.

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove the non synthesizable construct from the design.

Example Code and/or Schematic

Consider the following example in Verilog:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
port (
in1 : in real    := -1.0 ;
out1 : out real
);
end test;
```

```
architecture rtl of test is
begin
out1<= in1;
end ;
```

In the above example, the *SYNTH_5283* rule reports a violation because the in1 input and the output is declared of the type `real`. The `real` type constructs are not synthesizable by SpyGlass.

Default Severity Label

SynthesisError

SYNTH_5284

Non synthesizable construct: floating point type constant.

Description

The *SYNTH_5284* rule reports a violation if the type of a constant is a float value.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Non synthesizable construct: floating point type constant.

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Specify the constant type that is synthesizable by SpyGlass.

Example Code and/or Schematic

Consider the following example in Verilog:

```
module test(in1,out1);
input [3:0] in1;
output [3:0] out1;
reg [8:0] out1;
always @(in1)
case(in1)
2.3 : out1 = 2;
endcase
```

```
endmodule
```

In the above example, the *SYNTH_5284* rule reports a violation because the floating-point value `2.3` is used as the `case` item expression.

Default Severity Label

SynthesisError

SYNTH_5285

System functions not synthesizable.

Description

The *SYNTH_5285* rule reports a violation if system functions are used in the code.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] System function '<function>' is not synthesizable

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove the reported system function from the code.

Example Code and/or Schematic

Consider the following example in Verilog:

```
module test(input [2:0] depth, output int dim);
bit [0:5] bitarray [2:5];
always_comb
case ($length1(bitarray,1))
1: dim = 2;
default: dim = -1;
endcase
endmodule
```

In the above example, the *SYNTH_5285* rule reports a violation because of the usage of the system function `$length1`.

Default Severity Label

SynthesisError

SYNTH_5286

Passing invalid Interface object as module port

Language

Verilog, VHDL

Message Details

```
Passing invalid Interface object '<object>' as module port
```

Severity

SynthesisError

SYNTH_5287

Usage of 'Clocking Blocks' in not synthesizable

Language

Verilog, VHDL

Message Details

`Usage of 'Clocking Blocks' in not synthesizable`

Severity

SynthesisError

SYNTH_5288

Usage of 'event' is not synthesizable

Description

The *SYNTH_5288* rule reports a violation to indicate the usage of the `event` keyword in the code.

Language

Verilog

Messages and Suggested Fix

`[SynthesisError]` `Usage of 'event' is not synthesizable`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove the `event` keyword from the code.

Example Code and/or Schematic

Consider the following example in Verilog:

```
module test(input a, b, output c);
event e;
event e1;
always
e1 =e;
endmodule
```

In the above example, the *SYNTH_5288* rule reports a violation because of the usage of the `event` keyword in the `test` module.

Default Severity Label

SynthesisError

SYNTH_5290

Usage of 'Real' is not synthesizable

Description

The *SYNTH_5290* rule reports a violation to indicate the usage of the `real` construct in the code.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Usage of 'Real' is not synthesizable`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove the `real` construct from the code.

Example Code and/or Schematic

Consider the following example:

```
# Verilog Top Module #
#####################################################
module top (input bit [15:0] in1,output bit[15:0] out1);
localparam real p = 2.5;
m1 I1 (p,in1,out1);
endmodule
#####################################################

# Verilog Entity #
```

```
#######################################################
module m1 ( input int in1, input logic[15:0] in2, output
logic[15:0] out1);
assign     out1 = in1 & in2;
endmodule
#######################################################
```

In the above example, the *SYNTH_5290* rule reports a violation because the `p` parameter of the `real` type is used in the instantiation of the `m1` module.

Default Severity Label

SynthesisError

SYNTH_5291

Named block variables not supported in asynchronous always.

Language

Verilog, VHDL

Message Details

`Named block variables not supported in asynchronous always`

Severity

SynthesisError

SYNTH_5292

Illegal use of unknowns in case expression.

Language

Verilog, VHDL

Message Details

`Illegal use of unknowns in case expression`

Severity

SynthesisError

SYNTH_5293

Indexing into a scalar variable.

Language

Verilog, VHDL

Message Details

`Indexing into a scalar variable`

Severity

SynthesisError

SYNTH_5294

Unrollable repeat statement.

Language

Verilog

Rule Description

The expression, `expr`, in the `repeat` statement, `repeat (expr)`, should evaluate to a constant. If this value cannot be determined, the synthesis of `repeat` statement cannot happen. This is an error as per the LRM. In such cases, the RTL should be modified so that the `repeat` expression evaluates to a constant.

Message Details

```
Unrollable repeat statement
```

Severity

SynthesisError

SYNTH_5295

Unrollable while loop.

Language

Verilog, VHDL

Rule Description

If the while-loop condition expression does not evaluate to a constant value, the while-loop cannot be processed.

Message Details

`Unrollable while loop`

Severity

SynthesisError

SYNTH_5296

Sequential while/repeat loops are not supported for synthesis.

Language

Verilog, VHDL

Message Details

`Sequential while/repeat loops are not supported for synthesis`

Severity

SynthesisError

SYNTH_5297

Could not calculate choice value.

Description

The *SYNTH_5297* rule reports a violation if the choice value cannot be calculated.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] `Could not calculate choice value`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Modify the code such that the choice value can be calculated.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ent is
port (
in1 : in std_logic;
out1 : out std_logic
);
end ent;
architecture rtl of ent is
type arr is array (0 to 1) of bit;
```

```
signal sig: arr;
signal s1: integer;
begin
sig <= (s1=>'1');
end rtl;
```

Default Severity Label

SynthesisError

SYNTH_5298

Ports in blocks not supported.

Language

Verilog, VHDL

Message Details

`Ports in blocks not supported.`

Severity

SynthesisError

SYNTH_5299

Mismatch in the port width of the net passed and the original port declaration

Language

Verilog, VHDL

Message Details

```
Mismatch in port width of '<net>' during array of interface
instantiation
```

Severity

SynthesisError

SYNTH_5300

This rule has been deprecated and port connection expression in an array of interface is supported by default in SpyGlass

SYNTH_5301

Module name matches with that of built in primitive.

Language

Verilog, VHDL

Message Details

`Module name <name> matches with that of built in primitive.`

Severity

SynthesisError

SYNTH_5302

Non-static while loop cannot be unrolled and hence not synthesizable.

Language

Verilog, VHDL

Message Details

```
Non-static while loop cannot be unrolled and hence not
synthesizable.
```

Severity

SynthesisError

SYNTH_5303

Usage of 'TIME' is not synthesizable

Description

The *SYNTH_5303* rule reports a violation to indicate the usage of the `time` construct. Such constructs are not synthesizable by SpyGlass.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Usage of 'TIME' is not synthesizable`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove the time constructs from the code.

Example Code and/or Schematic

Consider the following example in Verilog:

```
module test (in1,in2,clk,out1);
input [31:0] in1,in2;
input clk;
output [63:0] out1;
reg [63:0] out1;
always @(posedge clk)
begin : block
time time_var;
time_var = { in1,in2};
```

```
out1 = time_var;
end
endmodule
```

In the above example, the *SYNTH_5303* rule reports a violation because of the usage of the `time` variable `time_var`.

Default Severity Label

SynthesisError

SYNTH_5304

Usage of REALTIME is not synthesizable

Description

The *SYNTH_5304* rule reports a violation to indicate the usage of the `realtime` construct. Such constructs are not synthesizable by SpyGlass.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

`[SynthesisError]` `Usage of REALTIME is not synthesizable`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How To Debug and Fix

Remove the `realtime` constructs from the code.

Example Code and/or Schematic

`Consider the following example in Verilog:`

```
module test(in1,out1);
input [63:0] in1;
output [63:0] out1;
realtime t;
assign out1=t;
endmodule
```

In the above example, the *SYNTH_5304* rule reports a violation because of

the usage of the `realtime` variable `t`.

Default Severity Label

SynthesisError

SYNTH_5305

`quiet attribute not supported for synthesis.

Description

The *SYNTH_5305* rule reports a violation when the `quiet attribute is used in the code.

Language

VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `quiet attribute not supported for synthesis

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the `quiet attribute from the code.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity pre_attrb3 is
port(in1 : integer;
clk : bit;
out1: out integer
);
end pre_attrb3 ;
architecture pre_attrb3 of pre_attrb3 is
begin
process
begin
wait until clk'quiet;
```

```
out1<= in1;
end process;
end;
```

In the above example, the *SYNTH_5305* rule reports a violation because the `quiet attribute is used in the expression clk`quiet.

Default Severity Label

SynthesisError

SYNTH_5306

Named task or block is not in the scope of a disable statement.

Description

The *SYNTH_5306* rule reports a violation if a block or a task is used in a `disable` statement.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Named task or block is not in scope of disable statement`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the task/block from the `disable` statement.

Example Code and/or Schematic

Consider the following example of a Verilog module:

```
module test(in1,in2,out1);
input [7:0] in1,in2;
output [7:0] out1;
reg [7:0] out1;
integer i;
task task1;
input [7:0] in1,in2;
output [7:0] out1;
out1=in1 & in2;
```

```
endtask
always @(in1 or in2)
begin
disable task1;
task1(in1,in2,out1);
end
endmodule
```

In the above example, the *SYNTH_5306* rule reports a violation because the `task1` task is used with the `disable` statement.

Default Severity Label

SynthesisError

SYNTH_5307

Usage of 1'bx / 1'bz in an expression is not synthesizable

Language

Verilog

Message Details

```
Usage of 1'bx/1'bz in expression <expression> is not
synthesizable
```

Severity

SynthesisError

SYNTH_5308

Variable used in a task or a function is not present in the modport list of a modport.

Description

The *SYNTH_5308* rule reports a violation if the variable used in a task or a function, which is called through a modport, is missing in the list of that modport.

Language

Verilog

Messages and Suggested Fix

The *SYNTH_5308* rule reports the following message:

```
[ERROR] Variable '<variable>' used in Task/Function '<name>'
(called through modport) is not present in the modport list of
modport '<modport>'
```

Consequences of Not Fixing

If you do not fix this violation, the reported variable is ignored from synthesis.

How to Debug and Fix

To fix this violation, specify a variable that is present in the list of the reported modport.

Example Code and/or Schematic

Consider the following example:

```
interface intf(input int data[1:0], output int q);
  logic [1:0]data;
  logic q;
  task add;
    q = data[0] + data[1];
```

```
  endtask
  modport m1(output q, import task add());
endinterface
module test(intf.m1 obj);
  always_comb obj.add;
endmodule
```

For the above example, the *SYNTH_5308* rule reports a violation because `data` that is used in the `add` task is not in the list of modport, and therefore, it will not be accessible to the `add` task.

However, if you specify `input data` in the modport list, synthesis will occur correctly.

Default Severity Label

SynthesisError

SYNTH_5309

Slice name indices should be static.

Language

Verilog, VHDL

Message Details

`Slice name indices should be static`

Severity

SynthesisError

SYNTH_5310

Undeterminable value for slice name index.

Description

The *SYNTH_5310* rule reports a violation when the value of an index cannot be determined.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Undeterminable value for slice name index in the range [<num1>:<num2>] for net <name>

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Correct the code so that the value of the index can be determined.

Example Code and/or Schematic

Consider the following example: of a VHDL entity:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity test is
port (
in1 : in std_logic_vector(15 downto 0);
out1 : out std_logic_vector(3 downto 0)
);
end test;
architecture rtl of test is
```

```
signal i : integer;
constant WIDTH : integer := 4;
begin
out1 <= in1(i*(WIDTH+1)-1 DOWNTO i*WIDTH);
end rtl;
```

In the above example, the *SYNTH_5310* rule reports a violation because the `i` signal is non-static and the slice index values `(5i-1-4i)` for `in1` are dependent on the values of `i`, and therefore, undeterminable.

Default Severity Label

SynthesisError

SYNTH_5311

Index trying to access out of range.

Description

The *SYNTH_5311* rule reports a violation if an index is specified with a range beyond its maximum limit.

Language

VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Index <index> trying to access out of range.

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Specify a correct range for the reported index.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity ent1 is
port(
in1 : in std_logic_vector(3 downto 0);
out1 : out boolean
);
end ent1;
architecture rtl of ent1 is
type arr1 is array (integer range <>) of bit;
subtype st1 is arr1 (-2 to 1);
function FUNC1 (
```

```
in2,in3 : in arr1
) return boolean is
begin
return (in3(-1 to 0) = in2(0 to 1));
end;
signal sig1 : st1;
begin
process( sig1)
begin
out1 <= FUNC1(sig1, sig1(0 to 1));
end process;
end rtl;
```

In the above example, the *SYNTH_5311* rule reports a violation because `sig1(0 to 1)` is passed as an argument to the `in3` input of the `FUNC1` function and is used as `in3(-1 to 0)`, which is out of range.

Default Severity Label

SynthesisError

SYNTH_5312

For signal of composite type, not all scalar sub-elements have equal number of drivers -- LRM Section 4.3.1.2

Language

VHDL

Message Details

```
For signal <signal> of composite type, not all scalar sub-
elements have equal number of drivers -- LRM Section 4.3.1.2
```

Severity

SynthesisError

SYNTH_5313

reg / nets / arrays / memories defined inside packages or root scope are not synthesizable

Description

The *SYNTH_5313* rule reports a violation if registers, nets, arrays, or memories are defined inside a package.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Variable declaration '<declaration>' inside <package> is not supported for synthesis

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the reported declaration from the reported package.

Example Code and/or Schematic

Consider the following example of a Verilog module:

```
package pack1;
var bit [3:0] var1;
endpackage
module test (in1,out1);
input [3:0] in1;
output [3:0] out1;
import pack1::*;
assign var1 = in1;
assign out1 = var1;
endmodule
```

In the above example, the *SYNTH_5313* rule reports a violation because `var1` of the type `var` is declared inside the `pack1` package.

Default Severity Label

SynthesisError

SYNTH_5314

Subprogram body not defined.

Description

The *SYNTH_5314* rule reports a violation if a function is declared but no body is defined for the same.

Language

Verilog, VHDL

Message and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Subprogram body for <name> is not defined.`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Either remove the function declaration or define a body for the function.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library IEEE;
use IEEE.std_logic_1164.all;
package pkg is
function func1 (arg:std_logic_vector) return
std_logic_vector;
end package;
library IEEE;
use IEEE.std_logic_1164.all;
use work.pkg.all;
entity test is
port (
in1     : in    std_logic_vector (3 downto 0);
```

```
out1     : out    std_logic_vector(3 downto 0)
);
end test;
architecture rtl of test is
begin
out1 <= func1(in1);
end rtl;
```

In the above example, the *SYNTH_5314* rule reports a violation because the body of the `func1` function is not defined. Only the declaration of `func1` is defined in the `pkg` package.

Default Severity Label

SynthesisError

SYNTH_5315

Non static access in generate block name referencing is not supported for synthesis

Description

The *SYNTH_5315* rule reports a violation if a non-static value is used in *generate* block name referencing.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Non static access (<non-static-expression-string>), in generate block name referencing is not supported for synthesis

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the non static expression from the `generate` block.

Example Code and/or Schematic

Consider the following example of a Verilog entity:

```
module m1 (in1,sel,out1);
input [3:0] in1;
output [3:0] out1;
input sel;
genvar j;
generate for (j=0;j<2;j=j+1) begin : gen1
wire [3:0] net1;
assign gen1[j].net1 = in1;
assign out1 =gen1[sel].net1 ;
```

```
end
endgenerate
endmodule
```

In the above example, the *SYNTH_5315* rule reports a violation because the non-static input signal `sel` is used in the `generate` block name referencing `(gen1[sel])`.

Default Severity Label

SynthesisError

SYNTH_5316

Unsynthesizable constructs inside interface.

Description

The *SYNTH_5316* rule reports a violation if an unsynthesizable construct is present inside an interface.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] '<constructs>' inside interfaces is not synthesizable

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the reported construct from the interface.

Example Code and/or Schematic

Consider the following example of a Verilog module:

```
interface intf1;
shortreal w4;
modport master (output w4);
endinterface
module top(input in1, input in2, output out);
intf1 i1();
mid1 INST1(i1.master);
endmodule
module mid1(intf1 i1);
always
i1.w4 = 1;
endmodule
```

In the above example, the *SYNTH_5316* rule reports a violation because of the usage of non-synthesizable construct `shortreal` inside an interface.

Default Severity Label

SynthesisError

SYNTH_5317

Always block that has both timing control statement and embedded event expression. This is not supported by synthesis.

Description

The *SYNTH_5317* rule reports a violation if an `always` block contains both timing control statement and embedded event expression.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Always block that has both a timing control statement as well as embedded event (@) expression is not supported by synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove either timing control statement or embedded event expression from the `always` block.

Example Code and/or Schematic

Consider the following example of a Verilog entity:

```
module test(out1,sel,in1,in2);
output  out1;
reg out1;
input sel,in1,in2;
always@ ( sel or in1 ,in2)
begin
case (sel)
2'b00 : out1 = in1;
2'b01 : out1 = in2;
```

```
endcase
@(in1 or in2) out1 = 2'b11;
end
endmodule
```

In the above example, the *SYNTH_5317* rule reports a violation because both timing control statement and embedded event expression is used in the `always` block.

Default Severity Label

SynthesisError

SYNTH_5319

Clocks found in multiple branches.

Description

The *SYNTH_5319* rule reports a violation if clock events are found at multiple branches.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Clocks in multiple branches`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Keep only one clock event in the module.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity test is
port ( in1,in2 : in std_logic;
clk,en : in std_logic;
out1 : out std_logic);
end test;
architecture rtl of test is
begin
process (clk)
begin
if (en = '0') then
if (clk'event and clk = '1') then
```

```
out1 <= in1;
end if;
elsif (en = '1') then
if (clk'event and clk = '1') then
out1 <= in2;
end if;
end if;
end process;
end rtl;
```

In the above example, the *SYNTH_5319* rule reports a violation because out1 is driven in multiple clock events.

Default Severity Label

SynthesisError

SYNTH_5320

Assignment of a particular signal at both if and else part of clocked event or assignment before clocked event is not synthesizable.

Description

The *SYNTH_5320* rule reports a violation in the following cases:

- When there is an assignment of a particular signal at both TRUE and FALSE paths of `event.
- When there is an assignment of a particular signal before `event and at `event.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Assignment of '<signal>' before Clocked event or at the else part of if-else is not synthesizable

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the assignment of the reported signal from the clocked event.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
port (clk1,d: in BIT;
out1: out BIT);
end test ;
architecture rtl of test is
```

```
begin
process (clk1,d)
begin
out1 <= '0' ;
if clk1'EVENT and clk1 = '0' then
out1 <= d ;
end if ;
end process;
end rtl ;
```

In the above example, the *SYNTH_5320* rule reports a violation because `out1` is also driven before the `clk'` event as well.

Default Severity Label

SynthesisError

SYNTH_5321

Clocked if-else construct is not synthesizable to standard logic

Description

Consider the following example:

```
process (rst, clk1, clk2, d1, d2)
  begin
  if (rst = '1') then
    q2 <= '0';
  elsif (clk2'event and clk2 = '1') then
    q2 <= d2;
  else
    q3 <= q1;
  end if;
end process;
```

In the above example, the `else` statement after CLOCK is not synthesizable.

Language

Verilog, VHDL

Message Details

`Clocked if-else construct is not synthesizable to standard logic`

Severity

SynthesisError

SYNTH_5322

Multiple wait statements used which are not identical

Description

The *SYNTH_5322* rule reports a violation if a `wait` statement is used multiple times with different conditions.

Language

VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Multiple wait statements used which are not identical`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Keep one condition in a `wait` statement.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library ieee;
use ieee.std_logic_1164.all;
entity ent is
port (d1, d2     : in  std_logic;
clk1, clk2 : in  std_logic;
q          : out std_logic);
end ent;
architecture arc of ent is
begin
process

begin
```

```
wait until clk1'event and clk1 = '1';
q <= d1;
wait until clk1'event and clk1 = '0';
q <= d2;
end process;
end arc;
```

In the above example, the *SYNTH_5322* rule reports a violation because the `wait` statement is used multiple times with different conditions.

Default Severity Label

SynthesisError

SYNTH_5323

Non synthesizable wait statement.

Description

The *SYNTH_5323* rule reports a violation for a non synthesizable `wait` statement.

Language

VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Non-synthesizable Wait statement`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Correct the logic of the `wait` statement to make it synthesizable or remove the `wait` statement from the code.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity test is
port (in1,in2: in std_logic;
out1 : out std_logic
);
end test;
architecture arch of test is
begin
process
```

```
begin
out1 <= in1;
wait;
out1 <= in2;
end process;
end arch;
```

In the above example, the *SYNTH_5323* rule reports a violation because the `wait` statement is specified without any condition, and therefore, it is not synthesizable.

Default Severity Label

SynthesisError

SYNTH_5324

Non-synthesizable statement.

Language

Verilog, VHDL

Message Details

`Non-synthesizable statement`

Severity

SynthesisError

SYNTH_5325

Clock events on more than one clock signals.

Description

The *SYNTH_5352* rule reports a violation if a clock event is specified on multiple clock signals.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

```
[SynthesisError] Clock events on more than one clock signals
```

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Update the logic so that a clock event is used for only one clock signal.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity test is
port ( clk : in bit ;
set : in bit ;
input : in integer ;
output : out integer
) ;
end test ;
architecture rtl of test is
```

```
begin
process
begin
wait until ((clk'event and clk= '1') or (set'event and set
='1'));
output <= input;
end process ;
end rtl ;
```

In the above example, the *SYNTH_5325* rule reports a violation because the clock event is used with both the `clk` and `set` signal.

Default Severity Label

SynthesisError

SYNTH_5326

Clock direction is not specified.

Description

The *SYNTH_5326* rule reports a violation when the clock direction is not specified.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

```
[SynthesisError] Clock direction is not specified
```

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Update the logic so that clock direction can be determined.

Example Code and/or Schematic

Consider the following example of a VHDL entity:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
port (
d,clk1: in STD_LOGIC;
set  : boolean ;
out1: out STD_LOGIC
);
end test ;
architecture rtl of test is
begin
process (clk1,set)
```

```
begin
if (set) and (not clk1'EVENT) then
out1 <= '1' ;
end if ;
end process;
end rtl ;
```

In the above example, the *SYNTH_5326* rule reports a violation because of the missing expression `clk='1'` or `clk='0'` with the `clk1'` event. Therefore, the clock direction is unknown in this case.

Default Severity Label

SynthesisError

SYNTH_5328

Expressions cannot be evaluated statically, hence exiting synthesis

Language

Verilog

Rule Description

Consider the following example:

```
module test ( in1,in2, out1);
parameter pBuswidth  = 1;
input[15:0]            in1;
input[1:0]             in2;
output[pBuswidth-1:0]  out1;
reg[pBuswidth-1:0]     regA,regB;
reg[pBuswidth-1:0]     out1;
parameter pZero      = 32'b0;

always @(in2 or in1 or regA or regB)
  begin
    casez(8'b00x1x0xz)
      8'b00xxxxxz : out1 = regA + regB;
      default: out1 = 'bx;
    endcase
  end
endmodule
```

In the above example, the value of case expression, `8'b00x1x0xz`, cannot be evaluated. Hence, SpyGlass flags the following synthesis error:

```
Expressions with unknown (8'b00x1x0xz) cannot be evaluated
statically, hence exiting synthesis
```

Message Details

```
Expressions with unknown (<value>) cannot be evaluated
statically, hence exiting synthesis
```

Severity

SynthesisError

SYNTH_5329

Function returning objects of different sizes.

Description

The *SYNTH_5329* rule reports a violation if a function returns objects of incorrect sizes.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Function returning objects of different sizes

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Update the logic so that the function returns objects of correct size.

Example Code and/or Schematic

Consider the following VHDL example:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
generic (
SIZE : Integer := 4
);
port (
a : in std_logic_vector(SIZE downto 0);
a1 : in std_logic_vector(SIZE-1 downto 0);
b : out std_logic_vector(SIZE downto 0)
);
end  test;
```

```
architecture rtl of test is
signal s1 : boolean;
function IT_COND_OP(COND: BOOLEAN; ARG1, ARG2:
STD_LOGIC_VECTOR) return STD_LOGIC_VECTOR is
begin
if (COND) then
return ARG1;
else
return ARG2;
end if;
end IT_COND_OP;
begin
b((SIZE - 1) downto 0) <= it_cond_op(s1,a,a1);
end rtl;
```

In the above example, the *SYNTH_5329* rule reports a violation because for the return arguments `ARG1` and `ARG2` in the `it_cond_op` function, the `a` and `a1` signals are passed, which are of different sizes, `SIZE` and `SIZE-1`, respectively.

Default Severity Label

SynthesisError

SYNTH_5330

Number of encodings too less than the number of values.

Language

Verilog, VHDL

Message Details

```
Number of encodings (<num>) too less than the number of values
(<num-values>)
```

Severity

SynthesisError

SYNTH_5331

Number of encodings too much than the number of values.

Language

Verilog, VHDL

Message Details

Number of encodings (<num>) too much than the number of values (<num-val>)

Severity

SynthesisError

SYNTH_5332

All encodings must be of equal length in enumeration.

Language

Verilog, VHDL

Message Details

`All encodings must be of equal length in enumeration`

Severity

SynthesisError

SYNTH_5333

Item in case should be constant.

Language

Verilog, VHDL

Message Details

`Non-constant item in case`

Severity

SynthesisError

SYNTH_5334

Only range type 'downto' and 'to' expected.

Language

Verilog, VHDL

Message Details

```
Only range type 'downto' and 'to' expected
```

Severity

SynthesisError

SYNTH_5335

Error in Type-Casting

Language

Verilog, VHDL

Message Details

`Error in Type-Casting`

Severity

SynthesisError

SYNTH_5336

Signal name generated by using the generate block -name scoping matches with another signal with the same name

Description

The *SYNTH_5336* rule reports a violation if the name of the signal generated by using a `generate` block-name scoping matches with the name of another signal.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Non similar duplicate signal name found <signal-name>

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, either change the name of the `generate` block or change the name of the reported signal.

Example Code and/or Schematic

Consider the following example:

```
module test (in1,out1);
input in1;
output out1;
genvar i;
generate begin:B1
```

```
for (i = 0; i <= 2; i = i + 1)
begin : B2
wire [1:0] net1;
end
assign \B2[0].net1 =in1;
end
endgenerate
endmodule
```

For the above example, the *SYNTH_5336* rule reports a violation because `\B2[0].net1` is the duplicate signal name that matches with the name `B2[0].net1` generated by using the generate block-name scoping.

Default Severity Label

SynthesisError

SYNTH_5337

Non-static base should not have a negative exponent.

Description

The *SYNTH_5337* rule reports a violation for a non-static base with a negative exponent.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5337* rule reports the following message:

```
[ERROR] Non-static base should not have a negative exponent
```

Consequences of Not Fixing

If you do not fix this violation, the non-static base having a negative exponent is ignored during synthesis.

How to Debug and Fix

To fix this violation, provide a positive exponent for the non-static base.

Example Code and/or Schematic

Consider the following example:

```
module test (in1, out1, out2);
  input [7:0]in1;
  output [7:0]out1, out2;
  assign out1[7:0] = in1 ** (-1.2); //SYNTH_5337 violation
  assign  out1[7:0]  =  in1 **  ( 12 ); // non-static base
                  // with positive exponent is fine
 assign out2[7:0] = 3 ** (-3);   // static base with
                     //negative exponent is fine
endmodule
```

For the above example, the *SYNTH_5337* rule reports a violation for `in1 ** (-1.2)` but not for `3 ** (-3)` where the base is static.

Default Severity Label

SynthesisError

SYNTH_5338

Exponentiation is supported only when base can be statically evaluated and power of 2.

Description

The *SYNTH_5338* rule reports a violation if a base cannot be evaluated statically and is not a power of two.

Rule Exception

For Verilog, SpyGlass supports non-static base only when the power of exponentiation is either 0, 1, or 2.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5338* rule reports the following message:

[ERROR] `Exponentiation is supported only if the base is a power of 2 or the exponent is 0, 1, or 2`

Consequences of Not Fixing

If you do not fix this violation, SpyGlass ignores exponentiation when the base cannot be evaluated statically and is not a power of two.

How to Debug and Fix

To fix this violation, specify the base as the power of two or specify the exponent as 0, 1, or 2.

Example Code and/or Schematic

The *SYNTH_5338* rule reports a violation in the following case:

```
module test1 (in1,in2,out1);
  input integer in1,in2;
```

```
  output [63:0] out1;
  assign out1 = in1 ** in2;
endmodule
```

In the above example, `in1` and `in2` are non-static. Therefore, exponentiation is not supported in this case and they are not synthesized.

To synthesize them, declare `in2` as a parameter with the value as 0,1 or 2.

Default Severity Label

SynthesisError

SYNTH_5339

Each entity should have at least one architecture.

Language

VHDL

Message Details

```
No architecture found for entity <entity>
```

Severity

SynthesisError

SYNTH_5340

Pragma map_to_entity is used on procedure with inout ports

Language

Verilog, VHDL

Message Details

```
Pragma map_to_entity is used on procedure <procedure> with
inout ports
```

Severity

SynthesisError

SYNTH_5341

This clock-event has assignment in its else portion, not synthesizable

Language

VHDL

Message Details

```
This clock-event has assignment in its else portion, not
synthesizable
```

Severity

SynthesisError

SYNTH_5342

Enums declared inside named for-generate statement, with the generate variable as one of Enum items, is not synthesizable

Language

Verilog, VHDL

Message Details

```
Enums declared inside named for-generate statement, with the
generate variable as one of Enum items, is not synthesizable
```

Severity

SynthesisError

SYNTH_5343

Simulation not possible. So quitting synthesis

Language

Verilog, VHDL

Message Details

```
Simulation not possible. So quitting synthesis
```

Severity

SynthesisError

SYNTH_5344

Base of an exponentiation should be a power of two for a non-static exponent.

Description

The *SYNTH_5344* rule reports a violation if the base of exponentiation for a non-static exponent is not the power of two.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5344* rule reports the following message:

[ERROR] `Base of an exponentiation should be a power of 2 for a non-static exponent`

Consequences of Not Fixing

If you do not fix this violation, the non-static exponent is ignored during synthesis.

How to Debug and Fix

To fix this violation, specify the base of exponentiation as the power of two for the non-static exponent.

Example Code and/or Schematic

Consider the following example:

```
module test( in1, out1, out2);
  input [7:0]in1;
  output [7:0]out1, out2;
  assign out1[7:0] = 11 ** (in1); // SYNTH_5344 violation
  assign out1[7:0]  =  16 ** (in1); //Base is power of '2'
                       // which is fine
 assign out2[7:0] = 11 ** (3);  // exponent is static                    /
/ which is fine
```

```
endmodule
```

In the above example, the *SYNTH_5344* rule reports a violation for `11 ** in1` that has a non-static exponent. To fix this violation, replace 11 by any number (say 16) that is a power of 2.

Default Severity Label

SynthesisError

SYNTH_5345

This type of clock construct within for loop is not synthesizable.

Language

Verilog, VHDL

Message Details

```
This type of clock construct within for loop is not
synthesizable
```

Severity

SynthesisError

SYNTH_5346

No driver in this region for 'driving_value attribute of signal.

Language

Verilog, VHDL

Message Details

```
No driver in this region for 'driving_value attribute of signal
<signal>
```

Severity

SynthesisError

SYNTH_5347

Always/Process block with multiple embedded edge event expressions is not supported by synthesis.

Description

The *SYNTH_5347* rule reports a violation if an always or process block contains multiple embedded edge event expressions.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Always/Process block that has multiple embedded edge event expressions is not supported by synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, move the edge event expressions outside the `for` loop appearing in the `always` block.

Example Code and/or Schematic

Consider the following example:

```
module test (in1,out1,clk);
  input [3:0] in1;
  output reg [3:0] out1;
  input clk;
  integer i;
  always
```

```
  begin
  for(i=0;i<4;i=i+1)
  begin
  @(posedge clk) out1<=in1;
  end
  end
endmodule
```

For the above example, the *SYNTH_5347* rule reports a violation because the event expression `(posedge clk1)` is called four times because of the `for` loop.

To make this `always` block synthesizable, move the edge event expression outside the `for` loop.

Default Severity Label

SynthesisError

SYNTH_5349

Generic parameter should be initialized.

Language

Verilog, VHDL

Message Details

`Uninitialized generic parameter <parameter>`

Severity

SynthesisError

SYNTH_5351

Configuration not in library.

Language

VHDL

Message Details

`Configuration <configuration> not found in library <library>`

Severity

SynthesisError

SYNTH_5352

Error in creating design hierarchy.

Language

Verilog, VHDL

Message Details

```
Error in creating design hierarchy
```

Severity

SynthesisError

SYNTH_5353

Entity not in library.

Language

VHDL

Message Details

```
Entity <entity> not found in library <library>
```

Severity

SynthesisError

SYNTH_5354

Design unit not found.

Language

Verilog, VHDL

Message Details

`No design unit <design-unit> found`

Severity

SynthesisError

SYNTH_5355

Design unit not defined.

Language

Verilog, VHDL

Message Details

`Design unit <design-unit> not defined.`

Severity

SynthesisError

SYNTH_5356

Could not obtain elaborated entity.

Language

Verilog, VHDL

Message Details

`Could not obtain elaborated entity`

Severity

SynthesisError

SYNTH_5358

No up-to-date architecture found for entity, please clean-up your work library.

Language

VHDL

Message Details

```
No up-to-date architecture found for entity <entity>, please
clean-up your work library
```

Severity

SynthesisError

SYNTH_5359

Architecture for entity not found in library.

Language

VHDL

Message Details

```
Architecture <architecture> for entity <entity> not found in
library <library>
```

Severity

SynthesisError

SYNTH_5360

SHORT REAL declarations inside SV constructs are not supported for synthesis

Description

SpyGlass reports the *SYNTH_5360* violation if any `short-real` declaration is present inside the SV constructs, such as structs, unions, typedefs, and enums, while creating the type-tree for SV constructs.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] `SHORT REAL Declarations inside SV Constructs are not supported for synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Remove the `short-real` declaration from SV constructs.

Example Code and/or Schematic

The following example shows the `short-real` declaration inside the SV constructs:

```
module test;

typedef struct packed { // packed structure
  int a;
  byte b;
} packet_t;

typedef struct { // unpacked structure
```

```
  int a;
  shortreal b;
} data_t;

packet_t [23:0] packet_array;
// packed array of 24 structures

data_t data_array [23:0];
// unpacked array of 24 structures

initial begin
  $display("\n packet_array[0].a = %0d (expect 0)",
  packet_array[0].a);
  $display(" data_array[0].b = %1.1f (expect 0.0)\n",
  data_array[0].b);
  $finish;
  end

endmodule
```

Default Severity Label

SynthesisError

SYNTH_5361

Use of unsupported alias

Description

The *SYNTH_5361* rule reports a violation if an alias is used inside a `package` or a `process` block.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] This type of alias usage is not supported

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Remove the alias usage from the `package` or a `process` block.

Example Code and/or Schematic

Example 1

The following example shows the usage of an alias (in red) inside a `package` block:

VHDL entity

```
#########################################################
library ieee;
use ieee.std_logic_1164.all;
package pack0 is
signal sig_tsb : std_logic;
alias T_T_TSB2 : std_logic is sig_tsb;
end;
```

```
library ieee;
use ieee.std_logic_1164.all;
use work.pack0.all;
entity alias_test is
end alias_test;
architecture rtl of alias_test is
signal Q_LSB: std_logic ;
begin
Q_LSB <= T_T_TSB2;
end rtl;
```

Example 2

The following example shows the usage of an alias (in red) inside the `process` block:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
port (A : in std_logic);
end test;
architecture rtl of test is
begin
process(A)
variable tempA : std_logic;
alias aliasA: std_logic is tempA;
procedure proc1 is
begin
aliasA := '0';
end proc1;
begin
proc1;
end process;
end rtl;
```

Default Severity Label

SynthesisError

SYNTH_5362

Illegal bit select

Language

Verilog, VHDL

Message Details

`Illegal bit select on <name>`

Severity

SynthesisError

SYNTH_5363

Complex SV parameters inside task/function are not supported

Language

Verilog

Rule Description

SpyGlass reports this synthesis error if complex SV parameters are specified inside a task/function.

For example, this rule reports a violation in the following example in which a complex SV parameter is specified inside a task:

```
typedef struct packed
  {
  reg [3:0] reg1;
  } struc1;
  task ABC(.....)
    parameter struc1 const1 = '{reg1:4'b1011};
    ......
    .....
endtask
```

Message Details

`Complex SV parameters inside task/function is not supported`

Severity

SynthesisError

SYNTH_5364

Always/Process block that has embedded edge event expression associated with forever construct is not supported by synthesis

Description

The *SYNTH_5364* rule reports a violation if an `always`/`process` block, which contains an embedded edge-event expression, is associated with a `forever` construct.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] `Always/Process block that has embedded edge event expression associated with forever construct is not supported by synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Modify the code so that the `forever` construct is not used with the `always`/`process` block containing an embedded edge-event expression.

Example Code and/or Schematic

Consider the following example:

```
module test(sel, d, q3);
  input [3:0] d;
  input [2:0] sel;
  output q3;
```

```
  reg q3;
  wire clk;
integer i,j,k;
  always@(posedge clk)
     begin : mux03
 case (sel[1:0])
   2'b00 :
        forever@ (posedge clk) ;
   2'b01 :
        repeat(j);
   2'b10 :
        for (i=0 ; j<10; k=k+1) ;
   2'b11 :
        while(j<10) ;
 endcase
     end
endmodule
```

In the above example, `forever@ (posedge clk)` is not a supported construct. Therefore, this rule reports a violation.

Default Severity Label

SynthesisError

SYNTH_5365

Non-synthesizable statement.

Description

The *SYNTH_5365* rule reports non synthesizable statements.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] <statement> type not synthesizable.

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Double-click on the violation to check where is reported statement is being used.

Remove the reported unsynthesizable statement from the code.

Example Code and/or Schematic

Consider the following example:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity sfifo is
port (
  d : in std_logic;
  q : out integer
);
end entity;
```

```
architecture rtl of sfifo is
  type arrtype is array (std_logic) of integer;
    // arrtype not synthesizable


  constant arr : arrtype := (1, 11, 111, 2, 22, 222, 3, 33,
  333); begin
  process begin
    q <= arr(d);
  end process;
end architecture;
```

In the above example, an unsupported range type exists in the array declaration, thereby causing the *SYNTH_5365* violation.

Default Severity Label

SynthesisError

SYNTH_5366

Uninitialized deferred constants are not supported for synthesis

Description

The *SYNTH_5366* rule reports a violation if uninitialized deferred constants are used in the code.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

```
[SynthesisError] Uninitialized Deferred Constants are not supported for synthesis
```

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Initialize the deferred constant with a constant value.

Example Code and/or Schematic

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;
package SRRC_coeffs is
   subtype ROMx3_WORD is STD_LOGIC;
   type ROMx3_TABLE is array (0 to 1000) of ROMx3_WORD;
   constant ROM_SRRCx3 : ROMx3_TABLE ;
end SRRC_coeffs ;
use work.SRRC_coeffs.all;
```

```
entity ROMx3 is
port(
          SRRC_out : out ROMx3_WORD
    );
end;
architecture ROMx3_arch of ROMx3 is
    begin
        SRRC_out <= ROM_SRRCx3(1);
end ROMx3_arch;
```

For the above example, the *SYNTH_5366* rule reports a violation because the `ROM_SRRCx3` constant is uninitialized.

Default Severity Label

SynthesisError

SYNTH_5368

Clock signal read in asynchronous portion of clocked if-else construct - not synthesizable.

Language

Verilog, VHDL

Message Details

```
Clock signal '<signal>' read in asynchronous portion of clocked
if-else construct - not synthesizable
```

Severity

SynthesisError

SYNTH_5369

Sub program recursion more than the recursion limit.

Description

The *SYNTH_5369* rule reports a violation if a sub-program recursion occurs multiple times.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] SubProgram '<sub-program>' recursion more than the recursion limit (<limit>)

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Double click on the violation to view the code where recursion is occurring, and remove the recursion.

Example Code and/or Schematic

Example 1

Consider the following example of a module:

```
module test(input integer in1, output integer out1);
  function automatic integer my(input integer in1, i);
    integer i1;
      if (i>0)
        i1 = my(in1, i-1) + i + in1;
      else
        i1 = 0;
```

```
    return i1;
  endfunction
  assign out1 = my(in1, 230);
endmodule
```

In the above example, loop recursion occurs more than the maximum allowed recursion limit. Therefore, the *SYNTH_5369* rule reports a violation.

Example 2

Consider the following example:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity test is
    port (in1: in integer; out1: out integer);
end test;
architecture arch_test of test is
    function my(in1, i: in integer)
                return integer is
        variable s1: integer;
    begin
        if (i > 0) then
            s1 := my(in1, i-1) + i + in1;
        else
            s1 := 0;
        end if;
        return s1;
    end function;
begin
    out1 <= my(in1, 230);
end arch_test;
```

In the above example, loop recursion occurs more than the maximum allowed recursion limit. Therefore, the *SYNTH_5369* rule reports a violation.

Default Severity Label

SynthesisError

SYNTH_5370

Return Type of subprogram is unconstrained

Language

Verilog, VHDL

Message Details

`Return Type of subprogram <sub-program> is unconstrained`

Severity

SynthesisError

SYNTH_5371

Use of signals / shared variables declared inside packages is not synthesizable.

Language

Verilog, VHDL

Message Details

```
Use of signals / shared variables '<name>' declared inside
packages is not synthesizable
```

Severity

SynthesisError

SYNTH_5372

No. of <variables / terminals> exceeds the maximum limit of that can be handled.

Description

The *SYNTH_5372* rule reports a violation when <variables / terminals> exceeds the maximum limit that can be handled.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] No. of <variables / terminals> exceeds the maximum limit of that can be handled.

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Reduce the size of the design.

Example Code and/or Schematic

Consider the following example:

```
module test (in1,clk,en,addr,out1);
parameter pa = 1120;
parameter pb = 116;
input [pb-1:0] in1[pa-1:0];
input clk,en;
input [13:0] addr;
output reg [pb-1:0] out1 ;
genvar  i,j;
generate begin
reg [pb-1:0] reg1 [pa-1:0];
for (i=0;i<pa;i=i+1)
```

```
for (j=0;j<pb;j=j+1)
always @(posedge clk)
  reg1[i][j] = in1[i][j];
end
endgenerate
endmodule
```

Default Severity Label

SynthesisError

SYNTH_5373

Unable to write a conversion function for inout port

Language

Verilog, VHDL

Message Details

`Unable to write a conversion function for inout port <port>`

Severity

SynthesisError

SYNTH_5375

An instance is having problem in binding with its master.

Description

The *SYNTH_5375* rule reports a violation if there is a binding error in an instance.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] Binding error in user instance '<inst-name>' leads to black-boxing of parent module

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Double-click on the violation to view the line in the code where the binding error exists.

Analyze the cause of the error and rectify the code accordingly.

Example Code and/or Schematic

Consider the following example:

```
-------------------------------------------------------------
```

Verilog Top Module:

```
module top(A, Z);
input A;
output Z;
```

```
myNOT I(.P(A),.Z(Z));
endmodule
-------------------------------------------------------------


-------------------------------------------------------------
```

VHDL Entity:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity myNOT is
port ( A :in bit ;Z:out bit );
end myNOT;
architecture rtl of myNOT is
begin
Z <= not (A);
end rtl;
-------------------------------------------------------------
```

In the above example, the port interface mentioned during instantiation has `P` and `Z`. However, the entity definition has ports as `A` and `Z`.

Due to this mismatch, the `I` instance of `myNOT` has an elaboration error, the binding could not take place.

The module having this instance is made a black box for further rule checking.

Default Severity Label

SynthesisError

SYNTH_5377

`event is only supported for single bit signals.

Language

VHDL

Message Details

`event is only supported for single bit signals.

Severity

SynthesisError

SYNTH_5378

Complex expression is not allowed in event specification for synthesis.

Description

The *SYNTH_5378* rule reports a violation if a complex expression is present in an event specification.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] Complex expression '<expression>' is not allowed in event specification for synthesis

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Remove the reported complex expression from the event specification.

Example Code and/or Schematic

Consider the following example:

```
module test (in1,clk,out1);
  input in1,clk;
  output logic out1;
  always @(posedge (!clk) )
    out1<= in1;
endmodule
```

For the above example, the *SYNTH_5378* reports a violation for the

complex expression `(!clk)` used in the sensitivity list.

Default Severity Label

SynthesisError

SYNTH_5379

WAIT statement inside a procedure is not yet supported.

Language

VHDL

Message Details

`WAIT statement inside a procedure is not yet supported.`

Severity

SynthesisError

SYNTH_5380

NULL Range not allowed.

Description

The *SYNTH_5380* rule reports a violation if a null range is specified for an expression.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Null Range on RHS and not on LHS...So Range Mismatch`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Correct the logic so fix the range mismatch.

Example Code and/or Schematic

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;
entity test is
generic (
SIZE : Integer := 4
);
port (
a : in std_logic_vector(SIZE*2 downto 0);
b : out std_logic_vector(SIZE downto 0)
);
end  test;
```

```
architecture rtl of test is
begin
b((SIZE - 1) downto 0) <= a(SIZE downto SIZE*2-1);
end rtl;
```

In the above example, the *SYNTH_5380* rule reports a violation because the range `4 downto 7` is considered as the null range. In this case, the range should be `7 downto 4`.

Default Severity Label

SynthesisError

SYNTH_5381

NULL Range not allowed.

Language

Verilog, VHDL

Message Details

`Null Range on LHS and not on RHS...So Range Mismatch`

Severity

SynthesisError

SYNTH_5383

Error condition occurred

Language

Verilog, VHDL

Message Details

```
Error condition occurred
```

Severity

SynthesisError

SYNTH_5384

Could not calculate value putting MSB instead.

Language

Verilog, VHDL

Message Details

`Could not calculate value putting MSB instead`

Severity

SynthesisError

SYNTH_5385

Range should be static

Language

Verilog, VHDL

Rule Description

This error occurred while calculating the range.

Message Details

`Should be static`

Severity

SynthesisError

SYNTH_5386

x/z/? cannot occur in repetition multiplier

Description

The *SYNTH_5386* rule reports a violation if a repetition multiplier in a concatenation expression contains X/Z/?.

Repetition multiplier for a concatenation should always be a constant value.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Repetition multiplier in a concatenation expression <expression> is not constant expression

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Specify a constant value to the reported repetition multiplier for concatenation.

Example Code and/or Schematic

Consider the following example:

```
module test(output out1);
assign out = {2'b1x{1'b1}};
endmodule

module test1(output out1);
assign out = {2'b1z{1'b1}};
```

```
endmodule

module test2(output out1);
assign out = {2'b1?{1'b1}};
endmodule
```

For the above example, the *SYNTH_5386* rule reports a violation due to the usage of the `2'b1x`, `2'b1z`, and `2'b1?` expressions in the repetition multiplier.

Default Severity Label

SynthesisError

SYNTH_5387

Index take values outside array bound

Description

The *SYNTH_5387* rule reports a violation when two index ranges are disjoint.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Index <index> take values outside array bound <array-bound>

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Modify the logic to specify the correct index range.

Example Code and/or Schematic

The following example shows the scenario in which this rule reports a violation:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ent1 is
port (
in1 : in std_logic_vector(3 downto 0);
out1 : out std_logic_vector(3 downto 0)
);
end ent1;


architecture rtl of ent1 is
```

```
type t1 is array (integer range <>) of std_logic;
signal s1 : t1(1 to 4);
begin
s1(0) <= in1(0);
end rtl;
```

In the above example, value ‘0’ is outside the array bound s1[1:4] in assignment s1[0].

Default Severity Label

SynthesisError

SYNTH_5392

Range for STD_ULOGIC type not calculable.

Language

VHDL

Message Details

`Range for STD_ULOGIC type not calculable.`

Severity

SynthesisError

SYNTH_5393

Unable to retrieve range for STD_ULOGIC type.

Language

VHDL

Message Details

Unable to retrieve range for STD_ULOGIC type.

Severity

SynthesisError

SYNTH_5394

Encountered an undefined std_ulogic type.

Language

VHDL

Message Details

`Encountered an undefined std_ulogic type.`

Severity

SynthesisError

SYNTH_5395

Improper asynchronous style of modeling. Not synthesizable.

Description

The *SYNTH_5395* rule reports a violation when an incorrect asynchronous modeling style is used.

Correct Usage of Asynchronous Modeling Style

For synthesis of asynchronous modeling style, the design should meet the following conditions:

- The `if-else if` statement chain conditional expression should check for all the asynchronous reset/set signals first.
- In the implicit style sequential state machine, the event control statement should have more than one edge specification.
- In asynchronous reset/set always block, comparison should not be made to a non-constant expression in reset/set condition.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Improper asynchronous style of modeling. Not synthesizable`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

See *Correct Usage of Asynchronous Modeling Style*.

Example Code and/or Schematic

Consider the following example:

```
module test(set,reset,clk, in,out, en);
input set,reset,clk, en;
output out;
reg out;
input [3:0] in;
always
begin
   begin
   @(posedge clk or posedge reset);
   end
   if (reset)
   out = 0;
   else
   begin
   out = in[0];
   @(posedge clk or posedge reset)
      begin if (reset)
      out = 0;
      else
      out = in[1];
      end
   end
end
endmodule
```

For the above example, the *SYNTH_5395* rule reports a violation because the edge control events are nested.

Default Severity Label

SynthesisError

SYNTH_5396

Multidimensional array of instances is not supported with "set_option handlememory 1"

Description

The *SYNTH_5396* rule reports a violation when the `handlememory` command is used with multidimensional array of instances.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

```
[SynthesisError] Multidimensional array of instances is not supported with 'set_option handlememory 1'
```

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Either remove the `handlememory` command from the project file or remove the multidimensional array of instances from the RTL.

Example Code and/or Schematic

Consider the following example:

```
typedef struct {
struct {
logic signed [3:0] a;
reg [7:0] b;
}ab;
int c[2];
```

```
}abc[2:0][1:0];
module botInst(input abc a, input in, output out);
  assign out = in ;
endmodule
module top(input in, output [4:0][3:0]out);
  abc abc1[4:0][3:0];
  botInst bI[4:0][3:0] (abc1, in, out);
endmodule
```

For the above example, if you specify the `set_option handlememory 1` project-file command, the *SYNTH_5396* rule reports a violation.

Default Severity Label

SynthesisError

SYNTH_5397

Unlabeled <name> is not supported in implicit state machine.

Language

Verilog, VHDL

Message Details

`Unlabeled <name> is not supported in implicit state machine.`

Severity

SynthesisError

SYNTH_5398

Signal is not declared in the current scope.

Language

Verilog, VHDL

Message Details

`Signal '<signal-name>' is not declared in the current scope.`

Severity

SynthesisError

SYNTH_5399

Complex alias types not handled yet.

Language

Verilog, VHDL

Message Details

`Complex alias types not handled yet`

Severity

SynthesisError

SYNTH_5400

Non-synthesizable clock construct

Language

Verilog, VHDL

Message Details

`Non-synthesizable clock construct`

Severity

SynthesisError

SYNTH_5401

Non-static loop or event waits in only some branches.

Language

Verilog, VHDL

Message Details

`Non-static loop or event waits in only some branches`

Severity

SynthesisError

SYNTH_5402

Non Synthesizable Design.

Language

Verilog, VHDL

Message Details

`Non Synthesizable Design`

Severity

SynthesisError

SYNTH_5403

Unsupported Output format

Language

Verilog, VHDL

Message Details

`Unsupported Output format <format>`

Severity

SynthesisError

SYNTH_5405

Clock expression must be one-bit wide.

Description

The *SYNTH_5405* rule reports a violation for clock expressions that are more than one-bit wide.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_5405* rule reports the following message:

```
[ERROR] Clock expression '<expression>' must be one bit wide
```

Consequences of Not Fixing

If you do not fix this violation, the reported clock expression is ignored during synthesis.

How to Debug and Fix

To fix this violation, update the reported clock expression to make them one-bit wide.

Example Code and/or Schematic

Consider the following example:

```
module test(in1, clk, out1);
  input [1:0]in1, clk;
  output reg [1:0]out1;
  always@(posedge clk)
  begin
    out1 = in1;
  end
endmodule
```

In the above example, the *SYNTH_5405* rule reports a violation because

`clk` is two bits wide.

Default Severity Label

SynthesisError

SYNTH_5406

Float Step is not supported for synthesis.

Language

Verilog, VHDL

Message Details

`Float Step '<step>' is not supported for synthesis`

Severity

SynthesisError

SYNTH_5409

Net range do not match with port range

Language

Verilog, VHDL

Message Details

`Net range do not match with port range`

Severity

SynthesisError

SYNTH_5410

Could not open target library, Mapping to generic gates.

Language

Verilog, VHDL

Message Details

```
Could not open target library <library>, Mapping to generic
gates
```

Severity

SynthesisError

SYNTH_5411

Zero or negative repetition multiplier found in concatenation expression

Description

The *SYNTH_5411* rule reports a violation if a concatenation expression contains zero or a negative repetition multiplier.

As per LRM Verilog 2005, a concatenation expression with a zero replication constant is valid only if it contains at least one positive-size item. See *Example 1* and *Example 2* for violating and non-violating scenarios, respectively, for this rule.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

```
[SynthesisError] Zero or negative repetition multiplier found in concatenation expression <expression>
```

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, use positive numbers in repetition multipliers. In case of zero replication constant, use at least one positive-size item.

Example Code and/or Schematic

Example 1

Consider the following example:

```
module test2(output out2);
  assign out2 = {0{1'b1}};
```

```
endmodule

module test3(output out3);
  assign out3 = {-1{2'b11}};
endmodule
```

For the above example, the *SYNTH_5411* rule reports violations for the `test2` and `test3` modules because of the usage of the `0` and `-1` repetition multipliers in the concatenation expressions.

Example 2

Consider the following example:

```
module test1(output out1);
  assign out1 = {{1'b0{2'b11}} , 1'b1};
endmodule
```

For the above example, the *SYNTH_5411* rule does not report any violation. This is because the positive size value, `1'b1`, is used inside the concatenation expression `{{1'b0{2'b11}} , 1'b1}`. And as per LRM Verilog 2005, such expressions are valid.

Default Severity Label

SynthesisError

SYNTH_5416

Nothing to synthesize.

Language

Verilog, VHDL

Message Details

`Nothing to synthesize`

Severity

SynthesisError

SYNTH_5418

Could not analyze library.

Language

Verilog, VHDL

Message Details

`Could not analyze library <library>.`

Severity

SynthesisError

SYNTH_5419

TCL ERROR: <message1> <message2>

Language

Verilog, VHDL

Rule Description

This violation appears when an invalid Tcl command is used in the a Tcl command file.

Message Details

`TCL ERROR: <message1> <message2`

Severity

SynthesisError

SYNTH_5420

Nested block is not supported in the Tcl file

Language

Verilog, VHDL

Message Details

`Nested block is not supported in the Tcl file`

Severity

SynthesisError

SYNTH_5421

SystemVerilog and Synthesis Optimization cannot be concurrently invoked

Language

Verilog

Message Details

`SystemVerilog and Synthesis Optimization cannot be concurrently invoked`

Severity

SynthesisError

SYNTH_5422

Could not analyze source files

Language

Verilog, VHDL

Message Details

`Could not analyze source files`

Severity

SynthesisError

SYNTH_5423

Synthesis failed for design unit.

Language

Verilog, VHDL

Message Details

```
Synthesis failed for design unit
```

Severity

SynthesisError

SYNTH_5425

Could not write to file <file-name>.

Language

Verilog, VHDL

Message Details

`Could not write to file <file-name>`

Severity

SynthesisError

SYNTH_5427

Language detection failed.

Language

Verilog, VHDL

Message Details

`Language detection failed.`

Severity

SynthesisError

SYNTH_5430

<name> must be followed by a level.

Language

Verilog, VHDL

Message Details

<name> must be followed by a level

Severity

SynthesisError

SYNTH_5435

Multiply driven nets are expected to have resolution function.

Language

Verilog, VHDL

Rule Description

NOTE: *This error is suppressed for memory net when the* `set_option handlememory 1` *project file command is specified.*

Message Details

```
Multiple drivers on net <net> with no resolution function
defined
```

Severity

SynthesisError

SYNTH_5436

User-defined function call on formal associations is not supported for synthesis.

Language

Verilog, VHDL

Message Details

```
User-defined function call on formal associations is not
supported for synthesis
```

Severity

SynthesisError

SYNTH_5437

User-defined function call on actual associations is not supported for synthesis

Language

Verilog, VHDL

Rule Description

This message appears to indicate that a user-defined function call on actual associations is not supported for synthesis.

Consider the following example:

```
entity tctrl is
  port (
     opb_dbus_a  : in  std_logic_vector(0 to 31);
     sl_dbus_a   : inout std_logic_vector(0 to 31)
     );
 end tctrl;
 architecture rtl of tctrl is
   component tctrl_opb_if
     port (
       opb_dbus    : in  std_ulogic_vector(0 to 31);
       sl_dbus     : inout std_ulogic_vector(0 to 31)
       );
   end component tctrl_opb_if;
 begin
   tctrl_opb_if_i0 : tctrl_opb_if
     port map (
       opb_dbus  => to_stdulogicvector(opb_dbus_a),
         -- in std_logic_vector(0 to 31);

       sl_dbus   => to_stdulogicvector(sl_dbus_a
         -- out std_logic_vector(0 to 31);

       );
 end rtl;
```

In the above example, you cannot have a function call on `sl_dbus_a` on which you are writing.

Message Details

```
User-defined function call on actual associations is not
supported for synthesis
```

Severity

SynthesisError

SYNTH_12601

Range of values of T'VAL, T'SUCC or T'PRED index exceeds size of enum type T

Language

VHDL

Rule Description

This rule reports a violation when the range of values of `T'VAL`, `T'SUCC`, or `T'PRED` index exceeds the size of enum type T. In such cases, there is a possible simulation synthesis mismatch.

For example, this rule reports a violation in the following case:

```
package p is
    type color is (violet, blue, green, yellow, red, white);
    end p;
use work.p.all;
entity top is
       port (x : in integer;
        z :out color);
end top;
architecture behav of top is
  begin
   test:process
   begin
    z <= color'val(x);
     -- SYNTH_12601 violation because x can have a value
     -- that is greater than the size of the enum color
   end process test;
end behav;
```

Message Details

```
Range of values of T'<VAL|SUCC|PRED> index exceeds size of enum
type T. Possible simulation synthesis mismatch
```

Severity

SynthesisWarning

SYNTH_12602

Binding indication containing configuration as entity aspect is ignored for configuration specification during synthesis. Default binding is used for such cases during synthesis.

Language

VHDL

Message Details

```
Binding indication containing configuration as entity aspect is
ignored for configuration specification during synthesis.
Default binding is used for such cases during synthesis
```

Severity

SynthesisWarning

SYNTH_12603

Incorrect Translation of SystemVerilog. This might lead to incorrect synthesis.

Language

Verilog, VHDL

Message Details

```
Incorrect Translation of SystemVerilog. This might lead to
incorrect synthesis
```

Severity

SynthesisWarning

SYNTH_12604

Unique Type case statement is being used which may have overlapping conditions

Language

VHDL

Rule Description

Consider the following example:

```
logic [1:0] b;
unique casex(b)
  2'b00: out1=in1+in2;
  2'b0x: out1=in1*in2;
  2'b10: out1=in1*in2;
  default : out1 = in1/in2;
endcase
```

In the above example, unique case conditions, `2'b00` and `2'b0x`, are overlapping. Hence, SpyGlass flags the SYNTH_12604 warning in this case.

Message Details

```
Used Unique Type <type> statement but it may have overlapping
conditions
```

Severity

SynthesisWarning

SYNTH_12605

Priority/Unique Type if/case statement is being used but all the conditions are not covered

Language

Verilog, VHDL

Rule Description

Consider the following example:

```
logic [1:0] b;
priority case(b)
  2'b00: out1=in1+in2;
  2'b01: out1=in1*in2;
  2'b11: out1=in1-in2;
endcase
```

In the above example, the case condition, `2'b10`, is missing. Hence, SpyGlass flags the SYNTH_12605 warning in this case.

Similar to the `priority-case` statement discussed in the above example, SpyGlass flags similar situations for the `priority-if`, `unique-case`, and `unique-if` statements.

Message Details

`Used <Priority | Unique> Type <if | case> statement but all the conditions are not covered`

Severity

SynthesisWarning

SYNTH_12606

This warning occurs when you assign a value to a variable inside an if(clock'event) block, and then attempt to read it after the end-if statement.

Language

Verilog, VHDL

Rule Description

Move the statement that reads the value into the if block, or change the variable into a signal and read the signal outside the process.

For example:

```
PROCESS (hclk)
    VARIABLE local_counter      : STD_LOGIC_VECTOR(11 DOWNTO
0);
    BEGIN
        IF (hclk'event AND hclk = '1') THEN
            IF (hreset = '1') THEN
                local_counter := "111111111111";
            ELSE
                local_counter := local_counter - 1;
            END IF;
        END IF;
    boot_up_timer_val <= local_counter;


END PROCESS;
```

Message Details

```
Tried to use a synchronized value
```

Severity

SynthesisWarning

SYNTH_12607

Out-of-range dimension specified for array query function.

Language

Verilog, VHDL

Message Details

```
Out-of-range dimension specified for array query function
'<function>'
```

Severity

SynthesisWarning

SYNTH_12608

The logic of the always block mismatches with the type of the always block

Description

The *SYNTH_12608* rule reports a violation when the logic within the `always` block mismatches with the logic expected in the `always_comb`, `always_latch`, or `always_ff` blocks.

Language

Verilog

Messages and Suggested Fix

The *SYNTH_12608* rule reports the following violation:

```
[WARNING] The logic of the always block mismatches with the
type of Always Block (which should be "always_<comb
(Combinational Block) | latch (Latch) | ff>
```

Consequences of Not Fixing

If you do not fix this violation, the logic within the reported block is ignored during synthesis.

How to Debug and Fix

To fix this violation, specify a valid logic that matches with the logic expected in the `always_comb`, `always_latch`, or `always_ff` blocks.

For `always_comb`, only combinational logic is valid and the logic should not produce any latches and flip-flops.

For `always_latch`, logic should be such that latches are produced. Combinational logic can be present in such blocks.

Example Code and/or Schematic

Example 1

Consider following example:

```
module test(in1,in2,clk, out1);
  input in1, in2, clk;
  output reg out1;
  always_comb
  begin
    if(clk)
      out1 = in1;
  end

  always_latch
  begin
    out1 = in1 & in2;
  end
endmodule
```

For the above example, the *SYNTH_12608* rule reports a violation for both `always_comb` and `always_latch` because the logic described mismatches with their definitions.

Example 2

Consider following example:

```
module test(in1, in2, in3, reset, clk, out1, out2, out3);
  input [3:0] in1, in2, in3;
  input clk, reset;
  output reg [3:0] out1, out2, out3;
  always_comb
     begin
       if(clk)
       out1 = in1;
     end
  always_latch
     begin
       out2 = in1 & in2;
```

```
  end
always_ff@(posedge clk or posedge reset)
  begin
     if(reset)
     out3 = in3;
  end
endmodule
```

In the above example, the *SYNTH_12608* rule reports a violation for `always_comb`, `always_latch`, and `always_ff` because the logic described mismatches with their definitions.

NOTE: *In this example, a violation appears for only for one bit so that there is less noise.*

Default Severity Label

SynthesisWarning

SYNTH_12609

Clocking blocks will be ignored for synthesis

Language

Verilog, VHDL

Message Details

```
Clocking blocks will be ignored for synthesis
```

Severity

SynthesisWarning

SYNTH_12610

Sequence blocks will be ignored for synthesis

Language

Verilog, VHDL

Message Details

```
Sequence blocks will be ignored for synthesis
```

Severity

SynthesisWarning

SYNTH_12611

Property blocks will be ignored for synthesis

Language

Verilog, VHDL

Message Details

```
Property blocks will be ignored for synthesis
```

Severity

SynthesisWarning

SYNTH_12612

This warning gets flagged when the designer uses multiple/ redundant uses of UNIQUE's 'unique if' and PRIORITY's in 'priority if'.

Language

Verilog

Rule Description

SpyGlass generates this synthesis warning to indicate the redundant usage of unique/priority keywords.

For example:

```
module redundant_unique (input integer sel, output bit [3:0]
out1);
always @*
  begin
    unique if (sel == 2)
      out1 = 2;
    else unique  if (sel == 4)
      out1 = 5;
    else unique  if (sel == 6)
      out1 = 6;
    else
      out1 = 0;
  end
endmodule

module redundant_priority (input integer sel, output bit
[3:0] out1);
always @*
  begin
    priority if(sel == 4)
      out1 = 2;
    else priority if (sel == 4)
      out1 = 5;
    else priority if (sel == 6)
      out1 = 6;
```

```
    else
      out1 = 0;
  end
endmodule
```

Message Details

```
Repetitive usage of keyword '<keyword>' is not required (some
synth tools might not synthesize this construct)
```

Severity

SynthesisWarning

SYNTH_12613

fork and join constructs are not synthesizable. Ignoring them for synthesis.

Language

Verilog, VHDL

Message Details

```
fork and join constructs are not synthesizable. Ignoring them
for synthesis.
```

Severity

SynthesisWarning

SYNTH_12801

Result of the T'VAL, T'SUCC, and T'PRED VHDL attributes must be within the T'LOW and T'HIGH range.

Description

The *SYNTH_12801* rule reports a violation when the result of the `T'VAL`, `T'SUCC`, and `T'PRED` VHDL attributes is not within the range of `T'LOW` and `T'HIGH`.

Language

VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Result of T'<VAL|SUCC|PRED> must be within range T'LOW and T'HIGH`

Consequences of not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, pass values within the `T'LOW` and `T'HIGH` range to VHDL attributes.

Example Code and/or Schematic

Consider the following example:

```
library IEEE;
use IEEE.std_logic_1164.all;
package p is
type color is (violet, blue, green, yellow, red, white);
end p;
```

```
library IEEE;
use IEEE.std_logic_1164.all;
use work.p.all;
entity top is
port (z :out color);
end top;
architecture behav of top is
begin
test:process
variable v : integer;
begin
v := 10;
z <= color'val(v);
end process test;
end behav;
```

For the above example, the *SYNTH_12801* rule reports a violation because the `v=10` is passed to `color'val` whereas the value assigned should be in the [0:5] range.

Default Severity Label

SynthesisError

SYNTH_12802

The T'VAL, T'POS, T'SUCC, and T'PRED VHDL attributes are supported for only enumerated types

Description

The *SYNTH_12802* rule reports a violation if the `T'VAL`, `T'POS`, `T'SUCC`, and `T'PRED` VHDL attributes are used with a non enumerated type.

These attributes are supported only for enumerated types.

Language

VHDL

Messages and Suggested Fix

[SynthesisError] `T'<VAL|POS|SUCC|PRED> attribute is only supported for enumerated types`

Consequences of not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, perform any of the following actions:

- Use the reported attribute for enumerated types only.
- Remove the reported attribute that is being used with a non enumerated type.

Example Code and/or Schematic

Consider the following example:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity top is
```

```
port (x : in integer;
z :out integer);
end top;
architecture behav of top is
begin
test:process
begin
z <= integer'val(x);
-- SYNTH_12802 violation for use of 'VAL with
-- non enumerated type
end process test;
end behav;
```

For the above examples, the *SYNTH_12802* rule reports a violation because `'val` is used with the `i` integer type, but these attributes are supported only for enumerated types.

Default Severity Label

SynthesisError

SYNTH_12803

Vector size mismatch on a port during last call.

Description

The *SYNTH_12803* rule reports a violation if there is a mismatch of vector size of a port.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Vector size mismatch on port '<port>' during last call

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Specify the correct vector size for the ports.

Example Code and/or Schematic

Consider the following example of a Verilog module:

```
module leaf(in[3:0], out[3:0]);
parameter p = 4;
input [p-1:0] in;
output [p-1:0] out;
assign out = in;
endmodule
module top(in1, out1);
parameter p1 = 3;
input [p1-1 : 0] in1;
output [p1-1 : 0] out1;
leaf #(p1) inst1 (in1, out1);
endmodule
```

In the above example, the *SYNTH_12803* rule reports violation because the `in1` input and the `out1` output declared as [3:0]. However, during instantiation, the value `[p1=3]` is passed thereby changing the declaration to [2:0]. This results in the vector size mismatch.

Default Severity Label

SynthesisError

SYNTH_12804

MSB or LSB of part-select on a port is out of bounds.

Language

Verilog, VHDL

Message Details

```
MSB or LSB of part-select on port '<port>' is out of bounds
```

Severity

SynthesisError

SYNTH_12805

Elements of aggregate do not cover all cases.

Description

The *SYNTH_12805* rule reports a violation if the elements of an aggregate do not cover all the cases.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

`[SynthesisError] Elements of aggregate do not cover all cases`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Update the logic so that all the cases are covered by the elements of an aggregate.

Example Code and/or Schematic

Consider the following example:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity test is
generic (test_gen:integer:=6);
port (
bus1     : out    std_logic_vector (9 downto 1)
);
end test;
architecture rtl of test is
function ret_const2 (arg:integer) return integer is
variable v1:integer :=8;
begin
```

```
v1:=arg ;
return v1;
end ret_const2;
component comp
generic(
param1 : INTEGER RANGE 1 TO 150 := test_gen ;
param2 : INTEGER RANGE 0 TO 1   := 0
);
port (
PO_DFT_TEN     : out    std_logic_vector
((ret_const2(param1)) downto (1-ret_const2(param2*2)))
);
end component;
begin
comp0 : comp
generic map(
param1 => 11,
param2 => 0)
port map (
PO_DFT_TEN(6 downto 2)  =>bus1(5 downto 1),
PO_DFT_TEN(1)  =>bus1(6)
);
end rtl;
```

In the above example, the *SYNTH_12805* rule reports a violation because all the cases are not covered by the elements of the aggregate.

Default Severity Label

SynthesisError

SYNTH_12806

Multiple declarations of same design unit found across languages.

Language

Verilog, VHDL

Message Details

```
Multiple declarations of same design unit <design-unit> found
across languages.
```

Severity

SynthesisError

SYNTH_12807

Port not found in the black box master '<master>' instantiated across languages.

Language

Verilog, VHDL

Message Details

```
Port <port> not found in the black box master '<master>'
instantiated across languages.
```

Severity

SynthesisError

SYNTH_12808

No port found corresponding to a connection in the black box master instantiated across languages.

Language

Verilog, VHDL

Message Details

```
No port found corresponding to the connection <connection> in
the black box master '<master>' instantiated across languages.
```

Severity

SynthesisError

SYNTH_12809

Port size does not match with that of connection of a size, instantiated in Verilog as a black box.

Language

Verilog, VHDL

Message Details

```
Port size (<port-size>) does not match with that of connection
<connection> of size <size>, instantiated in Verilog as black
box '<black-box>'.
```

Severity

SynthesisError

SYNTH_12810

Literal is not supported for synthesis.

Description

The *SYNTH_12810* rule reports a violation if a literal is used in the code.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] `Literal <literal> is not supported for synthesis`

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the reported literal from the code.

Example Code and/or Schematic

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;
entity ent is
port (
clk     : in  std_logic;
q2      : out std_logic bus);
end ent;
architecture arc of ent is
begin
process(clk)
begin
if clk'event and clk = '0' then
q2 <= null;
```

```
end if;
end process;
end arc;
```

In the above example, the *SYNTH_12810* rule reports a violation because of the usage of a literal.

Default Severity Label

SynthesisError

SYNTH_12811

Size mismatch in logical operation.

Description

The *SYNTH_12811* rule reports a violation if there is a size mismatch in a logical operation.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Size mismatch in logical operation.

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Update the logic to remove the size mismatch.

Example Code and/or Schematic

Consider the following example:

```
library ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
ENTITY test IS
PORT(
in1  : IN   STD_LOGIC_vector(30 DOWNTO 0);
out1    : OUT  STD_LOGIC_VECTOR(31 DOWNTO 0)
);
END test;
ARCHITECTURE synth OF test IS
FUNCTION TO_STDLOGICVECTOR  (ARG: NATURAL;   SIZE: NATURAL)
RETURN STD_LOGIC_VECTOR IS
BEGIN
```

```
RETURN STD_LOGIC_VECTOR (CONV_UNSIGNED (ARG, SIZE));
END;
CONSTANT BT_HW_VERSION                : STD_LOGIC_VECTOR:=
TO_STDLOGICVECTOR(16#0000#,32);
BEGIN
out1 <= (BT_HW_VERSION  or in1);
END synth;
```

In the above example, the *SYNTH_12811* rule reports a violation because of the size mismatch in the logical operation.

Default Severity Label

SynthesisError

SYNTH_12812

Type mismatch for string type parameter of an instance. Not supported.

Language

Verilog, VHDL

Message Details

```
Type mismatch for string type parameter '<parameter>' of
instance <instance>. Not supported
```

Severity

SynthesisError

SYNTH_12820

Cannot open source file for reading.

Language

Verilog, VHDL

Message Details

```
Cannot open source file '<file>' for reading
```

Severity

SynthesisError

SYNTH_12821

Illegal use of an object declared in a region ignored by synthesis.

Language

Verilog, VHDL

Message Details

```
Illegal use of object '<object>' declared in a region ignored
by synthesis
```

Severity

SynthesisError

SYNTH_12822

Non synthesizable use of a parameter in a clock construct.

Description

The *SYNTH_12822* rule reports a violation if a parameter is encountered in a clock construct.

A parameter should not be used inside a clock construct because the parameter is a fixed value and using it with posedge or negedge of a clock has no meaning.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_12822* rule reports the following message:

[ERROR] Non-synthesizable use of parameter '<parameter>' in clock construct

Consequences of Not Fixing

If you do not fix this violation, the reported parameter is ignored during synthesis.

How to Debug and Fix

To fix this violation, remove parameters from clock constructs. You must provide signals in such constructs.

Example Code and/or Schematic

Consider the following example:

```
module test( out, clk, in);
  output out;
  input clk, in;
  reg out;
  parameter p = 4'b0000;
```

```
  always @(posedge p[2])
  begin
    out = in;
  end
endmodule
```

For the above example, the *SYNTH_12822* rule reports a violation because the `p` parameter is used in the clock construct, `(always @(posedge p[2]))`.

Since posedge requires a signal, whereas a parameter is a constant value, this is a non synthesizable use of the parameter.

Default Severity Label

SynthesisError

SYNTH_12823

Resolution function is not written with the required number of drivers.

Description

The *SYNTH_12823* rule reports a violation if a resolution function does not cover all the drivers.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_12823* rule reports the following message:

[ERROR] `Resolution function '<function>()' used with <num1> drivers instead of <num2>`

Where:

- *<num1>* is the number of drivers that use resolution function.
- *<num2>* is number of drivers with which resolution function is expected to be used

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, update the reported resolution function to cover all the drivers.

Example Code and/or Schematic

Consider the following example:

```
library IEEE;
```

```
use IEEE.std_logic_1164.all;

entity test is
  port (in1, in2, in3 : in bit);
end test;

architecture test_f of test is
function myres(drivers : bit_vector( 0 to 1))return bit is
variable ac : bit := '0';
begin
  for i in 0 to 1 loop
    ac := ac or drivers(i);
  end loop;
    return (ac);
end myres;

signal temp : myres bit;
begin
    temp <= in1 and in2;
    temp <= in1 or in3;
    temp <= in2;
end test_f;
```

In the above example, the resolution function is written for two drivers. However, there are the following three drivers:

- `temp <= in1 and in2;`
- `temp <= in1 or in3;`
- `temp <= in2;`

Therefore, the *SYNTH_12823* rule reports a violation in this case.

Default Severity Label

SynthesisError

SYNTH_12824

Enumerated name has a value, which is already assigned to a previous member of the enumeration

Language

Verilog, VHDL

Message Details

```
Enumerated name '<name>' have a value '<value>' which is
already assigned to a previous member of the enumeration
```

Severity

SynthesisError

SYNTH_12825

Enumerated name has value that is more than the range of enum.

Description

The *SYNTH_12825* rule reports a violation when the value for an enumerated name exceeds the range of the enum.

Language

Verilog, VHDL

Messages and Suggested Fix

The *SYNTH_12825* rule reports the following message:

[ERROR] Enumerated name '<name>' have a value '<value>' which is more than the range specified for enumeration

Consequences of Not Fixing

If you do not fix this violation, the reported enumerated name is ignored during synthesis.

How to Debug and Fix

To fix this violation, update the value of the enumerated name so that the value is within the range of the enum.

Example Code and/or Schematic

Consider the following example:

```
module test();
  parameter p1=4;
  localparam p3=3;
  enum logic [2:0] {d=p1+p3,e} var1;
endmodule
```

For the above example, the *SYNTH_12825* rule reports a violation for the enumerated name `e` in `var1` as `var1` of width `3` can have value in the

[0-7] range, but the value 8 for e is more than this range.

Default Severity Label

SynthesisError

SYNTH_12826

Common expression has a value, which does not match with the corresponding dimension width of array.

Language

Verilog, VHDL

Message Details

```
Common expression have a value '<value>' which does not match
with the corresponding dimension width ('<width>') of array
```

Severity

SynthesisError

SYNTH_12827

Structure Member does not have corresponding element in assignment literals

Language

Verilog, VHDL

Message Details

```
Structure Member '<member>' does not have corresponding element
in assignment literals
```

Severity

SynthesisError

SYNTH_12828

Mismatch in number of literals in assignments

Language

Verilog, VHDL

Message Details

`Mismatch in number of literals in assignments`

Severity

SynthesisError

SYNTH_12829

Less number of elements in unpacked aggregate expression list

Language

Verilog, VHDL

Message Details

`Less number of elements in unpacked aggregate expression list`

Severity

SynthesisError

SYNTH_12830

More number of elements in unpacked aggregate expression list

Language

Verilog, VHDL

Message Details

```
More number of elements in unpacked aggregate expression list
```

Severity

SynthesisError

SYNTH_12831

Module with a generic interface as port should have instantiation

Description

The *SYNTH_12831* rule reports a violation if a module with a generic interface as a port does not have instantiation.

Language

Verilog

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] Module '<module>' with generic interface '<interface>' cannot be synthesized unless it has proper instantiation

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Update the logic so that the reported module has an instantiation.

Example Code and/or Schematic

```
interface intf_t ;
bit [1:0] req;
parameter pa =4;
generate
if (pa==4) begin: mps
modport client_mp (output req);
end
endgenerate
endinterface
module m1 (interface intf1);
endmodule
module bus ;
```

```
intf_t intf();
m1 I1 (.intf1 (intf.mps.client_mp));
endmodule
```

Default Severity Label

SynthesisError

SYNTH_12832

(Min:Typ:Max) constructs are not supported for synthesis

Description

The *SYNTH_12832* rule reports a violation if the *min:typ:max* type of constructs are specified.

Language

Verilog, VHDL

Messages and Suggested Fix

This rule reports the following violation:

[SynthesisError] (Min:Typ:Max) constructs are not supported for synthesis

Consequences of Not Fixing

If you do not fix this violation, the module in which this violation is reported is considered as a black box for which no SpyGlass analysis is done.

How to Debug and Fix

Remove the reported type of constructs from the code.

Example Code and/or Schematic

Consider the following example:

```
module test(input in, output reg out);
always
out = (5:4:2);
endmodule
```

In the above example, the *SYNTH_12803* rule reports violation because the *min:typ:max* type of constructs used in the assignment `out = (5:4:2)`.

Default Severity Label

SynthesisError

SYNTH_12833

Usage of 'ShortReal' is not synthesizable

Language

Verilog, VHDL

Message Details

`Usage of 'ShortReal' is not synthesizable`

Severity

SynthesisError

SYNTH_12834

Interface with a generic interface as port should have proper instantiation

Language

Verilog, VHDL

Rule description

Interface with a generic interface as port should have proper instantiation.

Message Details

```
Interface '<interface>' with generic interface
'<generic-interface>' cannot be synthesized unless it has
proper instantiation
```

Severity

SynthesisError

SYNTH_12835

Genvar assignments are not supported in synthesis

Language

Verilog

Message Details

```
Genvar assignments are not supported in synthesis
```

Severity

SynthesisError

SYNTH_12836

Interface with generic interface port cannot be resolved

Description

The *SYNTH_12836* rule reports a violation when an interface with generic interface port is specified, which cannot be resolved.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] Interface '<interface-name>' with generic interface port '<port-name>' cannot be resolved

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, make sure that no interface with generic interface port is specified.

Example Code and/or Schematic

Consider the following example:

```
interface intf(interface intf2_inst);
 wire in1;
 modport m1 (input in1);
 modport m2 (output in1);
 endinterface

 module mid(intf I1);
```

```
endmodule
```

In the above example, interface `intf` has generic interface port `intf2_inst` which cannot be resolved.

Default Severity Label

SynthesisError

SYNTH_12839

Synthesis failed for module since it has unsupported assignment pattern expression.

Language

Verilog, VHDL

Rule Description

This rule reports a violation when synthesis fails for module because the module contains unsupported assignment pattern expressions.

Message Details

```
Error in Assignment Pattern Expression
```

Severity

SynthesisError

SYNTH_12840

Width of interface port does not match size of connection expression

Description

The *SYNTH_12840* rule reports a violation when the width of an interface port does not match with the size of a connection expression.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] `Width of interface port (<port-name>) does not match size of connection expression`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

Modify the code so that the width of the reported port matches with the size of the connection expression.

Example Code and/or Schematic

Consider the following example:

```
interface intf(input logic a, output logic b);
  modport add(input a, output b);
endinterface
module test(input logic[1:0] clk1,clk2,output logic d1,d2);
  intf i1(clk1,d1);
endmodule
```

For the above example, the *SYNTH_12840* rule reports a violation for the interface instance `i1` because the connected expression `clk1` is of the size two while the interface port `a` is of the size one.

Default Severity Label

SynthesisError

SYNTH_12841

Tran gate is not supported for technology mapping in optimized synthesis flow

Description

The *SYNTH_12841* rule reports a violation when a `tran` gate is found in technology mapping.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] `Tran Gate is not supported for technology mapping`

Consequences of Not Fixing

If you do not fix this violation, the module containing the `tran` gate is considered as a black box for which no further rule checking is done.

How to Debug and Fix

To fix this violation, remove the `tran` gate from the module.

Example Code and/or Schematic

For the following example, this rule report a violation because of the usage of the `tran` gate:

```
module test(ino1,ino2);
   inout ino1,ino2;
   tran (ino1,ino2);      <-- SYNTH_12841 reported
endmodule
```

Default Severity Label

SynthesisError

SYNTH_12842

Error in unpacked concatenation expression

Description

The *SYNTH_12842* rule reports a violation if unpacked concatenation expression used in a module is unsupported.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] `Synthesis failed for module since it has unsupported unpacked concatenation expression`

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, make sure that the module does not have an unsupported unpacked concatenation expression.

Default Severity Label

SynthesisError

SYNTH_12843

Unresolved port is found in interface

Description

The *SYNTH_12843* rule reports a violation when an unresolved port is found in the interface.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[SynthesisError] Width of interface port (<port-name>) does not match size of connection expression

Consequences of Not Fixing

If you do not fix this violation, the module containing the reported issue is considered as a black box. Therefore, no SpyGlass checking is done on this module.

How to Debug and Fix

To fix this violation, make sure that the interface does not have any unresolved port.

Example Code and/or Schematic

Consider the following example:

```
module mid (intf i);
 endmodule

 interface intf(intf2.m4 intf2_inst);
 wire in1;
 endinterface
```

```
interface intf2(output out1);
modport m3(inout out1);
endinterface
```

In the above example, interface `intf` contains port `intf2_inst`, which is unresolved since interface `intf2` doesn't contain any modport named `m4`.

Default Severity Label

SynthesisError

ELAB_6202

Infinite for loop found in the design.

Language

Verilog, VHDL

Message Details

`Infinite for loop found in the design`

Severity

ElaborationWarning

ELAB_6203

For Generate range is greater than the maximum allowable limit (2048).

Language

Verilog, VHDL

Message Details

`For Generate range is greater than the maximum allowable limit (2048)`

Severity

ElaborationWarning

ELAB_6205

PortParam will be ignored, parameter value will remain unchanged.

Language

Verilog

Message Details

Consider the following design:

```
typedef int myint;
interface test_inf  ( );
parameter integer W1 = 4;
parameter [3:0] W2 = 4'b0100;
parameter myint W3 = 4;


  wire [W1-1:0] in1;
  wire [W2-1:0] in2;
  wire [W3-1:0] in3;
endinterface


module top (test_inf P1, output logic P2);
test I1 (.intf1(P1));
endmodule


module test ( test_inf intf1);
endmodule
```

In addition, consider the following portparam definition:

```
set_option portparam {"test.P1.W1=7" "top.P3.W1=1"
"top.P2.W2=4'b0011" "top.P1.W0=3" "top.P1.W2[0]=1'b1"
"top.P1.W3=4" "top.P1.W1=xxxx"}
```

For the above design, the ELAB_6205 rule reports the following messages:

Message 1

If the module name mentioned in the portparam option is not a top module

or any module with that name does not exist, the ELAB_6205 rule reports the following violation:

```
portParam input 'test.P1.W1=7' will be ignored (Reason : Module 'test' is not a top level module)
```

Message 2

If the port name mentioned in the portparam option does not exist in top module, the ELAB_6205 rule reports the following violation:

```
portParam input 'top.P3.W1=1' will be ignored (Reason : Port 'P3' is not found in Module 'top')
```

Message 3

If the port name specified in the portparam option is not of SV-Interface type, the ELAB_6205 rule reports the following violation:

```
portParam input 'top.P2.W2=4'b0011' will be ignored (Reason : Port 'P2' in module 'top' is not of SV Interface type)
```

Message 4

If the parameter name specified in the portParam input does not exist in SV Interface type port, the ELAB_6205 rule reports the following violation:

```
portParam input 'top.P1.W0=3' will be ignored (Reason : Parameter 'W0' is not found in SV Interface type port 'P1')
```

Message 5

If a parameter is partially overridden by using the portparam option, the ELAB_6205 rule reports the following violation:

```
portParam input 'top.P1.W2[0]=1'b1' will be ignored (Reason : Partial overriding of parameter W2[0] is done)
```

Message 6

If the parameter type is complex (such as, struct/union/user-defined/ enum/multi-dim) then portParam wont work to override such type of parameters and the ELAB_6205 rule reports the following violation:

```
portParam input 'top.P1.W3=4' will be ignored (Reason : Paramter 'W3' is of USER-DEFINED Type which is not supported)
```

Message 7

If the value assigned to parameter is invalid (only integer and base numbers are allowed), the ELAB_6205 rule reports the following violation:

```
portParam input 'top.P1.W1=xxxx' will be ignored (Reason :
Incorrect value 'xxxx' assigned to Parameter 'W1')
```

Severity

ElaborationWarning

ELAB_6206

Option dummy_top will be ignored, hierarchy of top module will not have any effect of dummy top.

Language

Verilog

Message Details

Consider the following design:

```
typedef int myint;
interface test_inf  ( );
parameter W1 = 4;
parameter type ptype = logic;

  wire [W1-1:0] w1;
  ptype w2;
endinterface

module top_wrapper ();
test_inf #(.W1(16), .ptype(myint)) inf1();
top inst1 (.bus(inf1));
endmodule

module top ( test_inf bus);
endmodule
```

In addition, consider the following `dummy_top` and the `top_instance` set_option commands in the project file:

```
set_option top top
set_option dummy_top top_wrapper
set_option top_instance inst0
```

For the above design, the ELAB_6206 rule reports the following message:

```
Option dummy_top will be ignored (Reason : Instance 'inst0' is
not found in dummy module 'top_wrapper' in module scope)
```

Severity

ElaborationWarning

ELAB_6302

Real type generics are not supported for synthesis.

Rule Description

The *ELAB_6302* rule reports a violation when the type of a generic is specified as real.

Language

VHDL

Messages and Suggested Fix

[ElaborationError] `Real type generic <generic> is not supported for synthesis`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Modify the code to remove the reported real-type generic.

Example Code and/or Schematic

Consider the following example:

VHDL

```
#######################################
library ieee;
use ieee.std_logic_1164.all;
entity e1 is
generic (r1 : real:=2.0 );
```

```
port (in1: in std_logic;
out1 : out std_logic);
end e1;
architecture struct of e1 is
begin
out1 <= in1;
end struct;
########################################
```

In the above example, the *ELAB_6302* rule reports a violation because the `r1` generic is of the type real.

Default Severity Label

ElaborationError

ELAB_6303

A generic interface, which is being used as a module port, is not resolved.

Description

The *ELAB_6303* rule reports a violation in the following cases.

1. If a top module contains a generic interface port, as shown in the following example:

   ```
   module M(interface a)
   ...
   endmodule
   ```

   Here, module `M` is a top-level module in the hierarchy.

2. If a non-top module contains a generic interface port but the place where it is instantiated, a proper interface port expression is not being passed. This is shown in the following example:

   ```
   module Top;
   ...
     wire a;
     M I(a);
   ...
   endmodule
   ```

Here, port `a` of module `M` is unresolved as port connection expression `a` is not proper.

Language

Verilog

Messages and Suggested Fix

[ElaborationError] `Unresolved generic interface port found in the module '<module>'`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module

becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Double-click on the violation to view the violating code. Review the code and update it so that the reported module does not contain any unresolved generic port.

Example Code and/or Schematic

Consider the following example:

```
Verilog
#######################################################
interface intf1 ;
logic [3:0] logic1;
logic [3:0] logic2;
modport m1 (input logic1,output logic2);
endinterface
module test (interface if1, input in1,output out1);
endmodule
#######################################################
```

For the above example, the *ELAB_6303* rule reports a violation because a generic interface is used as a module port in the `test` module.

Default Severity Label

ElaborationError

ELAB_6304

Generic parameter should be initialized and able to be evaluated.

Description

The *ELAB_6304* rule flags the following scenarios:

- A top entity contains an uninitialized generic parameter.

  See *Example 1*.
- A module is instantiated with an uninitialized generic parameter.

  See *Example 1*.
- A module is initialized with a value that cannot be evaluated.

  See *Example 2*.

Rule Exceptions

However, this rule does not report a violation in the following cases:

- If the generic has a default value in the module
- If the generic is initialized in VHDL component declarations
- If a value is passed from the hierarchy above

Language

Verilog, VHDL

Messages and Suggested Fix

The following message appears if an uninitialized generic parameter is used in the top module:

`Uninitialized generic parameter <parameter> for Entity <entity>`

The following message appears if an uninitialized generic parameter is used in an instance module:

`Uninitialized generic parameter <parameter> for Instance <instance>`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Initialize the reported generic parameter before using it in the reported module or instance.

Example Code and/or Schematic

Example 1

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all;


entity bottom is
generic(param1 : integer );
end bottom;

architecture rtl of bottom is
begin
end rtl;

library ieee;
use ieee.std_logic_1164.all;

entity top is
generic(param1 : integer;   -- uninitialized parameter for
                            -- entity 'top'
            param2 : integer := 10);
end top;
```

```
architecture rtl of top is
component bottom
generic(param1 : integer);
end component;
begin
I1 : bottom generic map (param1 =>param1); --uninitialized
                              -- parameter for instance 'I1'
I2 : bottom generic map (param1 =>param2);
end rtl;
```

In the above example:

- Generic `param1` of the top-level entity top is uninitialized. Therefore, *ELAB_6304* reports the following violation:

  ```
  Uninitialized generic parameter 'param1' for Entity 'top'
  ```

- `param1` in the `I1` instance of the `bottom` entity is uninitialized. Therefore, *ELAB_6304* reports the following violation:

  ```
  Uninitialized generic parameter 'param1' for Instance 'I1'
  ```

Example 2

Consider the following example:

```
library ieee;
use ieee.std_logic_1164.all,ieee.numeric_std.all;
entity ent is
  generic ( GEN : unsigned ( 2 downto 0) );
end ent;

architecture arc of ent is
begin
end arc;

library ieee;
use ieee.std_logic_1164.all,ieee.numeric_std.all;

entity top is
    port( a_top, b_top : in unsigned ( 2 downto 0);
```

```
          div_top : out unsigned ( 2 downto 0) );
end top;

architecture structure_top of top is
    component ent is
        generic ( GEN : unsigned ( 2 downto 0));
    end component;

begin
    INST : ent generic map ( GEN => a_top/b_top );
end structure_top;
```

In the above example, the GEN generic parameter of the INST instance is initialized with an expression that cannot be evaluated. Therefore, *ELAB_6304* reports the following violation:

```
Uninitialized generic parameter 'GEN' for Instance 'INST'
```

Default Severity Label

ElaborationError

ELAB_6306

Circular dependency found.

Language

Verilog

Rule Description

This message indicates that a design unit is instantiating itself (with a different set of values for parameter). Since parameter values are being changed during instantiation, it is possible that a terminating condition (for this self instantiation) is achieved (through use of "generate"). However, if such terminating condition is not achieved within 1024 recursions of self-instantiations, this message appears. This results in termination of SpyGlass execution, with a message on the screen pointing to the possibility of Circular Dependency.

Message Details

```
Circular dependency found for the Verilog design unit '<du-
name>'
```

Severity

ElaborationError

ELAB_6307

Circular dependency found.

Language

VHDL

Rule Description

This message indicates that a design unit is instantiating itself (with a different set of values for generic). Since generic values are being changed during instantiation, it is possible that a terminating condition (for this self instantiation) is achieved (through use of "generate"). However, if such terminating condition is not achieved within 1024 recursions of self-instantiations, this message appears. This results in termination of spyglass execution, with a message on the screen pointing to the possibility of Circular Dependency.

Message Details

```
Circular dependency found for VHDL design unit 'Library Name:
<lib-name>' 'Entity Name: <entity-name>' 'Architecture Name:
<arch-name>'
```

Severity

ElaborationError

ELAB_6312

Unsupported SV constructs found during Elaboration.

Description

The *ELAB_6312* rule reports unsupported SystemVerilog constructs found during design elaboration.

Language

Verilog

Messages and Suggested Fix

This rule reports the following message:

[ElaborationError] `Unsupported SV constructs '<SV-construct>' found during Elaboration`

Consequences of Not Fixing

If you do not fix this violation, the current module and the parent module becomes a black box. Therefore, no SpyGlass checking is done on this hierarchy.

For example, consider `x` as the top-level module that contains the instance `y`. The parent of `y` is `z`. The consequence of not fixing this violation will be that `x` and `z` will become a black box.

How to Debug and Fix

Remove the reported construct from the code.

Example Code and/or Schematic

Consider the following example:

```
module design_top(input int data[3:0],input int
bus[3:0],input bit[1:0] sel,output int out);
always@(*)
   begin
```

```
      if((sel>0) &&& (sel<=2)) begin
           out1 = in1[1]+in2[2];
        end
      end
endmodule
```

For the above example, the *ELAB_6312* rule reports a violation for the unsupported SystemVerilog construct `&&&`.

Default Severity Label

ElaborationError

Liberty File Parsing Built-In Messages

Overview

The rules of this category reports violations based on the parsing done on Synopsys Library™ format library files (.lib files).

LIBWRN_1

Library attribute current_unit should have "1uA", "10uA", "100uA", "1mA", "10mA","100mA", or "1A" as its value

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `current_unit` library attribute.

The correct values are:

1uA	10uA	100uA	1mA	10mA	100mA	1A

Message Details

```
Incorrect current_unit value '<value>' in library '<library>'
```

Severity

Warning

LIBWRN_2

Value of the library attribute delay_model should be one of "generic_cmos", "lsi_cmde", "table_lookup", "cmos2", "piecewise_cmos", "dcm" or "polynomial"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `delay_model` library attribute.

The correct values are:

generic_cmos	lsi_cmde	table_lookup	cmos2	piecewise_cmos
dcm	polynomial			

Message Details

```
Incorrect delay_model '<value>' in library '<library>'
```

Severity

Warning

LIBWRN_3

Value of the library attribute in_place_swap_mode should be one of "match_footprint", "ignore_footprint", or "no_swapping"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `in_place_swap_mode` library attribute.

The correct values are:

match_footprint	ignore_footprint	no_swapping

Message Details

```
Incorrect in_place_swap_mode '<value>' in library '<library>'
```

Severity

Warning

LIBWRN_4

Value of the library attribute leakage_power_unit should be one of "1pW", "10pW", "100pW", "1nW", "10nW", "100nW", "1uW", "10uW", "100uW", or "1mW"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `leakage_power_unit` library attribute.

The correct values are:

1pW	10pW	100pW	1nW	10nW	100nW
1uW	10uW	100uW	1mW		

Message Details

```
Incorrect leakage_power_unit '<value>' in library '<library>'
```

Severity

Warning

LIBWRN_5

Value of the library attribute piece_type should be one of "piece_length", "piece_total_cap", "piece_wire_cap", or "piece_pin_cap"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `piece_type` library attribute.

The correct values are:

piece_length	piece_total_cap	piece_wire_cap	piece_pin_cap

Message Details

```
Incorrect piece_type '<value>' in library '<library>'
```

Severity

Warning

LIBWRN_6

Value of the library attribute preferred_output_pad_slew_rate_control should be one of "none", "high", "low", or "medium"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `preferred_output_pad_slew_rate_control` library attribute.

The correct values are:

none	high	low	medium

Message Details

```
Incorrect preferred_output_pad_slew_rate_control '<value>' in
library '<library>'
```

Severity

Warning

LIBWRN_7

Value of the library attribute pulling_resistance_unit should be one of "1ohm", "10ohm", "100ohm", or "1kohm"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `pulling_resistance_unit` library attribute.

The correct values are:

1ohm	10ohm	100ohm	1kohm

Message Details

```
Incorrect pulling_resistance_unit '<value>' in library
'<library>'
```

Severity

Warning

LIBWRN_8

Value of the library attribute simulation should be either "true" or "false"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `simulation` library attribute.

The correct values are `true` and `false`.

Message Details

```
Incorrect value for simulation '<value>' in library '<library>'
```

Severity

Warning

LIBWRN_9

Value of the library attribute time_unit should be one of "1ps", "10ps", "100ps", or "1ns"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `time_unit` library attribute.

The correct values are:

1ps	10ps	1ns

Message Details

```
Incorrect time_unit '<value>' in library '<library>'
```

Severity

Warning

LIBWRN_10

Value of the library attribute voltage_unit should be one of "1mV", "10mV", "100mV", or "1V"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `voltage_unit` library attribute.

The correct values are:

1mV	10mV	1V

Message Details

```
Incorrect voltage_unit '<value>' in library '<library>'
```

Severity

Warning

LIBWRN_11

Value of resource of the library attribute define_cell_area should be one of "pad_slots", "pad_driver_sites", "pad_input_driver_sites" or "pad_output_driver_sites"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `define_cell_area` library attribute.

The correct values are:

pad_slots	pad_driver_sites	pad_input_driver_sites
pad_output_driver_sites		

Message Details

```
Incorrect define_cell_area '<value>' in library '<library>'
```

Severity

Warning

LIBWRN_12

Value of the library attribute default_wire_load_mode should be one of "top", "segmented", or "enclosed"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `default_wire_load_mode` library attribute.

The correct values are:

top	segmented	enclosed

Message Details

```
Incorrect value for default_wire_load_mode '<value>' in library
'<library>'
```

Severity

Warning

LIBWRN_13

Value of cell attribute auxiliary_pad_cell should be either "true" or "false"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `auxiliary_pad_cell` library attribute.

The correct values are `true` and `false`.

Message Details

```
Incorrect value for auxiliary_pad_cell '<value>' in cell
'<cell>'
```

Severity

Warning

LIBWRN_14

Value of the cell attribute dont_fault should be one of "sa0", "sa1", or "sa01"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `dont_fault` library attribute.

The correct values are:

sa0	sa1	sa01

Message Details

```
Incorrect value for dont_fault '<value>' in cell '<cell>'
```

Severity

Warning

LIBWRN_15

Value of the cell attribute dont_touch should be either "true" or "false"

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `dont_touch` library attribute.

The correct values are `true` and `false`.

Message Details

```
Incorrect dont_touch '<value>' in cell '<cell>'
```

Severity

Warning

LIBWRN_16

Value of the cell attribute dont_use should be either true or false

Language

Verilog, VHDL

Rule Description

SpyGlass reports this violation if you specify an incorrect value to the `dont_use` library attribute.

The correct values are `true` and `false`.

Message Details

```
Incorrect dont_use '<value>' in cell '<cell>'
```

Severity

Warning

LIBWRN_17

Value of the cell attribute handle_negative_constraint should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the cell attribute handle_negative_constraint should be either true or false.

Message Details

```
Incorrect handle_negative_constraint '<value>' in cell '<cell>'
```

LIBWRN_18

Value of the cell attribute interface_timing should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the cell attribute interface_timing should be either true or false.

Message Details

```
Incorrect interface_timing '<value>' in cell '<cell>'
```

LIBWRN_19

Value of the cell attribute map_only should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the cell attribute map_only should be either true or false.

Message Details

```
Incorrect map_only '<value>' in cell '<cell>'
```

LIBWRN_20

Value of the cell attribute pad_cell should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the cell attribute pad_cell should be either true or false.

Message Details

```
Incorrect pad_cell '<value>' in cell '<cell>'
```

LIBWRN_21

Value of the cell attribute pad_type should be clock

Language

Verilog, VHDL

Rule Description

Value of the cell attribute pad_type should be clock.

Message Details

```
Incorrect pad_type '<value>' in cell '<cell>'
```

LIBWRN_22

Value of the cell attribute preferred should b either true or false

Language

Verilog, VHDL

Rule Description

Value of the cell attribute preferred should b either true or false.

Message Details

```
Incorrect preferred '<value>' in cell '<cell>'
```

LIBWRN_23

Value of the scan_group should be low, medium or high

Language

Verilog, VHDL

Rule Description

Value of the scan_group should be low, medium or high.

Message Details

```
Incorrect scan_group '<value>' in cell '<cell>'
```

LIBWRN_24

Value of the ff attribute clear_preset_var1 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the ff attribute clear_preset_var1 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var1 '<value>' in ff '<attribute>'
```

LIBWRN_25

Value of the ff attribute clear_preset_var2 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the ff attribute clear_preset_var2 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var2 '<value>' in ff '<attribute>'
```

LIBWRN_26

Value of the ff_bank attribute clear_preset_var1 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the ff_bank attribute clear_preset_var1 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var1 '<value>' in ff '<attribute>'
```

LIBWRN_27

Value of the ff_bank attribute clear_preset_var2 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the ff_bank attribute clear_preset_var2 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var2 '<value>' in ff_bank '<attribute>'
```

LIBWRN_28

Value of the latch attribute clear_preset_var1 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the latch attribute clear_preset_var1 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var1 '<value>' in latch '<attribute>'
```

LIBWRN_29

Value of the latch attribute clear_preset_var2 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the latch attribute clear_preset_var2 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var2 '<value>' in latch '<attribute>'
```

LIBWRN_30

Value of the latch_bank attribute clear_preset_var1 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the latch_bank attribute clear_preset_var1 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var1 '<value>' in latch_bank
'<attribute>'
```

LIBWRN_31

Value of the latch_bank attribute clear_preset_var2 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the latch_bank attribute clear_preset_var2 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var2 '<value>' in latch_bank
'<attribute>'
```

LIBWRN_32

Layer name is invalid

Language

Verilog, VHDL

Rule Description

Layer name should be the following bus_net_layer, bus_net_name_layer, bus_osc_layer, bus_osc_name_layer, bus_port_layer, bus_port_name_layer, bus_port_width_layer, bus_ripper_layer, bus_ripper_name_layer, bus_ripper_type_layer, cell_layer, cell_name_layer, cell_ref_name_layer, clock_layer, constraint_layer, hierarchy_layer, hierarchy_name_layer, highlight_layer, net_layer, net_name_layer, osc_layer, osc_name_layer, pin_layer, pin_name_layer, port_layer, port_name_layer, template_layer, template_text_layer, text_layer, variable_layer

Message Details

```
Incorrect layer name '<name>'
```

LIBWRN_33

Value of the layer attribute scalable_lines should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the layer attribute scalable_lines should be either true or false.

Message Details

```
Incorrect value for scalable_lines '<value>'
```

LIBWRN_34

Value of the layer attribute visible should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the layer attribute visible should be either true or false.

Message Details

```
Incorrect value for visible '<value>'
```

LIBWRN_35

Incorrect value of variable_1 attribute of lu_table_template group

Language

Verilog, VHDL

Rule Description

Value of the lu_table_template attribute variable_1 should be one of "input_net_transition", "total_output_net_capacitance", "output_net_length", "output_net_wire_cap", "output_net_pin_cap", "constrained_pin_transition", "related_pin_transition", "connect_delay", "output_pin_transition", "fanout_number", "driver_slew", "related_out_total_output_net_capacitance", "related_out_output_net_length", "related_out_output_net_wire_cap", "related_out_output_net_pin_cap", "related_output_pin_capacitance", "fanout_pin_capacitance", or "rc_product".

Message Details

```
Incorrect Variable '<Variable>' in lu_table_template '<value>'
```

LIBWRN_36

Incorrect value of variable_2 attribute of lu_table_template group

Language

Verilog, VHDL

Rule Description

Value of the lu_table_template attribute variable_2 should be one of "input_net_transition", "total_output_net_capacitance", "output_net_length", "output_net_wire_cap", "output_net_pin_cap", "constrained_pin_transition", "related_pin_transition", "connect_delay", "output_pin_transition", "fanout_number", "driver_slew", "related_out_total_output_net_capacitance", "related_out_output_net_length", "related_out_output_net_wire_cap", "related_out_output_net_pin_cap", "related_output_pin_capacitance", "fanout_pin_capacitance", or "rc_product".

Message Details

```
Incorrect Variable_2 '<variable>' in lu_table_template
'<value>'
```

LIBWRN_37

Incorrect value of variable_3 attribute of lu_table_template group

Language

Verilog, VHDL

Rule Description

Value of the lu_table_template attribute variable_3 should be one of "input_net_transition", "total_output_net_capacitance", "output_net_length", "output_net_wire_cap", "output_net_pin_cap", "constrained_pin_transition", "related_pin_transition", "connect_delay", "output_pin_transition", "fanout_number", "driver_slew", "related_out_total_output_net_capacitance", "related_out_output_net_length", "related_out_output_net_wire_cap", "related_out_output_net_pin_cap", "related_output_pin_capacitance", "fanout_pin_capacitance", or "rc_product".

Message Details

```
Incorrect Variable_3 '<Variable>' in lu_table_template
'<value>'
```

LIBWRN_38

Value of the memory group attribute type should be either RAM or ROM

Language

Verilog, VHDL

Rule Description

Value of the memory group attribute type should be either ram or rom.

Message Details

```
Incorrect value for Type '<value>' in memory group
```

LIBWRN_39

Value of the attribute tree_type should be "best_case_tree", "worst_case_tree", or "balanced_tree"

Language

Verilog, VHDL

Rule Description

Value of the attribute tree_type should be "best_case_tree", "worst_case_tree", or "balanced_tree".

Message Details

```
Incorrect value for TreeType '<value>' in operating_condition
group
```

LIBWRN_40

Value of the clock attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the clock attribute should be either true or false.

Message Details

```
Incorrect value for clock(b/b/p) '<value>'
```

LIBWRN_41

Value of the direction attribute should be inout, input, internal, output, or tristate

Language

Verilog, VHDL

Rule Description

Value of the direction attribute should be inout, input, internal, output, or tristate

Message Details

```
Incorrect direction (b/b/p) '<direction>'
```

LIBWRN_42

Value of the dont_fault attribute should be "sa0", "sa1", or "sa01"

Language

Verilog, VHDL

Rule Description

Value of the dont_fault attribute should be "sa0", "sa1", or "sa01"

Message Details

```
Incorrect value for dont_fault (b/b/p) '<value>'
```

LIBWRN_43

Value of the hysteresis attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the hysteresis attribute should be either true or false.

Message Details

```
Incorrect value for hysteresis (b/b/p) '<value>'
```

LIBWRN_44

Value of the inverted_output attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the inverted_output attribute should be either true or false.

Message Details

```
Incorrect value for inverted_output (b/b/p) '<value>'
```

LIBWRN_45

Value of the is_pad attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the is_pad attribute should be either true or false.

Message Details

```
Incorrect value for is_pad (b/b/p) '<value>'
```

LIBWRN_46

Value of the multi_cell_pad_pin attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the multi_cell_pad_pin attribute should be either true or false.

Message Details

```
Incorrect value for multi_cell_pad_pin (b/b/p) '<value>'
```

LIBWRN_47

Value of the next_state attribute should be data, load, clear, preset, scan_in, scan_enable

Language

Verilog, VHDL

Rule Description

Value of the next_state attribute should be data, load, clear, preset, scan_in, scan_enable.

Message Details

```
Incorrect next_state (b/b/p) '<value>'
```

LIBWRN_48

Value of the pin_func_type attribute should be active_rising, active_falling, clock_enable, active_high, or active_low

Language

Verilog, VHDL

Rule Description

Value of the pin_func_type attribute should be active_rising, active_falling, clock_enable, active_high, or active_low.

Message Details

```
Incorrect pin_func_type (b/b/p) '<value>'
```

LIBWRN_49

Value of the prefer_tied attribute should be either 0 or 1

Language

Verilog, VHDL

Rule Description

Value of the prefer_tied attribute should be either 0 or 1.

Message Details

```
Incorrect value for prefer_tied (b/b/p) '<value>'
```

LIBWRN_50

Value of the primary_output attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the primary_output attribute should be either true or false.

Message Details

```
Incorrect value for primary_output (b/b/p) '<value>'
```

LIBWRN_51

Incorrect value of the signal_type attribute

Language

Verilog, VHDL

Rule Description

Value of the signal_type attribute should be "test_scan_enable", "test_scan_enable_inverted", "test_scan_clock", "test_scan_clock_a", "test_scan_clock_b", "test_clock", "test_scan_in", "test_scan_in_inverted", "test_scan_out", "test_scan_out_inverted", or "clocked_on_also".

Message Details

```
Incorrect signal_type (b/b/p) '<value>'
```

LIBWRN_52

Value of the slew_control attribute should be one of none, high, low, or medium

Language

Verilog, VHDL

Rule Description

Value of the slew_control attribute should be one of none, high, low, or medium.

Message Details

```
Incorrect slew_control (b/b/p) '<value>'
```

LIBWRN_53

Value of the test_output_only attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the test_output_only attribute should be either true or false.

Message Details

```
Incorrect value for test_output_only (b/b/p) '<value>'
```

LIBWRN_54

Value of the landscape attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the landscape attribute should be either true or false.

Message Details

```
Incorrect value for landscape '<value>' in symbol '<symbol>'
```

LIBWRN_55

Value of the ApproachDirection of the Symbol group should be UP, DOWN, LEFT, or RIGHT

Language

Verilog, VHDL

Rule Description

Value of the ApproachDirection of the Symbol group should be UP, DOWN, LEFT, or RIGHT.

Message Details

```
Incorrect Approach Direction '<direction>' in symbol pin
'<pin>'
```

LIBWRN_56

Invalid layer name for the text attribute of symbol group

Language

Verilog, VHDL

Rule Description

Layer name of the text attribute should be bus_net_layer, bus_net_name_layer, bus_osc_layer, bus_osc_name_layer, bus_port_layer, bus_port_name_layer, bus_port_width_layer, bus_ripper_layer, bus_ripper_name_layer, bus_ripper_type_layer, cell_layer, cell_name_layer, cell_ref_name_layer, clock_layer, constraint_layer, hierarchy_layer, hierarchy_name_layer, highlight_layer, net_layer, net_name_layer, osc_layer, osc_name_layer, pin_layer, pin_name_layer, port_layer, port_name_layer, template_layer, template_text_layer, text_layer, variable_layer

Message Details

```
Incorrect layer name (Text) '<name>' in symbol text '<symbol>'
```

LIBWRN_57

Invalid layer name of the variable attribute of the symbol group

Language

Verilog, VHDL

Rule Description

Layer name of the variable attribute should be bus_net_layer, bus_net_name_layer, bus_osc_layer, bus_osc_name_layer, bus_port_layer, bus_port_name_layer, bus_port_width_layer, bus_ripper_layer, bus_ripper_name_layer, bus_ripper_type_layer, cell_layer, cell_name_layer, cell_ref_name_layer, clock_layer, constraint_layer, hierarchy_layer, hierarchy_name_layer, highlight_layer, net_layer, net_name_layer, osc_layer, osc_name_layer, pin_layer, pin_name_layer, port_layer, port_name_layer, template_layer, template_text_layer, text_layer, variable_layer

Message Details

```
Incorrect layer name (Variable) '<name>' in symbol variable
'<symbol>'
```

LIBWRN_58

Value of the related_output_pin of the timing group should be related_out_total_output_net_capacitance, related_out_output_net_length, related_out_output_net_wire_cap, or related_out_output_net_pin_cap

Language

Verilog, VHDL

Rule Description

Value of the related_output_pin of the timing group should be related_out_total_output_net_capacitance, related_out_output_net_length, related_out_output_net_wire_cap, or related_out_output_net_pin_cap.

Message Details

```
Incorrect related_output_pin '<value>' in timing group
```

LIBWRN_59

Invalid value for the timing_type attribute of the timing group

Language

Verilog, VHDL

Rule Description

Value of the timing_type attribute should be three_state_disable, retaining_time, non_seq_setup_falling, skew_rising, nochange_low_low, nochange_high_low, skew_falling, combinational_fall, nochange_low_high, nochange_high_high, combinational_rise, rising_edge, three_state_enable, preset, setup_rising, three_state_disable_fall, hold_falling, non_seq_hold_rising, minimum_period, removal_falling, non_seq_setup_rising, three_state_enable_fall, clear, non_seq_hold_falling, three_state_disable_rise, recovery_rising, three_state_enable_rise, setup_falling, falling_edge, min_pulse_width, combinational, recovery_falling, removal_rising, or hold_rising

Message Details

```
Incorrect timing_type value '<value>' in timing group
```

LIBWRN_60

Value of the timing_sense attribute of the timing group should be non_unate, negative_unate, or positive_unate

Language

Verilog, VHDL

Rule Description

Value of the timing_sense attribute of the timing group should be non_unate, negative_unate, or positive_unate.

Message Details

```
Incorrect timing_sense value '<value>' in timing group
```

LIBWRN_61

Value of the base_type attribute of the type group should be array

Language

Verilog, VHDL

Rule Description

Value of the base_type attribute of the type group should be array.

Message Details

```
Incorrect value '<value>' for base_type in type group
```

LIBWRN_62

Value of the downto attribute of the type group should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the downto attribute of the type group should be either true or false.

Message Details

```
Incorrect value for downto '<value>' in type group
```

LIBWRN_63

Value of the driver_type attribute should be pull_up, pull_down, open_drain, open_source, bus_hold, resistive, resistive_0, or resistive_1

Language

Verilog, VHDL

Rule Description

Value of the driver_type attribute should be pull_up, pull_down, open_drain, open_source, bus_hold, resistive, resistive_0, or resistive_1.

Message Details

```
Incorrect value for driver_type(b/b/p) '<value>'
```

LIBWRN_64

Value of the driver_type attribute should be pull_up, pull_down, open_drain, open_source, bus_hold, resistive, resistive_0, or resistive_1

Language

Verilog, VHDL

Rule Description

Value of the driver_type attribute should be pull_up, pull_down, open_drain, open_source, bus_hold, resistive, resistive_0, or resistive_1.

Message Details

```
Incorrect value for driver_type(b/b/p) '<value>'
```

LIBWRN_65

Value of the driver_type attribute should be pull_up, pull_down, open_drain, open_source, bus_hold, resistive, resistive_0, or resistive_1

Language

Verilog, VHDL

Rule Description

Value of the driver_type attribute should be pull_up, pull_down, open_drain, open_source, bus_hold, resistive, resistive_0, or resistive_1.

Message Details

```
Incorrect value for driver_type(b/b/p) '<value>'
```

LIBWRN_66

The divisor should not be zero

Language

Verilog, VHDL

Rule Description

The divisor should not be zero

Message Details

```
Divide by 0
```

LIBWRN_67

The input library file should be of correct format

Language

Verilog, VHDL

Rule Description

The input library file should be of correct format.

Message Details

`Undefined reference to variable '<value>'`

LIBWRN_68

Invalid value for the clock_gating_integrated_cell attribute

Language

Verilog, VHDL

Rule Description

Value of the clock_gating_integrated_cell should be latch_posedge, latch_posedge_precontrol, latch_posedge_postcontrol, latch_posedge_precontrol_obs, latch_posedge_postcontrol_obs, latch_negedge, latch_negedge_precontrol, latch_negedge_postcontrol, latch_negedge_precontrol_obs, latch_negedge_postcontrol_obs, none_posedge, none_posedge_control, none_posedge_control_obs, none_negedge, none_negedge_control, none_negedge_control_obs, ff_posedge, ff_posedge_precontrol, ff_posedge_postcontrol, ff_posedge_precontrol_obs, ff_posedge_postcontrol_obs, ff_negedge, ff_negedge_precontrol, ff_negedge_postcontrol, ff_negedge_precontrol_obs, ff_negedge_postcontrol_obs.

Message Details

```
Incorrect value for clock_gating_integrated_cell '<value>'
```

LIBWRN_69

Incorrect value for the expression

Language

Verilog, VHDL

Rule Description

Incorrect value for the expression

Message Details

```
Incorrect value '<value>'
```

LIBWRN_70

Value of input_threshold_pct_fall, input_threshold_pct_rise, output_threshold_pct_fall, output_threshold_pct_rise, slew_lower_threshold_pct_fall, slew_lower_threshold_pct_rise, slew_upper_threshold_pct_fall, and slew_upper_threshold_pct_rise should lie between 0.0 and 100.0

Language

Verilog, VHDL

Rule Description

Value of input_threshold_pct_fall, input_threshold_pct_rise, output_threshold_pct_fall, output_threshold_pct_rise, slew_lower_threshold_pct_fall, slew_lower_threshold_pct_rise, slew_upper_threshold_pct_fall, and slew_upper_threshold_pct_rise should lie between 0.0 and 100.0

Message Details

```
threshold pct values must be in the range of 20.0 to 80.0
```

LIBWRN_73

The bus_naming_style has invalid syntax. Hence default style will be used

Language

Verilog, VHDL

Rule Description

The bus_naming_style can contain alphanumeric characters, brackets, underscores, dashes, or parentheses. It must contain one %s symbol and one %d symbol. The %s and %d symbols can appear in any order, but at least one non-numeric character must separate them.

Message Details

```
Incorrect BusNamingStyle <bus-naming-format>, keeping default
style '%%s[%%d]'
```

LIBWRN_74

This construct is not inferred by SpyGlass

Language

Verilog, VHDL

Rule Description

This construct is not inferred by SpyGlass

Message Details

```
Construct '<construct>' found but ignored
```

LIBWRN_75

Value of the variable_1 attribute of em_lut_template group should be "input_transition_time", "total_output_net_capacitance", or "equal_or_opposite_output_net_capacitance"

Language

Verilog, VHDL

Rule Description

Value of the variable_1 attribute of em_lut_template group should be "input_transition_time", "total_output_net_capacitance", "equal_or_opposite_output_net_capacitance"

Message Details

```
Incorrect variable_1 '<value>' for em_lut_template
```

LIBWRN_76

Value of the variable_2 attribute of em_lut_template group should be "input_transition_time", "total_output_net_capacitance", or "equal_or_opposite_output_net_capacitance"

Language

Verilog, VHDL

Rule Description

Value of the variable_2 attribute of em_lut_template group should be "input_transition_time", "total_output_net_capacitance", or "equal_or_opposite_output_net_capacitance"

Message Details

```
Incorrect variable_2 '<value>' for em_lut_template
```

LIBWRN_77

Value of the technology attribute should be either cmos or FPGA

Language

Verilog, VHDL

Rule Description

Value of the technology attribute should be either cmos or FPGA.

Message Details

```
Incorrect technology (cmos/FPGA) '<value>', keeping default
technology cmos
```

LIBWRN_78

Unknown simple or complex attribute found

Language

Verilog, VHDL

Rule Description

Check the spelling of the attribute on which violation is reported. If the attribute is user defined, then it should be previously registered using 'define' statement.

Message Details

```
Ignoring Unknown Attribute '<attribute>' in group '<group>'
```

LIBWRN_79

Value used for variables attribute of domain group is invalid

Language

Verilog, VHDL

Rule Description

Value for variables attribute of the domain group should be a list ofinput_net_transition, constrained_pin_transition, total_output_net_capacitance, output_net_length, output_net_wire_cap, output_net_pin_cap, related_pin_transition, related_out_total_output_net_capacitance, related_out_output_net_length, related_out_output_net_wire_cap, related_out_output_net_pin_cap, temperature, voltage, or voltage1.

Message Details

```
Invalid variable name '<name>'
```

LIBWRN_80

Value for mapping attribute of domain group should be either voltage or voltage1

Language

Verilog, VHDL

Rule Description

Value for mapping attribute of domain group should be either voltage or voltage1.

Message Details

```
Invalid value '<value>' for mapping
```

LIBWRN_81

In attribute variable_n_range, n can be 1, 2, 3, 4, 5, or 6

Language

Verilog, VHDL

Rule Description

In attribute variable_n_range, n can be 1, 2, 3, 4, 5, or 6.

Message Details

`Invalid index <index> for variable_n_range`

LIBWRN_82

coefs and orders are the only valid attributes for the domain group declared inside a timing group

Language

Verilog, VHDL

Rule Description

coefs and orders are the only valid attributes for the domain group declared inside a timing group.

Message Details

`Invalid attribute <attribute> for domain group defined in timing group`

LIBWRN_83

calc_mode, variables, mapping and variable_n_range are valid attributes for the domain group defined inside a poly_template group

Language

Verilog, VHDL

Rule Description

calc_mode, variables, mapping and variable_n_range are valid attributes for the domain group defined inside a poly_template group.

Message Details

```
Invalid attribute <attribute> for domain group defined in
poly_template group
```

LIBWRN_84

Index of the pin defined inside a bus group should be lie between the values of "to" and "from" attributes of the type group associated with the bus

Language

Verilog, VHDL

Rule Description

Index of the pin defined inside a bus group should be lie between the values of "to" and "from" attributes of the type group associated with the bus.

Message Details

```
Invalid index or index range for bused pin <pin>
```

LIBWRN_85

The pin name of the pin defined inside a bus group should conform to the bus_naming_style of that bus group

Language

Verilog, VHDL

Rule Description

The pin name of the pin defined inside a bus group should conform to the bus_naming_style of that bus group.

Message Details

`Invalid pin name <name> for bus <bus>`

LIBWRN_86

Value of edge_type attribute of the tlatch group should be either rising or falling

Language

Verilog, VHDL

Rule Description

Value of edge_type attribute of the tlatch group should be either rising or falling.

Message Details

```
Invalid edge_type '<value>' for tlatch group
```

LIBWRN_87

Value of the tdisable attribute of the tlatch group should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the tdisable attribute of the tlatch group should be either true or false.

Message Details

```
Invalid value '<value>' for tdisable for tlatch group. It
should be true or false.
```

LIBWRN_88

Value of lsi_pad attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of lsi_pad attribute should be either true or false.

Message Details

```
Incorrect value for lsi_pad (b/b/p) '<value>'
```

LIBWRN_89

Value of the dcm_timing attribute should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the dcm_timing attribute should be either true or false.

Message Details

```
Incorrect value for dcm_timing (b/b/p) '<value>'
```

LIBWRN_90

Value of the timing_report attribute of library group should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the timing_report attribute of library group should be either true or false.

Message Details

```
Incorrect value for timing_report '<value>' in library
'<library>'
```

LIBWRN_91

Value of the use_for_size_only attribute of cell group should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the use_for_size_only attribute of cell group should be either true or false.

Message Details

```
Incorrect value for use_for_size_only '<value>' in cell
'<cell>'
```

LIBWRN_92

Value of the is_clock_gating_cell attribute of the cell group should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the is_clock_gating_cell attribute of the cell group should be either true or false.

Message Details

```
Incorrect value for is_clock_gating_cell '<value>' in cell
'<cell>'
```

LIBWRN_93

Value of the default_timing attribute of the timing group should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the default_timing attribute of the timing group should be either true or false.

Message Details

```
Incorrect value for default_timing '<value>' in timing group
```

LIBWRN_94

Pin is not contained in the cell group.

Language

Verilog, VHDL

Message Details

<pin> Pin not found in Cell <cell>

LIBWRN_95

Pin specified in Boolean expression is missing from syntax

Language

Verilog, VHDL

Rule Description

It is an error if, e.g. 'function: "A * B"' is specified for an output pin, but pin A is not found in the parent cell group.

Message Details

```
Invalid Boolean expression for output function of pin <pin>.
Ignoring cell <cell>
```

LIBWRN_96

The input pin name in the function attribute is not valid.

Language

Verilog, VHDL

Message Details

```
Invalid Input Pin <pin> in function
```

LIBWRN_97

The pin name should not be nil.

Language

Verilog, VHDL

Message Details

`Pin name Null in cell <cell>`

LIBWRN_98

Pin name in a bus group is invalid

Language

Verilog, VHDL

Rule Description

If a pin does not belong to a bus but is being specified in the bus group, it is an error as in the following example:

```
bus(A){
...
  pin(B){
...
}
```

Message Details

`Invalid Pin <pin> in Bus <bus>. Ignoring cell <cell>`

LIBWRN_99

Cell name should not be nil.

Language

Verilog, VHDL

Message Details

```
No Cell Name Specified, ignoring cell in lib <library>
```

LIBWRN_100

Bus naming style should contain exactly one '%s' and exactly one %d

Language

Verilog, VHDL

Rule Description

Bus naming style should contain exactly one '%s' and exactly one %d

Message Details

`Invalid bus naming style <style>. Ignoring the cell <cell-name>`

LIBWRN_101

Value of the seq attribute clear_preset_var1 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the seq attribute clear_preset_var1 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var1 '<value>' in seq '<attribute>'
```

LIBWRN_102

Value of the seq attribute clear_preset_var2 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the seq attribute clear_preset_var2 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var2 '<value>' in seq '<attribute>'
```

LIBWRN_103

Value of the seq_bank attribute clear_preset_var1 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the seq_bank attribute clear_preset_var1 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var1 '<value>' in seq_bank '<attribute>'
```

LIBWRN_104

Value of the seq_bank attribute clear_preset_var4 should be H, L, N, T, U, or X

Language

Verilog, VHDL

Rule Description

Value of the seq_bank attribute clear_preset_var4 should be H, L, N, T, U, or X.

Message Details

```
Incorrect clear_preset_var4 '<value>' in seq_bank '<attribute>'
```

LIBWRN_105

The address width and the word width of the memory should match in the memory read and memory write group with the corresponding width in the bus group

Language

Verilog, VHDL

Rule Description

The address width and the word width of the memory should match in the memory read and memory write group with the corresponding width in the bus group.

Message Details

```
The address width and the word width should match in the memory
read and memory write group with the corresponding width in the
bus group '<group>' for Cell '<cell>'
```

LIBWRN_106

The address width and the word width should match in the memory read and memory write group with the corresponding width in the bundle group

Language

Verilog, VHDL

Rule Description

The address width and the word width should match in the memory read and memory write group with the corresponding width in the bundle group.

Message Details

```
The address width and the word width must match in the memory
read and memory write group with the corresponding width in
bundle group '<group>' for Cell '<cell>'
```

LIBWRN_107

The address width and the word width should match in the memory read and memory write group with the corresponding width in the pin group

Language

Verilog, VHDL

Rule Description

The address width and the word width should match in the memory read and memory write group with the corresponding width in the pin group

Message Details

```
The address width and the word width must match in the memory
read and memory write group with the corresponding width in pin
group '<group>' for Cell '<cell>'
```

LIBWRN_108

Function attribute for a pin is being used from test_cell block specified in a cell

Language

Verilog, VHDL

Rule Description

If function attribute is not specified for some output pin in its pin block but it is specified in test_cell, then function specified in test_cell is used.

Message Details

`Using function attribute from test_cell block for pin <pin> in Cell <cell>`

LIBWRN_109

Value of the cell attribute is_isolation_cell should be either true or false

Language

Verilog, VHDL

Rule Description

Value of the cell attribute is_isolation_cell should be either true or false.

Message Details

```
Incorrect is_isolation_cell '<value>' in cell '<cell>'
```

LIBWRN_110

Internal Pin found in a cell

Language

Verilog, VHDL

Rule Description

Internal Pin found in a cell

Message Details

```
Cell '<cell>' description contains internal pin '<pin>',
translation mayn't be correct
```

LIBWRN_111

An attribute/group is found which is not valid in the current group.

Language

Verilog, VHDL

Rule Description

An attribute/group is found which is not valid in the current group.

Message Details

```
Construct '<construct>' is not supported in the scope of
<group> group
```

LIBWRN_112

The pin directions of the individual pins in a bus should be consistent.

Language

Verilog, VHDL

Rule Description

The pin directions of the individual pins in a bus should be consistent.

Message Details

```
The pin directions in bus '<bus> in cell '<cell>' conflict with
each other. Using the first specified pin direction as the bus
parent direction
```

LIBWRN_113

Value of the level_shifter_enable_pin should be either true or false

Language

Verilog, VHDL

Rule Description

Value of `level_shifter_enable_pin` cell attribute should be either true or false.

Message Details

```
Incorrect value of level_shifter_enable_pin '<value>'
```

LIBWRN_114

Value of isolation_cell_enable_pin should be either true or false

Language

Verilog, VHDL

Rule Description

Value of `isolation_cell_enable_pin` cell attribute should be either true or false.

Message Details

```
Incorrect value of isolation_cell_enable_pin '<value>'
```

LIBWRN_115

Value of is_level_shifter should be either true or false

Language

Verilog, VHDL

Rule Description

Value of `is_level_shifter` cell attribute should be either true or false

Message Details

```
Incorrect value of is_level_shifter '<value>' in cell '<cell>'
```

LIBWRN_116

Value of the switch_pin should be either true or false.

Language

Verilog, VHDL

Rule Description

Value of `switch_pin` should be either true or false.

Message Details

```
Incorrect value of switch_pin '<value>' in pin group
```

LIBWRN_117

Value of the cell attribute switch_cell_type should be fine_grain or coarse_grain

Language

Verilog, VHDL

Rule Description

Value of the `switch_cell_type` cell attribute should be `fine_grain` or `coarse_grain`.

Message Details

```
Incorrect switch_cell_type '<value>' in cell '<cell>'
```

LIBWRN_118

Value of the cell attribute level_shifter_type should be LH, HL or HL_LH

Language

Verilog, VHDL

Rule Description

Value of the `level_shifter_type` cell attribute should be `LH`, `HL`, or `HL_LH`

Message Details

```
Incorrect level_shifter_type '<value>' in cell '<cell>'
```

LIBWRN_119

Value of the pg pin attribute pg_type should be primary_power, internal_power, primary_ground, backup_power, or backup_ground.

Language

Verilog, VHDL

Rule Description

Value of the pg pin attribute, `pg_type`, should be `primary_power`, `internal_power`, `primary_ground`, `backup_power`, or `backup_ground`.

Message Details

```
Incorrect pg_type '<value>' in pg_pin group
```

LIBWRN_120

Value of the std_cell_main_rail should be either true or false.

Language

Verilog, VHDL

Rule Description

Value of `std_cell_main_rail` should be either true or false.

Message Details

```
Incorrect value of std_cell_main_rail '<value>' in pg_pin group
```

LIBWRN_121

Value of the first argument of retention_pin attribute should be save, restore, save_restore or restore_save.

Language

Verilog, VHDL

Rule Description

Value of the first argument of the `retention_pin` attribute should be `save`, `restore`, `save_restore`, or `restore_save`.

Message Details

```
Incorrect value specified in retention_pin attribute '<value>'
in pin group
```

LIBWRN_122

Value of the second argument of retention_pin attribute should be either 0 or 1.

Language

Verilog, VHDL

Rule Description

Value of the second argument of the `retention_pin` attribute should be either 0 or 1.

Message Details

```
Incorrect value specified in 'retention_pin' attribute
'<value>' in pin group. Its valid values are literal 1 or 0
```

LIBWRN_123

Value of attribute clock_gating_flag should be either true or false.

Language

Verilog, VHDL

Rule Description

Value of the `clock_gating_flag` attribute should be either true or false.

Message Details

```
Incorrect value of clock_gating_flag '<value>' in timing group
```

LIBWRN_124

Voltage Name given in pg_pin group should be present in the voltage_map list.

Language

Verilog, VHDL

Rule Description

Voltage Name given in pg_pin group should be present in the voltage_map list.

Message Details

```
Voltage Name '<name>' specified in pg_pin group '<group>' does
not match with any voltage name in voltage_map list
```

LIBWRN_125

One primary power pin and one primary ground pin must be present in a cell with pg_pins

Language

Verilog, VHDL

Rule Description

One primary power pin and one primary ground pin must be present in a cell with pg_pins.

Message Details

```
Primary power pin and/or primary ground Pin missing in cell
'<cell>'. Default related power/ground pin will not be set
```

LIBWRN_126

Multiple ff/ff_bank/latch/latch_bank group specifications is not allowed in a library cell

Language

Verilog, VHDL

Rule Description

This rule reports a violation if multiple `ff`/`ff_bank`/`latch`/`latch_bank` group specifications are specified for a library cell.

Message Details

```
Multiple ff/ff_bank/latch/latch_bank group specifications found
in cell '<cell>' of library '<library>'. Functionality of this
cell has not been inferred. Only cell interface has been
interpreted
```

LIBWRN_127

Value of the level_shifter_data_pin should be either "true" or "false"

Language

Verilog, VHDL

Rule Description

This rule reports a violation if the value of the `level_shifter_data_pin` attribute is other than `true` or `false`.

Message Details

```
Value of the level_shifter_data_pin should be either "true" or
"false"
```

LIBWRN_128

Value of the isolation_cell_data_pin should be either "true" or "false"

Language

Verilog, VHDL

Rule Description

This rule reports a violation if the value of the `isolation_cell_data_pin` attribute is other than `true` or `false`.

Message Details

```
Value of the isolation_cell_data_pin should be either "true" or
"false"
```

LIBERROR_301

A memory_write bus should not contain both 'clocked_on' and 'enable' attributes

Language

Verilog, VHDL

Rule Description

A memory_write bus should not contain both 'clocked_on' and 'enable' attributes

Message Details

```
Write bus '<bus>' is defined to be both synchronous and
asynchronous. Ignoring cell <cell>
```

LIBERROR_302

Type group of bus cannot be located in the library

Language

Verilog, VHDL

Rule Description

Type group of bus cannot be located in the library

Message Details

`Bus Type <type> not found for Bus <bus>. Ignoring Cell <cell>`

LIBERROR_303

For a flip-flop with both clear and preset, clear_preset value is mandatory

Language

Verilog, VHDL

Rule Description

For a flip-flop with both clear and preset, clear_preset value is mandatory.

Message Details

```
clear_preset value for internal variables not defined for FF
Cell <Cell> of lib <library>. Ignoring Cell
```

LIBERROR_304

If clear_preset is defined then flip-flop should have both clear and preset

Language

Verilog, VHDL

Rule Description

If clear_preset is defined then flip-flop should have both clear and preset

Message Details

```
clear_preset values for internal variables defined, but Clear/
Preset Pin not defined for FF Cell <cell> of lib <library>.
Ignoring Cell
```

LIBERROR_305

For a flip-flop bank with both clear and preset, clear_preset value is mandatory

Language

Verilog, VHDL

Rule Description

For a flip-flop bank with both clear and preset, clear_preset value is mandatory

Message Details

```
clear_preset value for internal variables not defined for
FFBank Cell <cell> of lib <library>. Ignoring Cell
```

LIBERROR_306

If clear_preset is defined then flip-flop bank should have both clear and preset

Language

Verilog, VHDL

Rule Description

If clear_preset is defined then flip-flop bank should have both clear and preset

Message Details

```
clear_preset values for internal variables defined, but Clear/
Preset Pin not defined for FFBank Cell <cell> of lib <library>.
Ignoring Cell
```

LIBERROR_307

For a latch with both clear and preset, clear_preset value is mandatory

Language

Verilog, VHDL

Rule Description

For a latch with both clear and preset, clear_preset value is mandatory

Message Details

```
clear_preset value for internal variables not defined for Latch
Cell <cell> of lib <library>. Ignoring Cell
```

LIBERROR_308

If clear_preset is defined then latch should have both clear and preset

Language

Verilog, VHDL

Rule Description

If clear_preset is defined then latch should have both clear and preset

Message Details

```
clear_preset values for internal variables defined, but Clear/
Preset Pin not defined for Latch Cell <cell> of lib <library>.
Ignoring Cell
```

LIBERROR_309

For a latch bank with both clear and preset, clear_preset value is mandatory

Language

Verilog, VHDL

Rule Description

For a latch bank with both clear and preset, clear_preset value is mandatory

Message Details

`clear_preset value for internal variables not defined for LatchBank Cell <cell> of lib <library>. Ignoring Cell`

LIBERROR_310

If clear_preset is defined then latch bank should have both clear and preset

Language

Verilog, VHDL

Rule Description

If clear_preset is defined then latch bank should have both clear and preset

Message Details

```
clear_preset values for internal variables defined, but Clear/
Preset Pin not defined for LatchBank Cell <cell> of lib
<library>. Ignoring Cell
```

LIBERROR_311

downto attribute should be true if 'bit_from' is greater than 'bit_to'

Language

Verilog, VHDL

Rule Description

If value of 'bit_from' is greater than that of 'bit_to', then set 'downto' attribute as true

Message Details

```
Incorrect value of downto attribute
```

LIBERROR_312

Value of 'bit_width' should be coherent with the values of 'bit_from' and 'bit_to'

Language

Verilog, VHDL

Rule Description

bit_width = |bit_from - bit_to| + 1

Please check if correct bit_width has been specified.

Message Details

```
Incorrect value '<value>' of bit_width attribute. It should be
'<correct-value>'
```

LIBERROR_313

Repeated libraries are ignored

Language

Verilog, VHDL

Rule Description

Repeated libraries are ignored

Message Details

`Ignoring repeated library <library>`

LIBSTX_401

The input file is unreadable

Language

Verilog, VHDL

Rule Description

The input file is found, but it is unreadable. Check the read permission of the file.

Message Details

```
Error reading input file '<file>'
```

LIBSTX_402

The input file cannot be opened for reading

Language

Verilog, VHDL

Rule Description

Check if the input library file exists at the mentioned path and read permission has been provided.

Message Details

```
Could not open input file '<file>' (Reason <reason>)
```

LIBSTX_403

Unexpected token encountered

Language

Verilog, VHDL

Rule Description

Unexpected token encountered

Message Details

```
Parse error on or before token '<token>'
```

LIBSTX_404

Another call for include file is found inside an include file

Language

Verilog, VHDL

Rule Description

Recursive call for include file is not support. Remove the call for include file inside the included file.

Message Details

`Recursive call for include file`

LIBSTX_405

The poly_template group allows a maximum of six variables

Language

Verilog, VHDL

Rule Description

Make the number of variables defined in poly_template group less than or equal to six.

Message Details

```
A maximum of six variables is allowed in '<poly_template>'
attribute of '<group>' group
```

LIBSTX_406

The domain group attribute variable_n_range has syntax error

Language

Verilog, VHDL

Rule Description

'n' in the variable_n_range attribute should be an integer lying between 1 and 6.

Message Details

```
Invalid attribute '<attribute>' for domain group
```

LIBSTX_409

Value of the defined attribute should be of type string, float, integer or Boolean

Language

Verilog, VHDL

Rule Description

Value of the defined attribute should be of type string, float, integer or Boolean.

Message Details

```
Invalid type '<type>' for defined Attribute. Should be string,
float, integer or Boolean
```

LIBSTX_410

A new user defined attribute should be used only in the group in which it is declared

Language

Verilog, VHDL

Rule Description

A new user defined attribute should be used only in the group in which it is declared.

Message Details

```
Attribute declaration is in the group '<group>' and not in the
desired group '<desired-group>'
```

LIBSTX_411

The value of the user-defined attribute is not of the same type as stated in the declaration of the attribute

Language

Verilog, VHDL

Rule Description

The value of the user-defined attribute is not of the same type as stated in the declaration of the attribute.

Message Details

```
Attribute type '<type>' does not tally with the defined one
```

LIBSTX_412

Incorrect value for the expression

Language

Verilog, VHDL

Rule Description

Incorrect value for the expression.

Message Details

```
Incorrect value '<value>'
```

LIBSTX_413

The defined attribute was defined for the different group, and used in a different group

Language

Verilog, VHDL

Rule Description

The defined attribute was defined for the different group, and used in a different group.

Message Details

```
The '<defined-attribute>' attribute was defined to be used in
the '<group1>' group, not the '<group2>' group
```

LIBSTX_414

The define_group attribute was defined for the different group, and used in a different group

Language

Verilog, VHDL

Rule Description

The define_group attribute was defined for the different group, and used in a different group

Message Details

```
The '<group1>' group was not defined to be used in the
'<group2>' group
```

LIBSTX_415

Constructs used in IGNORE_LIB_CONSTRUCT must be of the form [group::]attribute where group:: is optional.

Language

Verilog, VHDL

Rule Description

Constructs used in IGNORE_LIB_CONSTRUCT must be of the form [group::]attribute where group:: is optional.

Message Details

```
<construct> is not used according to ignore
IGNORE_LIB_CONSTRUCT format,[group::]attribute
```

LIBINFO_701

Repeated cells are ignored (is controlled by an internal flag)

Language

Verilog, VHDL

Rule Description

Repeated cells are ignored (is controlled by an internal flag)

Message Details

```
Ignoring repeated cell <cell> in library <library>
```

LIBINFO_702

Repeated cells are replaced (is controlled by an internal flag)

Language

Verilog, VHDL

Rule Description

Repeated cells are replaced (is controlled by an internal flag)

Message Details

```
Replacing repeated cell <cell> in library <library>
```

LIBINFO_704

In case of missing voltage names in pg_pin groups first voltage name in voltage_map name, value pair is used

Language

Verilog, VHDL

Rule Description

In case of missing voltage names in pg_pin groups, first voltage name in voltage_map name-value pair is used.

Message Details

```
Missing voltage name in pg_pin '<group>'. Using '<name>' (first
voltage_map voltage name)
```

LIBINFO_705

In case of missing related power/ground pins in pin groups, first primary power/ground pg_pin is used.

Language

Verilog, VHDL

Rule Description

In case of missing related power/ground pins in pin groups, first primary power/ground pg_pin is used.

Message Details

```
Missing related_power/ground_pin in pin '<pin>'. Using default
'<value>'(first primary power/ground pg_pin)
```

LEF Parsing Messages

Overview

The rules of this category report violations based on the parsing done on LEF files (.lef files).

LEFSTX_1

Unexpected token encountered

Language

Verilog, VHDL

Rule Description

Unexpected token encountered

Message Details

```
Parse error at <location>
```

Severity

Syntax

LEFSTX_2

Unexpected Syntax encountered

Language

Verilog, VHDL

Rule Description

Unexpected syntax encountered

Message Details

<message>

Severity

Syntax

LEFSTX_3

Pin directions of bus pins of a parent bus are different

Language

Verilog, VHDL

Rule Description

SpyGlass reports this syntax error when bus pins of the same parent bus are of different directions.

Pin directions of all the bus pins of a parent bus must be same.

Message Details

```
Pin directions of bus pins of the parent bus <bus-name> are
different
```

Severity

Syntax

LEFSTX_4

Pin types of bus pins of a parent bus are different

Language

Verilog, VHDL

Rule Description

SpyGlass reports this syntax error when bus pins of the same parent bus have different pin types.

Pin types of all bus pins of a parent bus must be same.

Message Details

```
Pin types of bus pins of the parent bus <bus-name> are
different
```

Severity

Syntax

LEFWRN_1

Duplicate construct

Language

Verilog, VHDL

Rule Description

This construct is repeated in LEF file

Message Details

```
Ignoring repeated construct <construct> in library '<library>'
```

Severity

Warning

LEFWRN_2

Pin_type of pin is missing

Language

Verilog, VHDL

Rule Description

`Pin_type` is mandatory feature of pin to get power ground information

Message Details

```
Pin_type of pin '<pin>' of macro '<macro>' is missing, assuming
it to be 'signal'
```

Severity

Warning

LEFWRN_3

Pin direction of pin is missing

Language

Verilog, VHDL

Rule Description

Pin direction is mandatory feature of pin to get electrical direction of pin

Message Details

```
Pin direction of pin '<pin>' of macro '<macro>' is missing,
assuming it to be 'input'
```

Severity

Warning

LEFWRN_4

Macro has no pin information

Language

Verilog, VHDL

Rule Description

There are no pins in the macro so ignoring this macro

Message Details

```
Ignoring macro '<macro>' as it contains no pin information
```

Severity

Warning

LEFWRN_5

BusBitChar is not a delimiter pair

Language

Verilog, VHDL

Rule Description

SpyGlass reports this warning when a `BusBitChar` string in a LEF file consists of more than two characters.

This string should always be a delimiter pair.

Message Details

```
BusBitChar '<string>' is not a delimiter pair
```

Severity

Warning

LEFWRN_6

The missing bus pins of the lef bus will be inferred internally

Language

Verilog, VHDL

Rule Description

SpyGlass reports this warning to indicate missing bus pins of a LEF bus.

SpyGlass infers the missing bus pins internally.

Message Details

```
The missing bus pins of indices '<list-of-indices-of-missing-
bus-pins>' of the lef bus '<bus-name>' will be inferred
internally
```

Severity

Warning

DEF Parsing Messages

Overview

The rules of this category report violations based on the parsing done on DEF files (.def files).

DEFSTX_1

Unexpected token encountered

Language

Verilog, VHDL

Rule Description

Unexpected token encountered

Message Details

```
Parse error at <location>
```

Severity

Syntax

DEFSTX_3

Pins in cell of the def file do not match with the given library

Language

Verilog, VHDL

Rule Description

This rule is flagged when any instance uses a pin that is not present in the cell specified in the library definition.

Message Details

```
Pin '<pin>' is not found in the corresponding cell <cell> in
the library"
```

Severity

Fatal

DEFWRN_1

Cell is not present in the libraries

Language

Verilog, VHDL

Rule Description

When any instance of a cell that is not present in the supplied libraries, the warning is flagged and the cell is treated as a black box.

Message Details

```
Def cell '<cell>' not found in the supplied libraries
```

Severity

Warning

DEFWRN_2

For a vector pin the index of the bus is missing

Language

Verilog, VHDL

Rule Description

This rule is flagged when accessing any vector pin, the index of the bit to be accessed is not specified.

Message Details

```
Bus index of '<vector-pin>' is not provided
```

Severity

Warning

DEFWRN_3

Component that is not declared is used in the def design

Language

Verilog, VHDL

Rule Description

This rule is flagged when a component declaration is missing in the *COMPONENTS* section of the DEF file, and it's connectivity is mentioned in the *NETS* section of the DEF file.

Message Details

```
Component declaration of '<component>' is missing, but nets are
connected to it
```

Severity

Warning

SDC Parsing Messages

Overview

The rules of this category report violations based on the parsing done on Synopsys Design Constraints files (SDC files) that are mainly used by the SpyGlass Constraints solution.

NOTE: *If you specify the* `sdcMsg2Fatal` *command, SpyGlass forces all SDC_* rules to be of the FATAL severity.*

SDC_01

The specified port is not found in the design.

Language

Verilog, VHDL

Cause

The port specified in the constraint file is not found in the design.

Hint

Check for name mismatch with the design; it can be a typo mistake.

Example

```
create_clock -name CLK -period 10 [get_ports in1]
```

The object `in1` should be a port in the design.

Message

```
Port: <port-name> specified in constraint file not found in
design
```

Severity

Error

SDC_02

The specified pin is not found in the design.

Language

Verilog, VHDL

Cause

The pin specified in the constraint file is not found in the design.

Hint

- Check for the name mismatch in the design. It can be a typo mistake.

  Modules inside the design files specified by using the `set_option v <file-name>` or `set_option y <directory-path>` commands are treated as black boxes (except when you specify the `set_option sdc2sgdc yes` command). Therefore, pins specified in such modules will not be found.
- There can also be a mismatch in the hierarchy separator used.

Example

Example 1

```
create_clock -name CLK -period 10 [get_pins F1/q]
```

The `F1/q` object should be a pin/terminal in the design.

Example 2

```
create_clock -name CLK -period 10 [get_pins F1.q]
```

The `F1/q` object is present in the design, but not `F1.q`. You are recommended to either use '/' as a separator or specify the following command:

```
set_hierarchy_separator "."
```

Message

```
Pin: <pin-name> <val> specified in constraint file not found in design
```

Where <*val*> states that inside hierarchy has been specified with `the set_option v <file-name>` or `set_option y <directory-path>` command in the project file.

Severity

Error

SDC_03

The specified port or pin is not found in the design.

Language

Verilog, VHDL

Cause

The port or pin specified in the constraint file is not found in the design.

Hint

- Check for the name mismatch with the design. It can be a typo mistake.
- In case of pins, it can also be a mismatch in the hierarchy-separator used.

Example

Example 1

```
create_clock -name CLK -period 10 {in1}
```

The `in1` object should be a port or pin in the design.

Example 2

```
create_clock -name CLK -period 10 [get_pins F1.q]
```

The `F1/q` object is present in the design, but not `F1.q`. You are recommended to either use '/' as a separator or specify the following command:

```
set_hierarchy_separator "."
```

Message

```
Port/Pin: <port/pin-name> specified in constraint file not
found in design
```

Severity

Error

SDC_04

The specified port, pin, or net is not found in the design.

Language

Verilog, VHDL

Cause

The port, pin, or net specified in the constraint file is not found in design.

Hint

- Check for the name mismatch with the design. It can be a typo mistake.
- In case of pins/nets, it can also be a mismatch in the hierarchy-separator used.

Example

Example 1

```
all_connected -leaf {U1/in1}
```

The `U1/in1` object should be port, pin, or net in the design.

Example 2

```
create_clock -name CLK -period 10 [get_pins F1.q]
```

The `F1/q` object is present in the design, but not `F1.q`. You are recommended to either use '/' as separator or specify the following command:

```
set_hierarchy_separator "."
```

Message

```
Port/Pin/Net: <port/pin/net-name> specified in constraint file
not found in design
```

Severity

Error

SDC_05

The specified pins are not found in the library cell.

Language

Verilog, VHDL

Cause

The pins specified in the constraint file are not found in the library cell.

Hint

Check for name mismatch with the library; it can be a typo mistake.

Example

```
set_disable_timing [get_lib_pins FD1/CP]
```

The object `CP` should be pin of `FD1` cell in the used library.

Message

```
Pins: <pin-name> specified in constraint file not found in
library
```

Severity

Error

SDC_06

The specified cell is not found in the design.

Language

Verilog, VHDL

Cause

The cells specified in the constraint file are not found in the design.

Hint

Check for the name mismatch with design. It can be a typo mistake.

Example

```
set_scan_path chain1 -include_elements {not_present}
```

In the above example, the `not_present` object should be a cell in the design.

Message

```
Cell: "<cell-name>" specified for option "<option-name>" in
set_scan_path not found in design
```

Severity

Warning

SDC_07

The specified port is not found in the design.

Language

Verilog, VHDL

Cause

The ports specified in the constraint file are not found in the design.

Hint

Check for the name mismatch with the design. It can be a typo mistake.

Example

```
set_scan_path chain1 -ordered_elements {not_present}
-class wrapper
```

In the above example, the `not_present` object should be a port in the design.

Message

```
Port: "<port-name>" specified in constraint file for
set_scan_path not found in design
```

Severity

Warning

SDC_08

This list should have at least one element.

Language

Verilog, VHDL

Cause

In case where a list is given as a value for an option, it should have at least one valid element.

NOTE: *Validity parameters vary according to the commands and options.*

Example

```
set_scan_path chain1 -include_elements {not_present}
-scan_slave_clock { [get_cells F1] not_present}
```

In the above example, none of the elements given for `scan_slave_clock` are valid. `F1` is a cell and `not_present` does not exist. Hence, SpyGlass flags the SDC_08 message.

Message

```
The list for "<list-name>" is empty
```

Severity

Error

SDC_09

The specified clock object(s) is not found in the design.

Language

Verilog, VHDL

Cause

The clock(s) used in the constraint file is not found due to any of the following reasons:

- The clock object has not been previously declared by using the `create_clock` or `create_generated_clock` commands.
- The clock object has been declared before, but the declarations resulted in some errors.
- The hierarchical path given for the `get_clocks` command is not present in the design.

Example

Example 1

```
// Start of SDC file
set_input_delay 2 -clock [get_clocks CLK] {in1}
```

Object, `CLK`, used with the `get_clocks` command should be declared previously by using the `create_clock`/`create_generated_clock` commands.

Example 2

```
create_clock -name CLK -period -5 [get_ports clka]
set_input_delay 2 -clock [get_clocks CLK] {in1}
```

Object, `CLK`, used with the `get_clocks` command is declared previously by using the `create_clock` command but is incorrect due to negative value for the `-period` option.

Example 3

```
get_clocks {in1/*}
```

For the object, `{in1/*}`, the `in1` cell should be present in the design

Message

The specified clock(s) : "<clock-name>" is not found in the design.

Severity

Error

SDC_10

The specified cells are not found in the design.

Language

Verilog, VHDL

Cause

The cells specified in the constraint file are not found in the design.

Hint

- Check for the name mismatch with the design. It can be a typo mistake.
- It can also be a mismatch in the hierarchy-separator used.

Example

Example 1

```
set_disable_timing -from CP -to Q [get_cells U1]
```

The object `U1` should be cell in the design.

Example 2

```
get_cells U1.F1 -hier
```

The `U1/F1` object is present in the design, but not `U1.F1`. You are recommended to either use '/' as a separator or specify the following command:

```
set_hierarchy_separator "."
```

Message

```
Cell: <cell-name> specified in constraint file not found in design
```

Severity

Error

SDC_11

Path specified by reset_path is more specific, due to limitation of support this command is ignored

Language

Verilog, VHDL

Cause

The path specified by the `reset_path` command is very specific, for which support is not provided.

Hint

Check the path specified with the `reset_path` command.

Example

```
set_false_path -from A
reset_path -from A -through C -to Z
```

In the above example, the path specified by the `reset_path` command is too specific.

Message

```
Path specified by reset_path is more specific, due to
limitation of support this command is ignored
```

Severity

Info

SDC_12

only one of -stop_propagation, -logical_stop_propagation, -positive, -negative or -pulse should be used

Language

Verilog, VHDL

Cause

Options such as `-stop_propagation`, `-logical_stop_propagation`, `-positive`, `-negative`, or `-pulse` are specified together for the `set_clock_sense` command.

Example

```
set_clock_sense -stop_propagation -positive [get_pins f1/w1]
```

The above command results in violation because usage of `-stop_propagation`, `-logical_stop_propagation`, `-positive`, `-negative`, or `-pulse` cannot be combined. Use one of them.

Message,

```
only one of -stop_propagation, -logical_stop_propagation,
-positive, -negative, or -pulse should be used
```

Severity

Error

SDC_13

The object specified in the -to option is not found in the design.

Language

Verilog, VHDL

Cause

The object specified in the -to option is not found in the design.

Hint

Check for name mismatch with the design; it can be a typo mistake.

Example

```
all_fanin -to qout
```

The object `qout` specified with `-to` option should be port, pin or net in the design.

Message

```
Object: <obj-name> specified in -to not found in design
(allowed types for <command> = port, pin and net)
```

Severity

Error

SDC_14

The object specified in the -from option is not found in the design.

Language

Verilog, VHDL

Cause

The object specified in the -from option is not found in the design.

Hint

Check for name mismatch with the design; it can be a typo mistake.

Example

```
all_fanout -from qin
```

The object `qin` specified with `-from` option should be port, pin or net in the design.

Message

```
Object: <obj-name> specified in -from not found in design
(allowed types for <command> = port, pin and net)
```

Severity

Error

SDC_15

Must specify one of these options: -stop_propagation, -logical_stop_propagation, -positive, -negative, or -pulse

Language

Verilog, VHDL

Cause

Any one of the `-stop_propagation`, `-logical_stop_propagation`, `-positive`, `-negative`, or `-pulse` options is not specified for the `set_clock_sense` command.

Example

```
set_clock_sense [get_pins f1/w1]
```

The above command results in violation because one option must be specified from the `-stop_propagation`, `-logical_stop_propagation`, `-positive`, `-negative`, or `-pulse` options.

Message

```
Must specify one of these options: -stop_propagation,
-logical_stop_propagation, -positive, -negative, or -pulse
```

Severity

Error

SDC_16

The specified designs are not found in the design.

Language

Verilog, VHDL

Cause

The designs specified in the constraint file are not found.

Hint

Check for name mismatch with the design; it can be a typo mistake.

Example

```
set_clock_gating_check -setup 0.75 -hold 0.5 [get_design test]
```

The design `test` should exist in the design.

Message

```
Designs: <design-name> specified in constraint file not found in design
```

Severity

Error

SDC_17

The specified net is not found in the design.

Language

Verilog, VHDL

Cause

The nets specified in the constraint file are not found in design.

Hint

- Check for the name mismatch with the design. It can be a typo mistake.
- It can also be a mismatch in the hierarchy-separator used.

Example

Example 1

```
all_connected -leaf [get_nets U1/in1]
```

The `U1/in1` object should be net in the design.

Example 2

```
all_connected -leaf [get_nets U1.in1]
```

The `U1/in1` object is present in the design, but not `U1.in1`. You are recommended to either use '/' as separator or specify the following command:

```
set_hierarchy_separator "."
```

Message

```
Net: <net-name> specified in constraint file not found in
design
```

Severity

Error

SDC_18

The specified source is not found in the design.

Language

Verilog, VHDL

Cause

The source specified in the constraint file is not found in the design. It can only be a port or pin name.

Hint

- Check whether the name specified is port/pin name.
- Check for the name mismatch with the design. It can be a typo mistake.
- In the case of pins, it can be a mismatch in the hierarchy-separator used.

Example

Example 1

```
create_generated_clock -name CLK -source clka [get_pins abc]
```

Object `clka` used with `-source` option should be a port or pin in the design.

Example 2

```
create_generated_clock -name CLK -source clka [get_pins F1.q]
```

The `F1/q` object is present in the design, but not `F1.q`. You are recommended to either use '/' as a separator or specify the following command:

```
set_hierarchy_separator "."
```

Message

```
Source: <src-name> specified in constraint file not found in
design (allowed types for <command> <option> = port and pin)
```

Severity

Error

SDC_19

The cell is either not available in the specified library or the library path has not been specified.

Language

Verilog, VHDL

Cause

The cell is either not available in the specified library or the library path has not been specified.

Hint

Check for name mismatch with the library; it can be a typo mistake.

Example

```
set_disable_timing -from CP -to Q [get_lib_cells FD1]
```

The object `FD1` should be cell in the used library.

Message

```
Cell: <cell-name> is either not available in specified library
or the library has not been specified
```

Severity

Error

SDC_20

The Lib pin is not available in the specified library.

Language

Verilog, VHDL

Cause

The library pin is not available in the specified library.

Hint

Check for name mismatch with the library; it can be a typo mistake.

Example

```
load_of lsi_10k/FD1/CP
```

The object `CP` should be pin of `FD1` cell in the `lsi_10k` library.

Message

```
Lib pin : <libpin-name> is not available in specified library
```

Severity

Warning

SDC_21

The min/max condition for set_operating_conditions is not found in the library.

Language

Verilog, VHDL

Cause

The library does not have the specific values for the min/max condition of `set_operating_conditions`.

Example

```
set_operating_conditions -min BCCOM -max WCCOM -
analysis_type bc_wc
```

The conditions `BCCOM/WCCOM` specified with `-max/-min` option should be available in the used library.

Message

```
min/max/condition: <condition> for set_operating_conditions not
found in library
```

Severity

Warning

SDC_22

The name of the wire load model is not found in the library.

Language

Verilog, VHDL

Cause

The name of the wire load model is not found in the library; either it does not exist or it is a typo mistake.

Example

```
set_wire_load_model -name “10x10” -library lsi_10k
```

The wire_load `10x10` should exist in the `lsi_10k` library.

Message

```
name of wire_load_model not found in library
```

Severity

Error

SDC_23

No clock group specified

Language

Verilog, VHDL

Cause

No clock group specified for the `set_clock_groups` command.

Example

```
set_clock_groups -asyn
```

The above command results in violation because at least one group should be specified.

Message

```
No clock group specified
```

Severity

Error

SDC_24

The clock is not specified using create_clock or create_generated_clock.

Language

Verilog, VHDL

Cause

The clock referenced in the command of the given message is not specified using `create_clock` or `create_generated_clock` in the constraints file.

Example

```
set_driving_cell -lib_cell AND2 -clock CLK1 -clock_fall {IN1}
```

The object `CLK1` used with the `-clock` option should be declared previously using `create_clock`/`create_generated_clock` commands.

```
create_clock -name CLK1 -period 10.0 [get_ports clka]
```

OR

```
create_generated_clock -name CLK1 -source clka [get_pins abc]
```

Message

```
Clock: <clk-name> used in ‘<command>’ command not specified
using create_clock or create_generated_clock
```

Severity

Error

SDC_25

The clock is not specified using create_clock or create_generated_clock.

Language

Verilog, VHDL

Cause

The clock is not specified using `create_clock` or `create_generated_clock` in the constraints file.

Example

```
find clock CLK1
```

The object `CLK1` that needs to be searched, is of type `clock` and should be declared previously using `create_clock`/ `create_generated_clock` commands.

```
create_clock -name CLK1 -period 10.0 [get_ports clka]
```

OR

```
create_generated_clock -name CLK1 -source clka [get_pins abc]
```

Message

```
Clock: <clk-name> not specified using create_clock or
create_generated_clock
```

Severity

Error

SDC_26

only one of -asynchronous, -physically_exclusive or -logically_exclusive should be used

Language

Verilog, VHDL

Cause

More than one option from `-asynchronous`, `-physically_exclusive` or `-logically_exclusive` are specified for the `set_clock_groups` command.

Example

```
set_clock_groups -asynchronous -physically_exclusive
-group c1
```

The above command results in violation because options `-asynchronous` and `-physically_exclusive` cannot be combined. Use one of them.

Message

```
only one of -asynchronous, -physically_exclusive or
-logically_exclusive be used
```

Severity

Error

SDC_27

Must specify one of these options: -asynchronous, -physically_exclusive or -logically_exclusive

Language

Verilog, VHDL

Cause

None of the options from `-asynchronous`, `-physically_exclusive` or `-logically_exclusive` is specified for the `set_clock_groups` command.

Example

```
set_clock_groups -group c1
```

The above command results in violation because one option from the `-asynchronous`, `-physically_exclusive`, and `-logically_exclusive` options must be specified for the `set_clock_groups` command.

Message

```
Must specify one of these options: -asynchronous, -
physically_exclusive or -logically_exclusive
```

Severity

Error

SDC_28

The specified port is not of input type.

Language

Verilog, VHDL

Cause

The port specified in the constraint file should be of input type.

Example

```
set_logic_dc X1
```

The port `X1` should be an input port.

Message

```
Port: <port-name> specified in constraint file not of type
input
```

Severity

Error

SDC_29

The option -allow_path should be used with the option -asynchronous.

Language

Verilog, VHDL

Cause

The option `-allow_path` should be used with the `-asynchronous` option for the `set_clock_groups` command.

Example

```
set_clock_groups -allow_path -group c1
```

The above command results in violation because option `-allow_path` should be used with the `-asynchronous` option for the `set_clock_groups` command.

Message

```
-allow_path should be used with the option
-asynchronous
```

Severity

Error

SDC_30

The specified port is not of type input or inout.

Language

Verilog, VHDL

Cause

The port specified in the constraint file should be of type input or inout.

Example

```
set_input_delay 10 -clock Clk [get_ports IN1]
```

The port `IN1` should be an input or inout port.

Message

```
Port: <port-name> specified in constraint file not of type
input or inout
```

Severity

Error

SDC_31

The specified port is not of type output or inout.

Language

Verilog, VHDL

Cause

The port specified in the constraint file should be of type output or inout.

Example

```
set_output_delay 10 -clock Clk [get_ports OUT1]
```

The port `OUT1` should be an output or inout port.

Message

```
Port: <port-name> specified in constraint file not of type
output or inout
```

Severity

Error

SDC_32

The specified pin is not the output pin of the specified cell.

Language

Verilog, VHDL

Cause

The specified pin should be the output pin of the cell.

Example

```
set_driving_cell -lib_cell AND2 -clock CLK1 -clock_fall {IN1}
-pin Z -from_pin A
```

The pin `Z` specified by `-pin` option should be an output pin of the `AND2` cell.

Message

```
Pin: <pin-name> is not the output pin of specified cell
```

Severity

Error

SDC_33

The object is not an output or inout port.

Language

Verilog, VHDL

Cause

The object should be an output or inout port.

Example

```
set_fanout_load 2 [get_ports OUT1]
```

The port `OUT1` should be an output or inout port.

Message

```
Object: <obj-name> is not an output/inout port
```

Severity

Error

SDC_34

The specified object either does not exist or is not of the required type.

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type.

Hint

Check name mismatch with the design; it can be a typo mistake.

Message

`Object: <obj-name> specified in constraint file either does not exist or is not of required type`

Severity

Error

SDC_35

The specified object is not a valid start point.

Language

Verilog, VHDL

Cause

The object specified in the constraint file is not a valid start point; a path cannot start from such a point.

A valid start point is:

- Primary input
- Clock
- Sequential cell
- Clock pin of a sequential cell
- Clock-gating cell having clock defined/propagating on any input pin
- Input pin of the clock-gating cell having clock defined/propagating on that pin
- A point that has delay specified by the `set_input_delay` command
- (For DC mode of operation only) Output pin of hierarchical cell directly connected to output pin of a sequential cell and not connected to any other logic within the same hierarchical cell

NOTE: *The SDC_229 and SDC_230 rules have severity as 'WARNING' whereas the SDC_35 and SDC_36 rules have severity as 'ERROR' .*

Example

```
set_false_path -from "u1/CP"
```

The `set_false_path` is defined at clock pin `CP` of the sequential cell `u1`.

Message

```
Object: <obj-name>(<type>) specified in constraint file is not
a valid start point for the <command> command
```

Where <*type*> can be output port, non-sequential cell, non-clock pin of a sequential cell, or pin of a non-sequential cell.

Severity

Error

SDC_36

The specified object is not a valid end point.

Language

Verilog, VHDL

Cause

The object specified in the constraint file is not a valid end point; a path cannot end at such a point.

A valid end point is:

- Primary output
- Sequential cell
- Input pin (except clock and enable pin) of a sequential cell
- Clock-gating cell having clock defined/propagating on any input pin
- Input pin of the clock-gating cell having clock defined/propagating on any pin
- A point that has delay specified by the `set_output_delay` command
- (For DC mode of operation only) Input pin of hierarchical cell directly connected to input pin (except clock and enable pin) of exactly one sequential cell

NOTE: *The SDC_229 and SDC_230 rules have severity as 'WARNING' whereas the SDC_35 and SDC_36 rules have severity as 'ERROR' .*

Example

```
set_false_path -to "u1/D"
```

The `set_false_path` is defined at input pin `D` of the sequential cell `u1`.

Message

```
Object: <obj-name>(<type>) specified in constraint file is not
a valid end point for the <command> command
```

Where *<type>* can be input port, non-sequential cell, clock, enable or output pin of a sequential cell, or pin of a non-sequential cell.

Severity

Error

SDC_37

The specified object is not a valid through point.

Language

Verilog, VHDL

Cause

The object specified in the constraint file is not a valid through point; a path cannot go through a latch or flip-flop.

The valid through points are hierarchical pins and leaf level cells. Additionally, only for PT mode of operation, the through list may specify a hierarchical cell, which implies all output pins of that hierarchical cell.

Example

```
set_false_path -through “u1”
```

The `set_false_path` is defined at leaf level cell `u1` specified by `-through` option, all the paths traversing `u1` are false paths.

Message

```
Object: <obj-name>(not leaf level cell) specified in constraint
file is not a valid through point for the <command> command
```

Severity

Error

SDC_38

Both -name and -all options cannot be specified together

Language

Verilog, VHDL

Cause

Both `-name` and `-all` options are specified together for the `remove_clock_groups` command.

Example

```
remove_clock_groups -asynchronous -name scg1 -all
```

The above command results in violation because options `-name` and `-all` cannot be specified together for the `remove_clock_groups` command. Use one of them.

Message

```
both -name and -all options cannot be specified together
```

Severity

Warning

SDC_39

Must specify one of -name and -all options

Language

Verilog, VHDL

Cause

None of the `-name` and `-all` options are used for the `remove_clock_groups` command.

Example

```
remove_clock_groups c1
```

The above command results in violation because one of `-name` and `-all` options should be specified for the `remove_clock_groups` command.

Message

```
Must specify one of -name or -all options
```

Severity

Error

SDC_40

The list returned by the nested command is empty.

Language

Verilog, VHDL

Cause

The list returned by the nested command must contain some objects.

Example

```
set_input_delay 2 -clock [get_clocks CLK] [get_ports in1]
```

The output of command `get_ports` is null, if port `in1` not found in the design.

Message

```
In ‘<command1>’ command list returned by nested command for
<command2> is empty
```

Severity

Error

SDC_41

The option is not valid for set_clock_uncertainty with the parameter pt set to "no".

Language

Verilog, VHDL

Cause

When the `pt` parameter is set to `no`, SpyGlass behaves in sync with the Design Compiler.

The specified option is not valid for `set_clock_uncertainty` command with the `pt` parameter set to `no`. Either set the `pt` parameter to `yes` or do not use the specified option.

Options which are valid with the `pt` parameter set to `no` are (`-rise`, `-fall`).

Options which are valid with the `pt` parameter set to `yes` are (`-from_edge`, `-to_edge`).

Example

```
set_clock_uncertainty -from clk1 to gclk1 -from_edge rise 3.5
```

The `-from_edge` option cannot be used with the `pt` parameter set to `no`.

Message

```
"<option>" option cannot be used for "set_clock_uncertainty"
with "set_parameter pt" set to "no"
```

Severity

Error

SDC_42

Clock group not found

Language

Verilog, VHDL

Cause

Clock group not found as specified in the `remove_clock_groups` command, either name not exist or phase relationship does not match with the original command.

Example

```
set_clock_groups -asy -name scg1 -group {c2} -group {c3}
remove_clock_groups -phy -name scg1
```

The above command results in violation because phase relationship does not match with the specified `set_clock_groups` command.

Message

`Clock group “<group-name>” not found`

Severity

Warning

SDC_43

The value for the option -analysis_type is invalid.

Language

Verilog, VHDL

Cause

The value for the option `-analysis_type` is invalid; the set of acceptable values are {`single bc_wc on_chip_variation`}.

Example

```
set_operating_conditions
  -analysis_type bc_wc
  -min_library [get_libs {lsi_10k.db:lsi_10k}]
  -max_library [get_libs {lsi_10k.db:lsi_10k}]
  -min fast_-40_1.32 -max slow_100_1.24
```

The valid set for `-analysis_type` option is {`single bc_wc on_chip_variation`}.

Message

```
"<value>" value for option -analysis_type is invalid
```

Severity

Warning

SDC_44

The control value “-high” is already specified for the object.

Language

Verilog, VHDL

Cause

The control value `-high` has already been specified for the object; it cannot be specified twice.

Example

```
set_clock_gating_check -setup 0.75 -hold 0.5 -high CHIP1
set_clock_gating_check -setup 0.55 -hold 0.3 -high CHIP1
```

The `-high` option is specified twice on the object `CHIP1`. This is invalid.

Message

```
Control value ‘-high’ is already specified for object <obj-name>
```

Severity

Warning

SDC_45

The control value “-low” is already specified for the object.

Language

Verilog, VHDL

Cause

The control value `-low` has already been specified for the object; it cannot be specified twice.

Example

```
set_clock_gating_check -setup 0.75 -hold 0.5 -low CHIP1
set_clock_gating_check -setup 0.55 -hold 0.3 -low CHIP1
```

The option `-low` is specified twice on the object `CHIP1`. This is invalid.

Message

```
Control value ‘-low’ is already specified for object <obj-name>
```

Severity

Warning

SDC_46

The option "clock" is used in the command "set_input_transition" while the parameter "pt" is set to no.

Language

Verilog, VHDL

Cause

When the `pt` parameter is set to `no`, then SpyGlass behaves in sync with the Design Compiler.

The option `clock` is used in the command `set_input_transition` while the `pt` parameter is set to `no`.

Either set the `pt` parameter to `yes` or do not use the option `clock`.

Example

```
set_input_transition -fall -min12.0 -clock Clk1 {C}
```

The option `-clock` should not be used when the `pt` parameter is set to `no`.

Message

```
Option "clock" used in command "set_input_transition" while
'set_parameter pt' is set to no
```

Severity

Warning

SDC_47

The option "clock_fall" is used in the command "set_input_transition" while the parameter "pt" is set to no.

Language

Verilog, VHDL

Cause

When the `pt` parameter is set to `no`, then SpyGlass behaves in sync with DesignCompiler.

The option `clock_fall` is used in the command `set_input_transition` while the `pt` parameter is set to `no`.

Either set the `pt` parameter to `yes` or do not use the option `clock_fall`.

Example

```
set_input_transition -fall -min 12.0 -clock Clk1 -clock_fall
{C}
```

The `-clock_fall` option should not be used when the `pt` parameter is set to `no`.

Message

```
Option "clock_fall" used in command "set_input_transition"
while 'set_parameter pt' is set to no
```

Severity

Warning

SDC_48

Unknown value is specified for the -level field.

Language

Verilog, VHDL

Cause

The value specified for the `-level` field is unknown; known values for the `-level` field are {`rtl prelayout postlayout all`}.

This option is to be used in SGDC file with `sdc_data` constraint.

Example

```
sdc_data -type test.sdc -level rtl -corner best -mode test
```

The valid set for option `-level` are {`rtl prelayout postlayout all`}.

Message

```
Unknown value (<value>) specified for -level field
```

Severity

Warning

SDC_49

Unknown value is specified for the -corner field.

Language

Verilog, VHDL

Cause

The value specified for the `-corner` field is unknown; known values for the `-corner` field are {`worst best typical`}.

This option is to be used in SGDC file with `sdc_data` constraint.

Example

```
sdc_data -type test.sdc -level rtl -corner best -mode test
```

The valid set for option `-corner` are {`worst best typical`}.

Message

```
Unknown value (<value>) specified for -corner field
```

Severity

Warning

SDC_50

The clock is not valid. It is either not present or not specified using create_clock.

Language

Verilog, VHDL

Cause

The clock referenced in the command of the given message is either not present in the design or it is not specified by using the `create_clock` command in the constraints file.

Example

Consider the following examples:

```
//case 1
all_registers -clock not_present //case 1
```

```
//case 2
create_clock -name CLK -period -1
all_registers -clock CLK  //case 2
```

SpyGlass reports a violation in both the above cases.

In the above example, the clock specified in case1 is not present in the design and in case2, the clock name is not specified by using the `create_clock` command.

Message

```
Clock: <clock-name> used in '<command-name>' command either not present or not correctly specified using create_clock
```

Severity

Error

SDC_51

The library name given as an argument cannot be found in the current set of technology libraries.

Language

Verilog, VHDL

Cause

The library name given as an argument by using the `-library` argument cannot be found in the current set of technology libraries. Following can be the possible reasons:

- By mistake, library file name has been specified as an argument instead of the library name.
- The library name specified is not found in the set of technology libraries specified by using the `sglib` option of the `read_file` command (in project file) for SpyGlass and `-tech` option for OA flow.

Example

SpyGlass flags the SDC_51 message in the following example as library file name has been specified as an argument instead of the library name:

```
set_wire_load_model -library {lsi_10k.lib} -name 5K_6LMP
```

SpyGlass flags the SDC_51 message in the following example if `lsi_10k.sglib` is not specified by the `sglib` option of the `read_file` project file command.

```
set_wire_load_model -library {lsi_10k} -name 5K_6LMP
```

Message

```
Library name : "<lib-name>" not found. Please provide the
appropriate technology library by using "<command>"
```

Severity

Warning

SDC_52

The file has been specified more than once.

Language

Verilog, VHDL

Cause

The file has been specified more than once in the option given in the message. Use a file-name only once in a particular option.

Example

```
source constraints/test.tcl
source constraints/test.tcl
```

In the above SDC file, if the same file is sourced multiple times this will result in above violation. For correction remove the duplicate source occurrence of the file.

Message

```
File ‘<file1-name>’ has been specified more than once in
<file2-name>
```

Severity

Warning

SDC_53

At least one of time, power, capacitance, resistance, voltage, current should be specified with set_units command

Language

Verilog, VHDL

Cause

None of `time`, `power`, `capacitance`, `resistance`, `voltage`, or `current` is specified with the `set_units` command.

Example

```
set_units
```

The above command results in violation because none of `time`, `power`, `capacitance`, `resistance`, `voltage`, or `current` is specified.

Message

```
At least one of time, power, capacitance, resistance, voltage,
current should be specified with set_units command
```

Severity

Error

SDC_54

The specified object should be a unique element.

Language

Verilog, VHDL

Cause

The specified object should be a unique element. It should either be a port, a pin or a net.

Example

```
all_connected -leaf [get_nets U1/in*]
```

If the above command `get_nets` returns multiple objects (such as, `U1/in1`, `U1/in2`), results in the violation, because command `all_connected` can have only unique objects.

Message

```
Object: ‘<obj-name>’ should be a unique element
```

Severity

Error

SDC_55

The specified file cannot be opened.

Language

Verilog, VHDL

Cause

The specified file cannot be opened. Either the name of the file is not entered correctly or the file is badly placed. The file should be available in the current working directory or the path of the file should be specified using the `set_option I <directory-name>` project file command.

Example

Consider that the following command is used in the .sgdc file:

```
sdc_data -file notExist.sdc
```

If the above file name does not exist or cannot be opened for reading, then this will result in a violation.

Message

```
Can’t open File <file-name>
```

Severity

Error

SDC_56

C-Style comments are not allowed

Language

Verilog, VHDL

Cause

C-Style comments are not allowed in the constraint file.

Example

```
 /* This is not a valid comment */
```

C-Style comments are not allowed. You can use comments in following manner:

```
# This is a valid comment
```

Message

```
C-Style comment not allowed
```

Severity

Error

SDC_57

Given scale is not a valid scale value

Language

Verilog, VHDL

Cause

Given scale is not a valid scale value. A valid scale value can be any of `f`, `p`, `n`, `u`, `m`, `k`, or `M`.

Example

```
set_units -time 10bs
```

The above command results in violation because `b` is not a valid scale value.

Message

```
‘<scale>’ is not a valid scale value, should be one of f, p, n, u, m, k, M
```

Severity

Error

SDC_58

Wrong unit specified

Language

Verilog, VHDL

Cause

The specified unit is incompatible with the given field.

Example

```
set_units -time 10mA
```

The above command results in violation because A is invalid unit for time.

Message

```
Unit: ‘<unit-name>’ specified is incompatible with ‘<field-name>’
```

Severity

Error

SDC_59

ERROR occurred due to improper use of Tcl commands.

Language

Verilog, VHDL

Cause

ERROR occurred due to improper use of Tcl commands.

Example

1. Consider the following command:

   ```
   set abc [get_ports in1] [get_pins F1/Q]
   ```

   The `set` command is having three arguments. However, for the correct usage only two arguments are allowed.
2. Consider the following command:

   ```
   source not_exist.sdc
   ```

   If the above mentioned file name doesn't exist or cannot be opened for reading, then this will result in a violation.

Message

```
 <error-message>
```

<error-message> is a message returned by Tcl.

Severity

Error

SDC_60

ERROR occurred in the filter expression.

Language

Verilog, VHDL

Cause

The filter command searches a group of objects and returns only those objects for which the conditional expression is true; the expression may be a wrong regular expression.

Example

```
filter_collection $cell_list “@name%%AN*”
```

The %% operator is invalid.

Valid operators are {==, !=, >, >=, <, <=, =~, !~}.

```
filter_collection $cell_list “@name==AN* FD*”
```

There are multiple values to compare.

Message

```
ERROR in filter expression <expr-name>
```

Severity

Error

SDC_61

Cannot set size_only on the specified cell.

Language

Verilog, VHDL

Cause

Cannot set `size_only` on the specified cell. `size_only` can be used only on leaf-level cells. Check whether or not the specified cell is a leaf-level cell.

Example

```
set_size_only U1 false
```

If the cell `U1` is not a leaf-level cell, you cannot specify `set_size_only` command on it.

Message

```
ERROR: Cannot set size_only on cell <cell-name>
```

Severity

Warning

SDC_62

The object specified in the option is of invalid type for the given command.

Language

Verilog, VHDL

Cause

The object specified in the option is of invalid type for the given command.

Example

```
create_generated_clock -name gclk -source [get_nets w1] -
divide_by 2 {F1/CP}
```

The net `w1` is wrong type for the `-source` option. The allowed types are port or pin.

Message

```
<obj-type> ‘<obj-name>’ is of wrong type for option <option> in
command <command>
```

Severity

Warning

SDC_63

Value is not of valid type

Language

Verilog, VHDL

Cause

The specified value in the constraint file is not of valid type.

Example

```
set_case_analysis 100 [get_ports IN]
```

The value `100` is invalid, valid values are {`0`, `1`, `zero`, `one`, `rise`, `fall`, `rising`, `falling`}.

Message

```
Value: <val-name> is not a valid value
```

Severity

Warning

SDC_64

More than one clock is specified with the -clock option.

Language

Verilog, VHDL

Cause

More than one clock is specified with the `-clock` option; the `-clock` option should have only one clock name.

Example

```
set_driving_cell
  -lib_cell AND2 -clock “CLK1 CLK2” -clock_fall {IN1}
```

Multiple clocks are specified with `-clock` option. The `-clock` option can have only one clock.

Message

`More than one clock specified with -clock option`

Severity

Warning

SDC_65

The collection has multiple objects

Language

Verilog, VHDL

Cause

Even though some tools allow multiple objects, most tools do not allow this command on a collection of multiple objects.

The collection has multiple objects. It should return a single object.

Example

```
get_object_name “FD1 U1”
```

The collection `FD1 U1` has more than one object, but the command `get_object_name` allows single object in the collection.

Message

```
Collection has multiple objects
```

Severity

Warning

SDC_66

The specified attribute does not exist.

Language

Verilog, VHDL

Cause

The specified attribute does not exist.

Example

```
get_attribute FD1 name
```

Message

```
Attribute <attribute-name> does not exist on <obj-name>
<command>
```

Severity

Warning

SDC_67

The specified attribute does not exist.

Language

Verilog, VHDL

Cause

The specified attribute does not exist.

Example

```
get_attribute [get_cells U1] period
```

The attribute `period` does not exist on cells. It is applicable only on clocks and generated clocks.

Message

```
Attribute <attribute-name> does not exist on <obj-name>
```

Severity

Error

SDC_68

The input pin of the cells should be specified when loads are set on the output port using the set_load command.

Language

Verilog, VHDL

Cause

When loads are set on the output port using the `set_load` command, the input pin of the cells should also be specified.

Example

```
set_load -pin_load [expr 30 * [load_of lsi_10k/FD1/D]]
[all_outputs]
```

Load is set on the output ports then the direction of pins used in the `load_of` command should be input.

Message

```
Input pin of the cells should be specified when loads are set
on output port using command set_load
```

Severity

Warning

SDC_69

The current design is not the same as the loaded design, hence all the subsequent constraints will not be parsed.

Language

Verilog, VHDL

Cause

The current design and the loaded design are not same, hence all the constraints will not be parsed.

Check the current_design option in the SGDC file.

Example

SGDC file:

```
current_design top
  sdcscehma -type top.sdc
```

SDC File (top.sdc):

```
current_design test
  create_clock -name CLK -period 10.0 [get_ports clka]
```

In SDC file top.sdc, the current design `test` is not same as the design loaded `top`. So, the commands for this current design will not be parsed. Either correct the design name in SGDC file or correct the current design in the SDC file.

Message

```
Current design ″<du1-name>″ is not the same as design loaded
″<du2-name>″. All the constraints for the current design ″<du1-
name>″ will not be parsed
```

Severity

Warning

SDC_70

Only a single object is allowed for the option, for the command; multiple objects are not allowed.

Language

Verilog, VHDL

Cause

Only a single object is allowed for the option, for the command; multiple objects are not allowed.

Example

```
create_generated_clock
  -name gclk1 -source {clka, data1}
  -divide_by 2 data1
```

The source specified with `-source` option should be single.

Message

```
Only a single object is allowed for the option "<option>", for
the command "<command>"
```

Severity

Warning

SDC_71

all_connected should be given a unique element.

Language

Verilog, VHDL

Cause

all_connected should be given a unique element.

Example

```
all_connected -leaf “U1/in1 U1/in2”
```

In the above command only single element is allowed.

Message

```
all_connected should be given a unique element
```

Severity

Error

SDC_72

Invalid value is passed to the is_true option

Language

Verilog, VHDL

Cause

Invalid value is passed to the `is_true` option; the valid values are `true`/`false` or `1`/`0`.

Example

```
set x abc
[is_true $x]
```

The value passed is invalid. Valid values are {`0`, `1`, `true`, `false`, `TRUE`, `FALSE`}.

Message

```
Invalid value passed to is_true
```

Severity

Warning

SDC_73

This particular command cannot be applied.

Language

Verilog, VHDL

Cause

The command `set_size_only` / `set_mode` / `reset_mode` cannot be applied on the given object list. It can only be used on the leaf-level cells. Check whether or not the object list contains only leaf-level cells.

Example

```
set_size_only U1 false
```

If the cell `U1` is not a leaf-level cell, you cannot specify `set_size_only` command on it.

Message

```
Command <command> cannot be applied on <object>
```

Severity

Error

SDC_74

Hierarchy separator specified is not acceptable

Language

Verilog, VHDL

Cause

The specified hierarchy separator is not acceptable; valid hierarchy separators are { / @ ^ # . | }.

Example

```
[get_pins -hsc @ FD1@CP]
```

Message

```
Hierarchy separator ‘<hier-sep>’ specified is not acceptable
```

Severity

Error

SDC_75

The get_license command is not understood/supported by the SpyGlass Constraints solution.

Language

Verilog, VHDL

Message

```
get_license command not understood/supported by SpyGlass
Constraints solution. Default return value is 1
```

Severity

Warning

SDC_76

blocksize command has -min value greater than the -max value

Language

Verilog, VHDL

Cause

The `blocksize` command has the `-min` value greater than the `-max` value; the value of `-max` should be greater than the value of `-min`.

Example

```
blocksize -min 50 -max 20
```

The value specified for max should be greater than the min value.

Message

```
blocksize command has -min value (<val1>) specified greater
than the -max value (<val2>)
```

Severity

Error

SDC_77

design-mode not given in map_design_mode command

Language

Verilog, VHDL

Cause

Design-mode not given in `map_design_mode` command.

Example

```
map_design_mode -from I1/CK
```

In the above command, design-mode name is not specified.

Message

```
design-mode not given in map_design_mode command
```

Severity

Error

SDC_78

While mapping design mode to cell mode, either specify a combination of cell-mode and instances OR specify a set of paths, but not a mixture of two

Language

Verilog, VHDL

Cause

While mapping design mode to cell mode, either specify a combination of cell-mode and instances OR specify a set of paths, but not a mixture of two.

Example

```
map_design_mode D1 read -from a1/*
remove_design_mode D1 read -from a1/*
```

In the above commands, a mixture of two mapping techniques is not allowed.

Message

```
This command cannot use both combination of the
cell-mode+instances and path-list.
```

Severity

Error

SDC_79

While mapping design mode to cell mode, either specify a combination of cell-mode and instances OR specify a set of paths, but not a mixture of two

Language

Verilog, VHDL

Cause

While mapping design mode to cell mode, either specify a combination of cell-mode and instances OR specify a set of paths, but not a mixture of two.

Example

```
map_design_mode D1
```

In the above command, none of the two mapping techniques are used.

Message

```
Incomplete map_design_mode command; specify either cell-mode+instances or path-list
```

Severity

Error

SDC_80

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are port and net.

Example

```
set_load 3 {F1/q}
```

The object `F1/q` is not of required type. It should be port or net.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <obj> =
Port and Net)
```

Severity

Error

SDC_81

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are port and design.

Example

```
set_max_transition 2.0 {F1/q}
```

The object `F1/q` is not of required type. It should be port or design.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <obj> =
Port and Design)
```

Severity

Error

SDC_82

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are port, cell, and design.

Example

```
set_wire_load_model -name “10x10” -library lsi_10k {F1/q}
```

The object F1/q is not of required type. It should be port, cell or design.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <obj> =
Port, Cell and Design)
```

Severity

Error

SDC_83

Objects specified in constraint file not found in design

Language

Verilog, VHDL

Cause

The object specified in the constraint file is not found in the design; the allowed types for the SDC command are port (input/inout) and design.

Example

```
set_max_fanout 18.5 in1
```

The object `in1` does not exist in the design. It should be port (input/inout) or design.

Message

```
Objects: <obj-name> specified in constraint file not found in
design (allowed types for <obj> = port(input/inout) and design)
```

Severity

Error

SDC_84

Objects specified in constraint file not found in design

Language

Verilog, VHDL

Cause

The object specified in the constraint file is not found in the design; the allowed type for the SDC command is pin.

Example

```
set_clock_sense -stop a1
```

The above command results in violation because object `a1` is not of required type; it should be pin.

Message

```
Objects: <obj-name> specified in constraint file not found in
design (allowed types for <obj> = pin)
```

Severity

Error

SDC_85

Objects specified in constraint file not found in design

Language

Verilog, VHDL

Cause

The object specified in the constraint file is not found in the design; the allowed types for the SDC command are port, clocks and design.

Example

```
set_max_transition 12.0 a1/CP
```

Here, object "a1/CP" is not of required type; it should be port, clock, or design.

Message

```
Objects: <obj-name> specified in constraint file not found in
design (allowed types for <obj> = port, clock and design)
```

Severity

Error

SDC_86

Cannot get attribute for more than one object.

Language

Verilog, VHDL

Cause

Cannot get attribute for more than one object.

Example

```
get_attribute {U1 U2} name
```

Here, the attribute`{name}` cannot be returned for the both`{U1 U2}` objects.

Message

```
Cannot get attribute '<attribute>' for more than one object
required type
```

Severity

Error

SDC_87

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are clock, pin, cell, and design.

Example

```
set_clock_gating_check -setup 0.75 -hold 0.5 CHIP1
```

The object `CHIP1` either does not exist or not of required type. It should be clock, pin, cell, or design.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <command> =
clock, pin, cell, design and port (when ”-pt” option is set to
yes/supermode))
```

Severity

Error

SDC_88

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are clock, pin, and latch cell.

Example

```
set_max_time_borrow 2.0 in1
```

The object `in1` either does not exist or not of required type. It should be clock, pin, or latch cell.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <obj> =
clock, pin and latch cell)
```

Severity

Error

SDC_89

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are net, cell, library-cell, reference, and design.

Example

```
set_dont_touch in1
```

The object `in1` either does not exist or not of required type. It should be net, cell, library-cell, reference, or design.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <command> =
net, cell, library-cell, reference and design)
```

Severity

Error

SDC_90

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are clock, port, and pin.

Example

```
set_clock_latency 0.8 -source -early “in1”
```

The object `in1` either does not exist or not of required type. It should be clock, port or pin.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <command> =
clock, port and pin)
```

Severity

Error

SDC_91

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are port, pin, library-pin, cell, and library-cell.

Example

```
set_disable_timing {U1}
```

The object `U1` either does not exist or is not of required type. It should be port, pin, library-pin, cell, or library-cell.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <obj> =
port, pin, library-pin, cell and library-cell)
```

Severity

Error

SDC_92

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are port, pin, and leaf cell.

Example

```
set_multicycle_path 2.0 -through {FD1/CP}
```

The object `FD1/CP` either does not exist or not of required type. It should be port, pin, or leaf-cell.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <command>
<option> = port, pin and leaf cell)
```

Severity

Error

SDC_93

The object specified in the constraint file either does not exist or is not of the required type.

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type. The allowed types for the SDC command are clock, port (output/inout), (enable/data/scanin/scanenable/clear/load/preset) pin of a sequential cell, pin with output delay set, and sequential cell.

Example

```
set_multicycle_path 2.0 -to {FD1/CP}
```

The object FD1/CP either does not exist or not of required type. It should be clock, port (output/inout), (enable/data/scanin/scanenable/clear/load/preset) pin of a sequential cell, pin with output delay set, or sequential cell.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <command>
<option> = clock, port(output/inout), (enable/data/scanin/
scanenable/clear/load/preset) pin of a sequential cell, pin
with output delay set and sequential cell)
```

Severity

Error

SDC_94

The object specified in the constraint file either does not exist or is not of the required type.

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type. The allowed types for the SDC command are clock, port (input/inout), clock pin of a sequential cell, (enable/scanenable/data/scanin/clear/preset/load) pin of a level-sensitive latch, pin with input delay set, and sequential cell.

Example

```
set_multicycle_path 2.0 -from {FD1/Q}
```

The object FD1/Q either does not exist or not of required type. It should be clock, port (input/inout), clock pin of a sequential cell, (enable/scanenable/data/scanin/clear/preset/load) pin of a level-sensitive latch, pin with input delay set, or sequential cell.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <command>
<option> = clock, port(input/inout), clock pin of a sequential
cell, (enable/scanenable/data/scanin/clear/preset/load) pin of
a level-sensitive latch, pin with input delay set and
sequential cell)
```

Severity

Error

SDC_95

The object specified in the constraint file either does not exist or is not of the required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are cells and designs.

Example

```
set_wire_load_selection_groups WCOM F1/Q
```

The above command results in violation because object `F1/Q` either does not exist or not of required type, it should be cell or design.

Message

```
Object: <object-name> specified in the constraint file either
does not exist or is not of the required type (allowed types
for <command-name> = cells and designs
```

Severity

Error

SDC_96

Pin not found in the specified cell

Language

Verilog, VHDL

Cause

The pin specified in the constraint file is not found in the specified cell.

Hint

Check for name mismatch with the design; it can be a typo mistake.

Example

```
set_driving_cell -lib_cell AND2 -pin Z -from_pin A {IN1}
```

Pin `Z` is not found in the specified cell `AND2`.

Message

```
Pin: <pin-name> not found in the specified cell
```

Severity

Error

SDC_97

More than one library is specified with the -library option

Language

Verilog, VHDL

Cause

More than one library is specified with the `-library` option; the `-library` option should have only one file name.

Example

```
set_lcd_pulse_width_multipliers -mult_worst 2.0 -
mult_nominal 1.5 -mult_best 1.8 -library “lsi_10k GTECH”
INLDT
```

More than one library `lsi_10k GTECH` cannot be specified with `-library` option.

Message

```
More than one library specified with -library option
```

Severity

Error

SDC_98

The command set_signal_type cannot be used when the "set_parameter pt" is set to yes/supermode.

Language

Verilog, VHDL

Cause

When the `set_parameter pt` command is set to `yes` or `supermode`, SpyGlass behaves in sync with prime time.

The command `set_signal_type` cannot be used with the `set_parameter pt` command set to `yes` or `supermode`. Either set the parameter `pt` to `no` or do not use the command.

Message

```
Command ”set_signal_type” used while ‘set_parameter pt’ option
is set to yes/supermode
```

Severity

Warning

SDC_99

Both cell-mode name and instance list should be given for successful mapping of the design-mode with the given cell-mode

Language

Verilog, VHDL

Cause

Both cell-mode name and instance list should be given for successful mapping of the design-mode with the given cell-mode.

Example

```
map_design_mode D1 read
```

In the above command, no instance list is provided.

Message

```
Missing instance list for map/remove_design_mode command
```

Severity

Error

SDC_100

Unsupported SDC version

Language

Verilog, VHDL

Cause

The specified version of SDC is not supported; only versions 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, and 2.0 are currently supported.

Example

```
set sdc_version 2.1
```

In the above example, SDC version `2.1` is not supported. The valid set of SDC versions are 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, and 2.0.

Message

```
Unsupported SDC Version <ver-num>. Only versions 1.1, 1.2, 1.3,
1.4, 1.5, 1.6, 1.7, 1.8, 1.9, and 2.0 are supported at this
time
```

Severity

Error

SDC_101

Setting SDC version has no effect here. It has already been set during parsing or set to default.

Language

Verilog, VHDL

Cause

Setting `sdc_version` (if required) should be the very first command in the first SDC file being parsed.

The complete SDC file is parsed with a single version only. This version can either be the default version (which is the latest version) or the one set as soon as parsing starts.

If `sdc_version` is set after some valid command has been parsed for a particular file, it does not have any effect. Hence, SpyGlass flags this warning.

Example

```
//START OF SDC FILE
set VARIABLE 1
set sdc_version 1.3
puts "$VARIABLE"
```

In the above case, the SDC version "1.3" is not set in the beginning of the SDC file. Hence, SpyGlass flags a warning in this case.

```
//START OF SDC FILE
set sdc_version 1.3
set VARIABLE 1
puts "$VARIABLE"
```

In the above case, the SDC version "1.3" is set at the beginning of the SDC file. Hence, no warning is flagged in this case.

Message

```
Setting SDC version has no effect here
```

Severity

Warning

SDC_102

Value not specified

Language

Verilog, VHDL

Cause

Value is not specified for the command.

Example

```
create_clock -name Clk1 -period
```

The value for `-period` option should be specified.

Message

```
value not specified for "<command>"
```

Severity

Error

SDC_103

Incorrect argument

Language

Verilog, VHDL

Cause

The given argument is incorrect for the command specified in the constraint file.

Example

```
create_generated_clock -name gclkC -source clka [get_pins F1/
Q] -master_clock [get_clocks clk1] -abc -divide_by 2
```

In the above command, the `-abc` option is incorrect for the `create_generated_clock` command.

Message

```
incorrect argument "<option>" for "<command>"
```

Severity

Error

SDC_104

Incorrect argument

Language

Verilog, VHDL

Cause

The given argument is incorrect for the command specified in the constraint file.

Example

```
set_annotated_delay -f {in1}
```

The option `-f` can be mapped to `-from` or `-fall`. So, there are too many interpretations. Use one of the option.

Message

```
incorrect argument "<option>" for "<command>" Too many
interpretations
```

Severity

Error

SDC_105

Duplicate argument

Language

Verilog, VHDL

Cause

The command specified in the constraint file contains duplicate argument.

Example

```
create_clock -name Clk1 -period 10 -period {clk1}
```

The option `-period` cannot be specified twice; so, remove the second occurrence.

Message

```
duplicate arguments "<arg-name>" for "<command>"
```

Severity

Error

SDC_106

Incorrect argument

Language

Verilog, VHDL

Cause

Either incorrect or too many arguments are passed to the command.

Example

```
create_clock -name Clk1 -period 10 {clk1} {in1}
```

There are too many arguments passed to the command; remove extra arguments.

Message

```
Incorrect argument "<arg-name>" for "<command>" (too many
arguments)
```

```
Incorrect argument "<arg-name>" for "<command>" (too many
arguments, -h is used as -hierarchical option)
```

Severity

Error

SDC_107

Incorrect value must be in a valid range

Language

Verilog, VHDL

Cause

The value specified in the command is incorrect; appropriate value is given in the message.

Example

```
group_path -name serious -from {I1 I2} -to {O5 O7} -weight
150
```

The value `150` specified with the `-weight` option is incorrect. It should be within the range of 0 to 100.

Message

```
incorrect value for ”<command>”, must be “<value>”
```

Severity

Error

SDC_108

Incorrect value must be one of the enumerated values

Language

Verilog, VHDL

Cause

The value specified in the command is incorrect.

Example

```
set_case_analysis 100 [get_ports IN]
```

The value `100` is incorrect. Valid values are {`0`, `1`, `zero`, `one`, `rise`, `fall`, `rising`, `falling`}.

Message

```
incorrect value for "<command>", must be {<val>}
```

Severity

Error

SDC_109

Incorrect value must be of float type and non zero

Language

Verilog, VHDL

Cause

The value specified in the command is incorrect. It must be of the float type or non zero.

Example

```
create_clock -name CLk1 -period in1 [get_ports in[abc]]
```

The value `in1` for `-period` option is incorrect. It must be of the float type and non zero.

Message

```
incorrect value for ”<command>”, must be float type and non zero
```

Severity

Error

SDC_110

Incorrect value must be of the integer type

Language

Verilog, VHDL

Cause

The value specified in the command is incorrect; it must be of integer type.

Example

```
create_generated_clock
  -name gclk1 -source clka
  -divide_by 2.5 [get_pins abc]
```

The value `2.5` for `-divide_by` option is incorrect. It must be of integer type.

Message

```
incorrect value for ”<command>”, must be integer type
```

Severity

Error

SDC_111

Incorrect value must be of the string type

Language

Verilog, VHDL

Cause

The value specified in the command is incorrect; it must be of string type.

Message

`incorrect value for ″<command>″, must be string type`

Severity

Error

SDC_112

Incorrect value must be of the list type

Language

Verilog, VHDL

Cause

The value specified in the command is incorrect; it must be of list type.

Message

`incorrect value for ”<command>”, must be list type`

Severity

Error

SDC_113

Incorrect type of elements of list

Language

Verilog, VHDL

Cause

The type of elements of the specified list is incorrect; they should be of the type given in the message.

Example

```
create_generated_clock
  -name gclk1 -source clka
  -edges {in1 in2 in3} [get_pins abc]
```

The list `{in1 in2 in3}` for `-edges` option is of incorrect type. It must be of float type.

Message

```
incorrect type of elements of list for ”<option>”, must be
<value-type> type
```

Severity

Error

SDC_114

Incorrect length of list

Language

Verilog, VHDL

Cause

The length of the specified list is incorrect.

Example

```
create_generated_clock
  -name gclk1 -source clka
  -edges {1 3 5 7} [get_pins abc]
```

The length of list `{1 3 5 7}` for `-edges` option is incorrect. It must be an odd number and greater than 2.

Message

```
incorrect length of list for ”<option>”
```

Severity

Error

SDC_115

Incorrect set of required parameters

Language

Verilog, VHDL

Cause

The set of parameters for the specified command is incorrect.

Message

`incorrect set of required parameters for ″<command>″`

Severity

Error

SDC_116

Not found in design

Language

Verilog, VHDL

Cause

Check the given object; it does not exist in the design.

Example

```
create_clock -name CLk1 -period 10 [get_ports in[abc]]
```

The object `in[abc]` does not exist in the design; use some integer value to use the bit object.

Message

```
<obj-name> not found in design
```

Severity

Error

SDC_117

Object list not specified

Language

Verilog, VHDL

Cause

The object_list is not specified for the specified command; it is a mandatory to specify the object_list.

Example

```
get_attribute period
```

The object_list is mandatory.

Message

```
object_list is not specified
```

Severity

Error

SDC_118

Cannot specify -from without -to

Language

Verilog, VHDL

Cause

The option `-from` cannot be specified without the `-to` option; the command `set_disable_timing` should have both the options.

Example

```
set_disable_timing my_lib/FD1 -from D
```

Option `-from` cannot be specified without `-to` option.

Message

```
cannot specify -from without -to
```

Severity

Error

SDC_119

Cannot specify -to without -from

Language

Verilog, VHDL

Cause

The option `-to` cannot be specified without the `-from` option; the command `set_disable_timing` should have both the options.

Example

```
set_disable_timing my_lib/FD1 -to Q
```

Option `-to` cannot be specified without `-from` option.

Message

```
cannot specify -to without -from
```

Severity

Error

SDC_120

Neither -name nor -default option is specified

Language

Verilog, VHDL

Cause

Neither `-name` nor `-default` option is specified; one of these options should be specified for the `group_path` command.

Example

```
group_path -name serious -from {I1 I2} -to {O5 O7}
group_path -default -to {OUT1 CLK2}
```

Message

```
neither -name nor -default option is specified
```

Severity

Error

SDC_121

Both -name and -default options cannot be specified together

Language

Verilog, VHDL

Cause

Both `-name` and `-default` options cannot be specified together for the `group_path` command.

Example

```
group_path -name serious -default -from {I1 I2} -to {O5 O7}
```

Options `-name` and `-default` cannot be specified together; so use one of them.

Message

```
both -name and -default together cannot be specified
```

Severity

Error

SDC_122

One of -from, -to, or -through options should be given

Language

Verilog, VHDL

Cause

One of `-from`, `-to`, or `-through` options should be given for the `group_path` command.

Example

```
group_path -name serious
```

One of `-from`, `-to`, and `-through` should be specified in the above command.

Message

```
one of -from, -to, or -through should be given
```

Severity

Error

SDC_123

scan_signal_type not given

Language

Verilog, VHDL

Cause

The option `scan_signal_type` is not given; it is mandatory for the `set_scan_signal` command.

Example

```
set_scan_signal -port my_scan_in -chain chain3
```

Option `scan_signal_type` is mandatory for the `set_scan_signal` command.

Message

```
scan_signal_type not given
```

Severity

Error

SDC_124

Value to the -port option not given

Language

Verilog, VHDL

Cause

Value to the `-port` option should be given for the `set_scan_signal` command.

Example

```
set_scan_signal test_scan_in -chain chain3
```

Option `-port` is mandatory for the `set_scan_signal` command.

Message

```
value to -port option not given
```

Severity

Error

SDC_125

signal_type missing

Language

Verilog, VHDL

Cause

The option `signal_type` is missing for the `set_signal_type` command.

Example

```
set_signal_type SIP1
```

Option `signal_type` is mandatory for the `set_signal_type` command.

Message

```
signal_type missing
```

Severity

Error

SDC_126

port_list is empty

Language

Verilog, VHDL

Cause

The port_list for the `set_signal_type` command should not be empty.

Example

```
set_signal_type “test_scan_in”
```

Port list is mandatory.

Message

```
port_list is empty
```

Severity

Error

SDC_127

mode_list should be specified

Language

Verilog, VHDL

Cause

The mode_list should be specified for the set_mode command.

Example

```
set_mode Uram1
```

mode_list is mandatory.

Message

```
mode_list should be specified for the <command> command
```

Severity

Error

SDC_128

instance_list should be specified for set_mode / reset_mode

Language

Verilog, VHDL

Cause

The instance_list should be specified in `set_mode`/`reset_mode` commands.

Example

```
set_mode READ
```

instance_list is mandatory.

Message

```
instance_list should be specified for in the set_mode/
reset_mode commands
```

Severity

Error

SDC_129

No value is specified either to the -net option or to the -cell option

Language

Verilog, VHDL

Cause

No value is specified either to the -net option or to the `-cell` option; the command `set_annotated_delay` should contain one of the options.

Example

```
set_annotated_delay
  -rise -min -load_delay net 12.3 -from U1/Z -to U2/A
```

One of `-net` or `-cell` option should be specified in the above command.

Message

```
neither value to -net nor -cell option is specified
```

Severity

Error

SDC_130

Both -net and -cell options cannot be specified together

Language

Verilog, VHDL

Cause

Both `-net` and `-cell` options cannot be specified together; the command `set_annotated_delay` should contain only one of these options.

Example

```
set_annotated_delay
  -net -cell -rise -min -load_delay net 12.3
  -from U1/Z -to U2/A
```

Both `-net` and `-cell` options cannot be specified together; use one of them.

Message

```
both -net and -cell together cannot be specified
```

Severity

Error

SDC_131

Both operating_condition and max/min operating_condition cannot be specified together

Language

Verilog, VHDL

Cause

Both operating_condition and max/min operating_condition cannot be specified together for the `set_operating_conditions` command.

Example

```
set_operating_conditions
  -min_library [get_libs {lsi_10k.db:lsi_10k}]
  -max_library [get_libs {lsi_10k.db:lsi_10k}]
  -min fast_-40_1.32 -max slow_100_1.24 WCIND
```

Both operating_condition (`WCIND`) and `-min`/`-max` options cannot be specified together; use one of them.

Message

```
both operating_condition and max/min operating_condition
together cannot be specified
```

Severity

Error

SDC_132

Cannot specify -min_library without -min

Language

Verilog, VHDL

Cause

The option `-min_library` cannot be specified without the `-min` option for the `set_operating_conditions` command.

Example

```
set_operating_conditions
  -min_library [get_libs {lsi_10k.db:lsi_10k}]
  -max_library [get_libs {lsi_10k.db:lsi_10k}]
  -max slow_100_1.24
```

Option `-min_library` cannot be specified without `-min` option.

Message

```
cannot specify -min_library without -min
```

Severity

Error

SDC_133

Cannot specify -max_library without -max

Language

Verilog, VHDL

Cause

The option `-max_library` cannot be specified without the `-max` option for the `set_operating_conditions` command.

Example

```
set_operating_conditions
  -min_library [get_libs {lsi_10k.db:lsi_10k}]
  -max_library [get_libs {lsi_10k.db:lsi_10k}]
  -min fast_-40_1.32
```

Option `-max_library` cannot be specified without `-max` option.

Message

```
cannot specify -max_library without -max
```

Severity

Error

SDC_134

Cannot specify -max without -min

Language

Verilog, VHDL

Cause

The option `-max` cannot be specified without specifying the `-min` option. Both the options should be specified together for the `set_operating_conditions` command.

Example

```
set_operating_conditions -max fast_-40_1.32
```

Option `-max` cannot be specified without `-min` option.

Message

```
cannot specify -max without -min
```

Severity

Error

SDC_135

Cannot specify -min without -max

Language

Verilog, VHDL

Cause

The option `-min` cannot be specified without specifying the `-max` option. Both the options should be specified together for the `set_operating_conditions` command.

Example

```
set_operating_conditions -min fast_-40_1.32
```

Option `-min` cannot be specified without `-max` option.

Message

```
cannot specify -min without -max
```

Severity

Error

SDC_136

library_cell_pin not specified

Language

Verilog, VHDL

Cause

The option `library_cell_pin` is not specified for the `load_of` command.

Example

```
load_of lsi_10k/FD1/CP
```

Use the library pin `lsi_10k/FD1/CP` with `load_of` command.

Message

```
library_cell_pin not specified
```

Severity

Error

SDC_137

transition_time not specified

Language

Verilog, VHDL

Cause

The option `transition_time` is not specified for the `set_ideal_transition` command.

Example

```
set_ideal_transition 1.2 -rise {A B C}
```

Option `transition_time` (`1.2`) should be specified for the `set_ideal_transition` command.

Message

```
transition_time not specified
```

Severity

Error

SDC_138

port_list is not specified

Language

Verilog, VHDL

Cause

The option `port_list` is not specified for the `set_ideal_transition` command.

Example

```
set_ideal_transition 1.2 -rise {A B C}
```

Port pin list `{A B C}` should be specified for the `set_ideal_transition` command.

Message

```
port_list is not specified
```

Severity

Error

SDC_139

-to list is empty

Language

Verilog, VHDL

Cause

The `-to` list returned by the `all_fanin` command is empty.

Example

```
all_fanin -flat -to
```

The list returned to the `-to` option should not be empty.

Message

```
-to list empty
```

Severity

Error

SDC_140

Neither -from nor -clock_tree is specified

Language

Verilog, VHDL

Cause

Neither `-from` nor `-clock_tree` is specified; one of these options should be specified for the `all_fanout` command.

Example

```
all_fanout -only_cells
```

Neither `-from` nor `-clock_tree` option is specified; specify one of them.

Message

```
neither -from nor -clock_tree is specified
```

Severity

Error

SDC_141

Both -from and -clock_tree options cannot be specified together

Language

Verilog, VHDL

Cause

Both `-from` and `-clock_tree` options cannot be specified together; only one of these options should be specified for the `all_fanout` command.

Example

```
all_fanout -clock_tree -only_cells -from tin
```

Both `-from` and `-clock_tree` options cannot be specified together; use one of them.

Message

```
both -from and -clock_tree together cannot be specified
```

Severity

Error

SDC_142

type not specified

Language

Verilog, VHDL

Cause

The option type is not specified for the find command.

Example

```
set ports [find port in1]
```

Type `port` of the object `in1` should be specified for the `find` command.

Message

```
type not specified
```

Severity

Error

SDC_143

attribute_name not specified

Language

Verilog, VHDL

Cause

The option `attribute_name` should be specified for the `get_attribute` command.

Example

```
set ports [get_attribute op1 full_name]
```

Attribute name `full_name` should be specified for the `get_attribute` command.

Message

```
attribute_name not specified
```

Severity

Error

SDC_144

both level_sensitive and edge_triggered options cannot be specified

Language

Verilog, VHDL

Cause

Both `level_sensitive` and `edge_triggered` options cannot be specified for the `all_inputs` and `all_outputs` commands.

Example

```
set ports [all_inputs -level_sensitive -edge_triggered]
```

Both `-level_sensitive` and `-edge_triggered` options cannot be specified together; use one of them.

Message

```
both level_sensitive and edge_triggered cannot be specified
```

Severity

Error

SDC_145

-period value missing

Language

Verilog, VHDL

Cause

The value for the `-period` option is missing for the command.

Example

```
create_clock -name CLK -period 10.0 [get_ports clka]
```

Value(`10.0`) for the option `-period` should be specified for the `create_clock` command.

Message

```
-period value missing
```

Severity

Error

SDC_146

Either -name or port_pin_list should be specified.

Language

Verilog, VHDL

Cause

One of the options `-name` or `port_pin_list` should be specified to `create_clock` command.

Example

```
create_clock -name CLK -period 10.0 [get_ports clka]
```

One of `-name` option or `port_pin_list` should be specified.

Message

```
either -name or port_pin_list should have been specified
```

Severity

Error

SDC_147

Collection is not specified

Language

Verilog, VHDL

Cause

The option collection is not specified for the `remove_from_collection` command.

Example

```
set cPorts [remove_from_collection {CLK in1 Clk1} CLK]
```

Collection `{CLK in1 Clk1}` should be specified for the `remove_from_collection` command.

Message

```
collection is not specified
```

Severity

Error

SDC_148

base_collection is not specified

Language

Verilog, VHDL

Cause

The option `base_collection` is not specified for the `add_to_collection` command.

Example

```
set col [add_to_collection $col [get_cells *data_reg*]]
```

Base collection `$col` should be specified for the `add_to_collection` command.

Message

```
base_collection is not specified
```

Severity

Error

SDC_149

object_spec is not specified

Language

Verilog, VHDL

Cause

The value of object_spec is missing for the `add_to_collection` command.

Example

```
set col [add_to_collection $col [get_cells *data_reg*]]
```

Value of object_spec `[get_cells *data_reg*]` should be specified for the `add_to_collection` command.

Message

```
object_spec is not specified
```

Severity

Error

SDC_150

Value not specified

Language

Verilog, VHDL

Cause

The value should be specified for this command.

Example

```
set_case_analysis 1 [get_ports IN]
```

Value (`1`) should be specified.

Message

```
value not specified
```

Severity

Error

SDC_151

Neither of -setup or -hold options specified

Language

Verilog, VHDL

Cause

The option `-setup` or `-hold` should be specified for the `set_clock_gating_check` command, otherwise default values of 0 would be used for each

Example

```
set_clock_gating_check CHIP1
```

One of `-setup` and `-hold` option should be specified.

Message

```
Neither of -setup or -hold options specified
```

Severity

Info

SDC_152

delay not specified

Language

Verilog, VHDL

Cause

The value for the `delay` option should be specified for the `set_clock_latency` command.

Example

```
set_clock_latency 0.8 -source -early [get_clocks CLK1]
```

The value(`0.8`) for the `delay` option should be specified for the `set_clock_latency` command.

Message

```
delay not specified
```

Severity

Error

SDC_153

Cannot specify these options with object_list

Language

Verilog, VHDL

Cause

The options `-from`, `-to`, `-rise_from`, `-fall_from`, `-rise_to`, `-fall_to`, `-from_edge`, `-to_edge`, `-rise`, or `-fall` cannot be specified with the `object_list` for the `set_clock_uncertainty` command.

Example

```
set_clock_uncertainty 0.4 -from PHI1 -to PHI1 {in1}
```

Options {`-from`, `-to`, `-rise_from`, `-fall_from`, `-rise_to`, `-fall_to`, `-from_edge`, `-to_edge`, `-rise`, or `-fall`} cannot be combined with the `object_list` for the `set_clock_uncertainty` command.

Message

```
Cannot specify ‘-from or -to rise_from or fall_from or rise_to
or fall_to or from_edge or to_edge or rise or fall’ with
‘object_list’
```

Severity

Error

SDC_154

Cannot specify -from/-rise_from/-fall_from without -to/-rise_to/-fall_to

Language

Verilog, VHDL

Cause

The options `-from/-rise_from/-fall_from` cannot be specified without the options `-to/-rise_to/-fall_to` for the `set_clock_uncertainty` command.

Example

```
set_clock_uncertainty 0.4 -from PHI1
```

Options {`-from/-rise_from/-fall_from`} cannot be specified without {`-to/-rise_to/-fall_to`} options for the `set_clock_uncertainty` command.

Message

```
Cannot specify -from/-rise_from/-fall_from without -to/-
rise_to/-fall_to
```

Severity

Error

SDC_155

Cannot specify -to/-rise_to/-fall_to without -from/-rise_from/-fall_from

Language

Verilog, VHDL

Cause

The options `-to/-rise_to/-fall_to` cannot be specified without the options `-from/-rise_from/-fall_from` for the `set_clock_uncertainty` command.

Example

```
set_clock_uncertainty 0.4 -to PHI1
```

Options {`-to/-rise_to/-fall_to`} cannot be specified without {`-from/-rise_from/-fall_from`} options for the `set_clock_uncertainty` command.

Message

```
Cannot specify -to/-rise_to/-fall_to without -from/-rise_from/-fall_from
```

Severity

Error

SDC_156

Cannot specify '-from' with '-rise_from/-fall_from'

Language

Verilog, VHDL

Cause

The option `-from` cannot be specified with the options `-rise_from/-fall_from` for the `set_clock_uncertainty` command.

Example

```
set_clock_uncertainty 0.4 -from PHI1 -to PHI1
```

Option `-from` cannot be specified without {`-rise_from/-fall_from`} options for the `set_clock_uncertainty` command.

Message

```
Cannot specify '-from' with '-rise_from/-fall_from'
```

Severity

Error

SDC_157

Cannot specify '-rise_from' with '-from/-fall_from'

Language

Verilog, VHDL

Cause

The option `-rise_from` cannot be specified with the options `-from/-fall_from` for the `set_clock_uncertainty` command.

Example

```
set_clock_uncertainty 0.4 -rise_from PHI1 -rise_to PHI1
```

Option `-rise_from` cannot be specified without {`-from/-fall_from`} options for the `set_clock_uncertainty` command.

Message

```
Cannot specify '-rise_from' with '-from/-fall_from'
```

Severity

Error

SDC_158

Cannot specify '-to' with '-rise_to/-fall_to'

Language

Verilog, VHDL

Cause

The option `-to` cannot be specified with the options `-rise_to/-fall_to` for the `set_clock_uncertainty` command.

Example

```
set_clock_uncertainty 0.4 -from PHI1 -to PHI1
```

Option `-to` cannot be specified without {`-rise_to/-fall_to`} options for the `set_clock_uncertainty` command.

Message

```
Cannot specify '-to' with '-rise_to/-fall_to'
```

Severity

Error

SDC_159

Cannot specify '-rise_to' with '-to/-fall_to'

Language

Verilog, VHDL

Cause

The option `-rise_to` cannot be specified with the options `-to/-fall_to` for the `set_clock_uncertainty` command.

Example

```
set_clock_uncertainty 0.4 -rise_from PHI1 -rise_to PHI1
```

Option `-rise_to` cannot be specified without {`-to/-fall_to`} options for the `set_clock_uncertainty` command.

Message

```
Cannot specify '-rise_to' with '-to/-fall_to'
```

Severity

Error

SDC_160

Cannot specify '-rise_from or -fall_from' with '-from_edge'

Language

Verilog, VHDL

Cause

The options `-rise_from` or `-fall_from` cannot be specified with the option `-from_edge` for the `set_clock_uncertainty` command.

Example

```
set_clock_uncertainty
  0.4 -rise_from PHI1 -to PHI2 -from_edge rise
```

Options `-rise_from`/`-fall_from` cannot be specified with `{-from_edge}` option for the `set_clock_uncertainty` command.

Message

```
Cannot specify '-rise_from or -fall_from' with '-from_edge'
```

Severity

Error

SDC_161

Must specify one of these options: clock_list, -from/-rise_from/-fall_from or -to/-rise_to/-fall_to

Language

Verilog, VHDL

Cause

One option must be specified from `clock_list`, `-from`/`-rise_from`/`-fall_from` or `-to`/`-rise_to`/`-fall_to` for the `set_clock_uncertainty` command. Usage of -rise/-fall with -rise_from/-fall_from, -rise_to/-fall_to is permitted.

Example

```
set_clock_uncertainty 0.4 -from_edge rise
```

One option must be specified from the {`clock_list`, `-from`/`-rise_from`/`-fall_from` or `-to`/`-rise_to`/`-fall_to`} options for the `set_clock_uncertainty` command.

Message

```
Must specify one of these options: clock_list, -from/-
rise_from/-fall_from or -to/-rise_to/-fall_to
```

Severity

Error

SDC_162

Cannot specify '-rise_to or -fall_to' with '-rise, -fall or -to_edge'

Language

Verilog, VHDL

Cause

The options `-rise_to` or `-fall_to` cannot be specified with the options `-rise`, `-fall`, or `-to_edge` for the `set_clock_uncertainty` command.

Example

```
set_clock_uncertainty
  0.4 -rise_to PHI1 -rise_from PHI2 -to_edge rise
```

Options `-rise_to`/`-fall_to` cannot be specified with {`-rise`/`-fall`/`-to_edge`} options for the `set_clock_uncertainty` command.

Message

```
Cannot specify '-rise_to or -fall_to' with '-rise, -fall or -
to_edge'
```

Severity

Error

SDC_163

source not specified

Language

Verilog, VHDL

Cause

The source is not specified for the clock in the `create_generated_clock` command.

Example

```
create_generated_clock
  -name gclk1 -divide_by 2 [get_pins abc]
```

The option `-source` should be specified.

Message

```
source not specified
```

Severity

Error

SDC_164

port_pin_list not specified

Language

Verilog, VHDL

Cause

The option `port_pin_list` is not specified, which is a mandatory option for the `create_generated_clock` command.

Example

```
set_annotated_transition -rise -min 12.3
```

The option `port_pin_list` should be specified.

Message

```
port_pin_list not specified
```

Severity

Error

SDC_166

One of -edges, -multiply_by, -divide_by, or -combinational must be specified

Language

Verilog, VHDL

Cause

Either of the options `-edges`, `-multiply_by`, `-divide_by`, or `-combinational` must be specified for the `create_generated_clock` command.

Example

```
create_generated_clock
  -name gclk1 -source clka [get_pins abc]
```

One of options {`-divide_by`, `-multiply_by`, `-edges`, or `-combinational`} should be specified.

Message

```
one of -edges, -multiply_by, -divide_by, or -combinational must
be specified
```

Severity

Error

SDC_167

Only one of -edges, -multiply_by or -divide_by would be used

Language

Verilog, VHDL

Cause

Only one of the options from `-edges`, `-multiply_by`, or `-divide_by` should be specified for the `create_generated_clock` command. If you specify more than one option with this command, the one with the highest priority is retained and others are ignored. The priority order (from highest to lowest) among the options is given below:

1. `-divide_by`
2. `-multiply_by`
3. `-edges`

Example

```
create_generated_clock
  -name gclk1 -source clka -divide_by 2
  -edges {1 3 5} [get_pins abc]
```

In the above command, `-divide_by` option will be populated and `-edges` option will be ignored.

Message

`Only one of -edges, -multiply_by or -divide_by would be used`

Severity

Error

SDC_168

lib_cell not specified

Language

Verilog, VHDL

Cause

The option `lib_cell` should be specified in the `set_driving_cell` command.

Example

```
set_driving_cell {IN1}
```

Option `-lib_cell` should be specified for the `set_driving_cell` command.

Message

```
lib_cell not specified
```

Severity

Error

SDC_169

The option -clock_fall should be used with the option -clock.

Language

Verilog, VHDL

Cause

The option `-clock_fall` should be used with the option `-clock`; the option `-clock` cannot be specified alone for the `set_driving_cell` command.

Example

```
set_driving_cell -lib_cell AND2 -clock_fall {IN1}
```

Option `-clock_fall` should be used with the `-clock` option for the `set_driving_cell` command.

Message

```
-clock_fall should be used with -clock
```

Severity

Error

SDC_170

delay_value not specified

Language

Verilog, VHDL

Cause

The value of the option `delay_value` should be specified for the command.

Example

```
set_input_delay 1.2 -clock [get_clocks CLK1] [all_inputs]
```

Value(`1.2`) for the delay option should be specified.

Message

```
delay_value not specified
```

Severity

Error

SDC_171

area value not specified

Language

Verilog, VHDL

Cause

The value of the area should be specified for the `set_max_area` command.

Example

```
set_max_area 0.0
```

Value(`0.0`) for the area option should be specified for the `set_max_area` command.

Message

```
area value not specified
```

Severity

Error

SDC_172

net_list is not specified

Language

Verilog, VHDL

Cause

The option `net_list` is not specified for the `set_resistance` command.

Example

```
set_resistance 300 U1/U2/NET3
```

Option `net_list` (`U1/U2/NET3`) should be specified for the `set_resistance` command.

Message

```
net_list is not specified
```

Severity

Error

SDC_173

fanout_number not specified

Language

Verilog, VHDL

Cause

The value of the option fanout_number should be specified for the `set_port_fanout_number` command.

Example

```
set_port_fanout_number 5 [get_ports IN1]
```

Value(`5`) for the `fanout_number` option should be specified for the `set_port_fanout_number` command.

Message

```
fanout_number not specified
```

Severity

Error

SDC_174

transition_value not specified

Language

Verilog, VHDL

Cause

The value of the option `transition_value` should be specified for the command.

Example

```
set_max_transition 2.0 [get_ports IN1]
```

Value(`2.0`) for the `transition_value` option should be specified for the `set_max_transition` command.

Message

```
transition_value not specified
```

Severity

Error

SDC_175

fanout_value not specified

Language

Verilog, VHDL

Cause

The value of the option `fanout_value` should be specified for the `set_max_fanout` command.

Example

```
set_max_fanout 18.5 TEST
```

Value(`18.5`) for the `fanout_value` option should be specified for the `set_max_fanout` command.

Message

```
fanout_value not specified
```

Severity

Error

SDC_176

transition value not specified

Language

Verilog, VHDL

Cause

The value of the option transition should be specified for the command.

Example

```
set_clock_transition 0.75 CLK
```

Value(`0.75`) for the transition option should be specified for the command.

Message

```
transition value not specified
```

Severity

Error

SDC_177

clock_list is not specified

Language

Verilog, VHDL

Cause

The option `clock_list` is not specified for the `set_clock_transition` command.

Example

```
set_clock_transition 0.75 CLK
```

Option `clock_list (CLK)` should be specified for the `set_clock_transition` command.

Message

```
clock_list is not specified
```

Severity

Error

SDC_178

resistance not specified

Language

Verilog, VHDL

Cause

The option resistance should be specified for the `set_drive` command.

Example

```
set_drive 2.0 {A, B, C}
```

Value(`2.0`) for the resistance option should be specified for the `set_drive` command.

Message

```
resistance not specified
```

Severity

Error

SDC_179

porosity_value not specified

Language

Verilog, VHDL

Cause

The option `porosity_value` should be specified for the `set_min_porosity` command.

Example

```
set_min_porosity 20 TEST
```

Value(`20`) for the `porosity_value` option should be specified for the `set_min_porosity` command.

Message

```
porosity_value not specified
```

Severity

Error

SDC_180

capacitance_value not specified

Language

Verilog, VHDL

Cause

The option `capacitance_value` should be specified for the commands `set_min_capacitance` and `set_max_capacitance`.

Example

```
set_min_capacitance 12.0 [get_ports in1]
```

Value(`12.0`) for the `capacitance_value` option should be specified for the `set_min_capacitance` and `set_max_capacitance` command.

Message

```
capacitance_value not specified
```

Severity

Error

SDC_181

path_multiplier value not specified

Language

Verilog, VHDL

Cause

The value of the option `path_multiplier` should be specified for the `set_multicycle_path` command.

Example

```
set_multicycle_path 2.0 -from {FD1/CP}
```

Value(`2.0`) for the `path_multiplier` option should be specified for the `set_multicycle_path` command.

Message

```
path_multiplier value not specified
```

Severity

Error

SDC_182

size not specified

Language

Verilog, VHDL

Cause

The option `size` should be specified for the `set_wire_load_min_block_size` command.

Example

```
set_wire_load_min_block_size 200.0
```

Option `size` (`200.0`) should be specified for the `set_wire_load_min_block_size` command.

Message

```
size not specified
```

Severity

Error

SDC_183

mode_name not specified

Language

Verilog, VHDL

Cause

The option `mode_name` should be specified for the `set_wire_load_mode` command.

Example

```
set_wire_load_mode enclosed
```

Option `mode_name` (`enclosed`) should be specified for the `set_wire_load_mode` command.

Message

```
mode_name not specified
```

Severity

Error

SDC_184

-name not specified

Language

Verilog, VHDL

Cause

The option `-name` should be specified for the command.

Example

```
set_wire_load_model -name “10x10”
```

Option `-name` should be specified for the `set_wire_load_model` command.

Message

```
-name not specified for the "<command>" command
```

Severity

Error

SDC_185

group_name not specified

Language

Verilog, VHDL

Cause

The option `group_name` should be specified for the `set_wire_load_selection_group` command.

Example

```
set_wire_load_selection_group “default_by_area” -library
CORE9GPLL
```

Option `group_name` (`default_by_area`) should be specified for the `set_wire_load_selection_group` command.

Message

```
group_name not specified
```

Severity

Error

SDC_186

-library not specified

Language

Verilog, VHDL

Cause

The option `-library` should be specified for the `create_operating_conditions` command.

Example

```
create_operating_conditions -name OC3 -library
IBM_CMOS5S6_SC -proc 1.0 -temp 100.0 -volt 4.0 -rail_voltages
{VTT 3.5 VDDQ 3.5}
```

Option `-library` should be specified for the `create_operating_conditions` command.

Message

```
-library not specified
```

Severity

Error

SDC_187

-temperature not specified

Language

Verilog, VHDL

Cause

The option `-temperature` should be specified for the `create_operating_conditions` command.

Example

```
create_operating_conditions -name OC3 -lib IBM_CMOS5S6_SC -
proc 1.0 -temp 100.0 -volt 4.0 -rail_voltages {VTT 3.5 VDDQ
3.5}
```

Option `-temperature` should be specified for the `create_operating_conditions` command.

Message

```
-temperature not specified
```

Severity

Error

SDC_188

-voltage not specified

Language

Verilog, VHDL

Cause

The option `-voltage` should be specified for the command.

Example

```
create_operating_conditions -name OC3 -lib IBM_CMOS5S6_SC -
proc 1.0 -temp 100.0 -volt 4.0 -rail_voltages {VTT 3.5 VDDQ
3.5}
```

Option `-voltage` should be specified for the `create_operating_conditions` command.

Message

```
-voltage not specified for the "<command>" command
```

Severity

Error

SDC_189

-process not specified

Language

Verilog, VHDL

Cause

The option `-process` should be specified for the `create_operating_conditions` command.

Example

```
create_operating_conditions -name OC3 -lib IBM_CMOS5S6_SC -
proc 1.0 -temp 100.0 -volt 4.0 -rail_voltages {VTT 3.5 VDDQ
3.5}
```

Option `-process` should be specified for the `create_operating_conditions` command.

Message

```
-process not specified
```

Severity

Error

SDC_190

-op_worst not specified

Language

Verilog, VHDL

Cause

The option `-op_worst` should be specified for the `create_lcd_operating_conditions` command.

Example

```
create_lcd_operating_conditions -op_worst value -mult_worst
0.6 -op_nominal value -mult_nominal 0.2 -op_best value -
mult_best 0.1 -library lsi_10k lcd_late
```

Option `-op_worst` should be specified for the `create_lcd_operating_conditions` command.

Message

```
-op_worst not specified
```

Severity

Error

SDC_191

-op_nominal not specified

Language

Verilog, VHDL

Cause

The option `-op_nominal` should be specified for the `create_lcd_operating_conditions` command.

Example

```
create_lcd_operating_conditions -op_worst value -mult_worst
0.6 -op_nominal value -mult_nominal 0.2 -op_best value -
mult_best 0.1 -library lsi_10k lcd_late
```

Option `-op_nominal` should be specified for the `create_lcd_operating_conditions` command.

Message

```
-op_nominal not specified
```

Severity

Error

SDC_192

-op_best not specified

Language

Verilog, VHDL

Cause

The option `-op_best` should be specified for the `create_lcd_operating_conditions` command.

Example

```
create_lcd_operating_conditions -op_worst value -mult_worst
0.6 -op_nominal value -mult_nominal 0.2 -op_best value -
mult_best 0.1 -library lsi_10k lcd_late
```

Option `-op_best` should be specified for the `create_lcd_operating_conditions` command.

Message

```
-op_best not specified
```

Severity

Error

SDC_193

-mult_worst not specified

Language

Verilog, VHDL

Cause

The option `-mult_worst` should be specified for the `create_lcd_operating_conditions` command.

Example

```
create_lcd_operating_conditions -op_worst value -mult_worst
0.6 -op_nominal value -mult_nominal 0.2 -op_best value -
mult_best 0.1 -library lsi_10k lcd_late
```

Option `-mult_worst` should be specified for the `create_lcd_operating_conditions` command.

Message

```
-mult_worst not specified
```

Severity

Error

SDC_194

-mult_nominal not specified

Language

Verilog, VHDL

Cause

The option `-mult_nominal` should be specified for the `create_lcd_operating_conditions` command.

Example

```
create_lcd_operating_conditions -op_worst value -mult_worst
0.6 -op_nominal value -mult_nominal 0.2 -op_best value -
mult_best 0.1 -library lsi_10k lcd_late
```

Option `-mult_nominal` should be specified for the `create_lcd_operating_conditions` command.

Message

```
-mult_nominal not specified
```

Severity

Error

SDC_195

-mult_best not specified

Language

Verilog, VHDL

Cause

The option `-mult_best` should be specified for the `create_lcd_operating_conditions` command.

Example

```
create_lcd_operating_conditions -op_worst value -mult_worst
0.6 -op_nominal value -mult_nominal 0.2 -op_best value -
mult_best 0.1 -library lsi_10k lcd_late
```

Option `-mult_best` should be specified for the `create_lcd_operating_conditions` command.

Message

```
-mult_best not specified
```

Severity

Error

SDC_196

library name not mentioned

Language

Verilog, VHDL

Cause

The library name option should be specified.

Example

```
read_lib {lsi_10k}
```

Library name `lsi_10k` should be specified.

Message

```
library name not mentioned
```

Severity

Error

SDC_198

INTERNAL: unknown type of param.

NOTE: *The SDC_198 rule is an internal rule.*

Language

Verilog, VHDL

Cause

The parameter specified in the file is of unknown type.

Message

`INTERNAL: unknown type of param: “<param>”, ignored`

Severity

Warning

SDC_199

INTERNAL: error while executing user's checking procedure

NOTE: *The SDC_199 rule is an internal rule.*

Language

Verilog, VHDL

Cause

Error occurred while executing user’s checking procedure for the type specified in the file.

Message

```
INTERNAL: error while executing user’s checking procedure for
type: “<type>”, ignored
```

Severity

Warning

SDC_200

No such collection

Language

Verilog, VHDL

Cause

The collection which you specified does not exist.

Hint

Check for name mismatch with the design; it can be a typo mistake.

Example

```
filter_collection FD* “@name==FD1”
```

The object `FD*` is not a valid collection.

Message

```
Collection: <object> not found
```

Severity

Error

SDC_201

Library cell not found

Language

Verilog, VHDL

Cause

The specified library cell not found.

Example

```
set_dont_use [get_lib_cells tech_lib/G1]
```

Library cell `G1` not found.

Message

```
Library cell: <lib_cell-name> not found
```

Severity

Error

SDC_202

The object code library is not found. Use gateslib or sglib option of the read_file command to specify the liberty format library.

Language

Verilog, VHDL

Cause

The object code library specified for `get_libs` command is not found.

Use the `gateslib` or `sglib` option of the `read_file` command to specify a liberty format library.

Any option other than these two is invalid to specify the liberty format library.

Example

```
get_libs {misc_cmos}
```

Library `misc_cmos` not found.

Use `gateslib`/`sglib` options of the `read_file` project file command to specify the `misc_cmos` library.

Message

```
The object code library <lib-name> specified for get_libs
command not found or use gateslib or sglib option of the
read_file command to specify the liberty format library for
<lib-name>
```

Severity

Error

SDC_203

The object specified in the constraint file either does not exist or is not of the required type.

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type. Only ports are allowed for the `set_load` command because `-wire_load/-pin_load/-fall/-rise` option is specified in the command.

Example

```
set_load -pin_load -wire_load “F1/CP”
```

The object `F1/CP` either does not exist or is not of required type; only ports are allowed when `-wire_load/-pin_load/-fall/-rise` option is specified for the `set_load` command.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (Ports are allowed for
set_load command because -wire_load/-pin_load/-fall/-rise
option is specified in command)
```

Severity

Error

SDC_204

The object specified in the constraint file is not of the required type.

Language

Verilog, VHDL

Cause

The object specified in the constraint file is not of the required type; only cells are allowed for the `set_disable_timing` command because `-from`/`-to` option is specified in the command.

Example

```
set_disable_timing -from A -to Z {lsi_10k/AN2/A}
```

The object `lsi_10k/AN2/A` either does not exist or not of required type; only cells are allowed when `-from`/`-to` option is specified for the `set_disable_timing` command.

Message

```
Object: <obj-name> specified in constraint file is not of
required type (Cells are allowed for set_disable_timing command
because -from -to option is specified in command)
```

Severity

Error

SDC_205

Pin specified in -from option does not exist in the cell.

Language

Verilog, VHDL

Cause

The pin specified in the constraint file with the option `-from` does not exist in the cell. Therefore, the cell is not populated.

Example

```
set_disable_timing -from X -to Z [get_lib_cells lsi_10k/AN2]
```

The pin `X` specified with the `-from` option does not exist on the `AN2` cell.

Message

```
Pin: ‘<pin-name>’ specified in constraint file with -from
option does not exist in ‘<cell-name>’ cell. So cell ‘<cell-
name>’ is not populated
```

Severity

Error

SDC_206

Pattern not provided for unalias command

Language

Verilog, VHDL

Cause

No pattern is provided for the `unalias` command; specify a pattern for the `unalias` command.

Example

```
unalias getp
```

Pattern `getp` should be specified for the `unalias` command.

Message

```
Pattern not provided for unalias command
```

Severity

Error

SDC_207

The given pattern does not match any pattern to unalias.

Language

Verilog, VHDL

Cause

The given pattern does not match any pattern to unalias.

Example

```
unalias getp
```

The pattern `getp` does not match any pattern to unalias.

Message

```
<pattern-name> does not match any pattern to unalias
```

Severity

Error

SDC_208

The option -leaf should be specified for nets only.

Language

Verilog, VHDL

Cause

The option `-leaf` should be specified for nets only.

Example

```
all_connected -leaf [get_pins U1/in1]
```

The option `-leaf` is only for nets; either does not use `-leaf` option or specify nets in the object list.

Message

```
-leaf option should be specified for nets only
```

Severity

Error

SDC_209

Clock is not declared on the object

Language

Verilog, VHDL

Cause

Clock is not declared on the object specified with the argument `-source`.

Example

```
create_generated_clock -edges {1 3 5} -source clka [get_pins foo2]
```

No clock is declared on the object `clka` specified with the `-source` option.

Message

```
Clock not declared on object ‘<obj-name>’ specified with argument ”-source”
```

Severity

Error

SDC_210

The object specified in the constraint file either does not exist or is not of the required type.

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type; the allowed types for the SDC command are ports and cells.

Example

```
set_operating_conditions
  -min BCCOM -max WCCOM -object_list F1/q
```

The object F1/q either does not exist or not of required type; it should be port or cell.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <obj> =
ports or cells)
```

Severity

Error

SDC_211

The pin specified in the option -to does not exist in the cell.

Language

Verilog, VHDL

Cause

The pin specified in the constraint file with the option `-to` does not exist in the cell. Therefore, the cell is not populated.

Example

```
set_disable_timing -from A -to X [get_lib_cells lsi_10k/AN2]
```

The pin `X` specified with the `-to` option does not exist on the `AN2` cell.

Message

```
Pin: ‘<pin-name>’ specified in constraint file with -to option
does not exist in ‘<cell-name>’ cell. So cell ‘<cell-name>’ is
not populated
```

Severity

Error

SDC_212

Invalid magma commands are specified or the specified magma commands are not supported in the SpyGlass product.

Language

Verilog, VHDL

Cause

Invalid magma commands are specified or the specified magma commands are not supported in the SpyGlass product.

NOTE: *This rule has been deprecated and will be removed in a future release.*

Message

```
Invalid magma commands or Magma commands not supported in
Spyglass Constraints. For report please look into file
"TcM2PtNoSupport" in <wdir>/spyglass_spysch/spyglass_sdc
directory
```

Severity

Error

SDC_213

Magma commands encountered with syntax errors during translation in the SpyGlass product.

Language

Verilog, VHDL

Cause

Magma commands encountered with syntax errors during translation in the SpyGlass product.

NOTE: *This rule has been deprecated and will be removed in a future release.*

Message

```
Magma commands encountered with syntax error during translation
in Spyglass Constraints. For report please look into file
"TcM2PtNoTranslation" in <wdir>/spyglass_spysch/spyglass_sdc
directory
```

Severity

Error

SDC_214

Invalid SDC commands are specified in the input SDC constraints file.

Language

Verilog, VHDL

Cause

SpyGlass reports this warning if invalid SDC commands are specified in the input SDC constraints file.

SpyGlass also reports this warning when you specify shell commands, as shown in the following example:

```
echo [date]
echo [ls]
echo [history]
```

SpyGlass reports violation in the above case because `date`, `ls`, and `history` are UNIX shell commands and not Tcl commands. This means that the constraints may work in EDA tools that are shell-based, but you cannot be sure that these constraints would be tool independent. To achieve tool independence, ensure that you precede the shell commands with `exec`, as shown in the following example:

```
echo [exec date]
echo [exec ls]
```

NOTE: *The* `history` *command does not work even if you precede it with* `exec`*.*

Message

```
Invalid commands in input constraints file. For report please
look into file <file>
```

Severity

Warning

SDC_215

SDC commands were syntactically parsed but not used.

Language

Verilog, VHDL

Cause

SDC commands were syntactically parsed but not used by the SpyGlass tools such as constraints policy or SpyGlass-Physical.

Message

```
Commands syntactically parsed but not used. For report please
look into file <file>
```

Severity

Info

SDC_216

Commands not understood by the SpyGlass product but valid in other tools.

Language

Verilog, VHDL

Cause

Commands not understood by the SpyGlass product but valid in other tools.

Message

```
Commands not understood by SpyGlass Constraints solution but
valid in other tools. For report please look into file <file>
```

Severity

Warning

SDC_217

The direction of output port/pin specified in the SDC command is not correct.

Language

Verilog, VHDL

Cause

The direction of output port/pin specified in the SDC command is not correct.

Example

```
set_scan_signal test_scan_in -port SO -hookup [get_pins
io_pads/U1/into_pad_port1]
```

In the above mentioned command output pin `io_pads/U1/into_pad_port1` cannot be specified as `test_scan_in`; either use input/inout pin or specify `test_scan_out` for the `set_scan_signal` command.

Message

```
Output port/pin '<pin-name>' cannot be specified as '<type>' in
command <command>
```

Severity

Error

SDC_218

The direction of input port / pin specified in the SDC command is not correct.

Language

Verilog, VHDL

Cause

The direction of input port/pin specified in the SDC command is not correct.

Example

```
set_scan_signal test_scan_out -port SO -hookup [get_pins
io_pads/U1/into_pad_port2]
```

Input pin `io_pads/U1/into_pad_port2` cannot be specified as `test_scan_out`; either use output/inout pin or specify `test_scan_in` for the `set_scan_signal` command.

Message

```
Input port/pin '<pin-name>' cannot be specified as '<type>' in
command <command>
```

Severity

Error

SDC_219

The period value is not consistent with the waveform specification.

Language

Verilog, VHDL

Cause

The period value is not consistent with the waveform specification.

Example

```
create_clock
  -name CLK -period 10.0 -waveform {0 15}
  [get_ports clka]
```

The value of period (`10.0`) is not consistent with the waveform `{0 15}`; the waveform should be an even number of monotonically increasing values less than one period in duration.

Message

`Period value (<val>) is not consistent with waveform specification`

Severity

Error

SDC_220

Incorrect value is specified for the first edge in the -edges specification of the generated clock.

Language

Verilog, VHDL

Cause

Incorrect value is specified for the first edge in the `-edges` specification of the generated clock.

Example

```
create_generated_clock -edges {0.5 3 5} -source CLK [get_pins foo2]
```

The value (`0.5`) specified for the first edge is incorrect; it should be greater than 1.

Message

```
Incorrect value (<val>) specified for first edge in -edges specification for generated clock ″<clk-name>″
```

Severity

Error

SDC_221

The edge numbers must be in increasing order in the -edges specification of the generated clock

Language

Verilog, VHDL

Cause

The edge numbers must be in increasing order in the `-edges` specification of the generated clock

Example

```
create_generated_clock -edges {1 4 2} -source CLK [get_pins foo2]
```

The edges (`1 4 2`) should be in increasing order.

Message

```
Edge numbers must be in increasing order in the -edges specification for generated clock ”<clk-name>”
```

Severity

Error

SDC_222

range_value not specified

Language

Verilog, VHDL

Cause

The value of range is not specified for `set_critical_range` command

Example

```
set_critical_range 10.0 top
```

Value (`10.0`) for the range_value option should be specified for the `set_critical_range` command.

Message

```
range_value not specified
```

Severity

Error

SDC_223

designs not specified

Language

Verilog, VHDL

Cause

The name of the designs is not specified for `set_critical_range` command

Example

```
set_critical_range 10.0 top
```

Option `designs` (`top`) should be specified for the `set_critical_range` command.

Message

```
designs not specified
```

Severity

Error

SDC_224

max_library not specified

Language

Verilog, VHDL

Cause

The name of the library file is not specified for `set_min_library` command

Example

```
set_min_library LIB_WC_COM.db -min_version LIB_BC_COM.db
```

Library name `LIB_WC_COM.db` should be specified for the `set_min_library` command.

Message

```
max_library not specified
```

Severity

Error

SDC_225

Either -min_version or -none should have been specified

Language

Verilog, VHDL

Cause

One option from -min_version or -none should have been specified for `set_min_library` command.

Example

```
set_min_library LIB_WC_COM.db -min_version LIB_BC_COM.db -none
```

Options `-min_version` and `-none` cannot be combined; use one of them.

Message

```
either -min_version or -none should have been specified
```

Severity

Error

SDC_226

Commands ignored by spyglass policy due to file given by 'tc_ignored_commands' parameter

Language

Verilog, VHDL

Cause

Commands ignored by SpyGlass product due to file given by the `tc_ignored_commands` parameter.

Message

```
Commands ignored in spyglass constraints due to file given by
"set_parameter tc_ignored_commands" to ignore the invalid
commands. For report please look into file <file>
```

Severity

Info

SDC_227

Both -high and -low together cannot be specified

Language

Verilog, VHDL

Cause

Both `-high` and `-low` together cannot be specified for the specified command.

Example

```
set_clock_gating_check -setup 0.75 -hold 0.5 -high -low
[get_clocks CLK1]
```

Options `-high` and `-low` cannot be combined; use only one of them.

Message

```
both -high and -low together cannot be specified
```

Severity

Error

SDC_228

clock not specified with clock_fall and/or level_sensitive

Language

Verilog, VHDL

Cause

The option `clock` not specified with `clock_fall` and/or `level_sensitive` option.

Example

```
set_input_delay 2.5 -clock [get_clocks CLK1] -clock_fall
[all_inputs]
```

Option `-clock_fall/-level_sensitive` should be used with the `-clock` option for the `set_input_delay` command.

Message

```
clock not specified with clock_fall and/or level_sensitive
option in ″<command>″ command
```

Severity

Error

SDC_229

Object specified not a valid start point

Language

Verilog, VHDL

Cause

A combinational constraint passing through a point on the sequential path will introduce new reference points into the path.

This results in sequential paths being timed to and from those points, rather than between sequential cells as expected.

NOTE: *The SDC_229 and SDC_230 rules have severity as 'WARNING' whereas the SDC_35 and SDC_36 rules have severity as 'ERROR' .*

Example

Consider a sequential path from `I1/q` through `I2`, `I3`, `I4` (combinational gates) to `I5/d`. With no combinational constraint on this path it will be timed, as expected, as a sequential path. However, now suppose you set:

```
set_max_delay -from [get_pins I2/a] - to [get_pins I4/z]
```

This introduces new timing end-points at the specified pins and timing will now be checked:

```
from I1/q to I2/a

from I2/a to I4/z

from I4/z to I5/d
```

and the original (expected) sequential path from `I1/q` to `I5/d` will no longer be checked, possibly masking serious timing problems. In the degenerate case also:

```
set_max_delay -from [get_pins I1/q] - to [get_pins I5/d]
```

the sequential path check will be lost.

Message

```
Object: <obj-name> specified in constraint file is not a valid
start point. This introduces new timing points at the specified
```

pin

Severity

Warning

SDC_230

Objects specified not a valid end point

Language

Verilog, VHDL

Cause

A combinational constraint passing through a point on the sequential path will introduce new reference points into the path.

This results in sequential paths being timed to and from those points, rather than between sequential cells as expected.

NOTE: *The SDC_229 and SDC_230 rules have severity as 'WARNING' whereas the SDC_35 and SDC_36 rules have severity as 'ERROR' .*

Example

Consider a sequential path from `I1/q` through `I2`, `I3`, `I4` (combinational gates) to `I5/d`. With no combinational constraint on this path it will be timed, as expected, as a sequential path. However, now suppose we set:

```
set_max_delay -from [get_pins I2/a] - to [get_pins I4/z]
```

This introduces new timing end-points at the specified pins and timing will now be checked:

```
from I1/q to I2/a

from I2/a to I4/z

from I4/z to I5/d
```

and the original (expected) sequential path from `I1/q` to `I5/d` will no longer be checked, possibly masking serious timing problems. In the degenerate case also:

```
set_max_delay -from [get_pins I1/q] - to [get_pins I5/d]
```

the sequential path check will be lost.

Message

```
Object: <obj-name> specified in constraint file is not a valid
end point. This introduces new timing points at the specified
```

pin

Severity

Warning

SDC_231

Value of the derate not specified

Language

Verilog, VHDL

Cause

Value of the derate not specified for the `set_timing_derate` command.

Example

```
set_timing_derate -max -early 0.8 U1
```

Value (`0.8`) for the derate option should be specified for the `set_timing_derate` command.

Message

```
derate_value not specified
```

Severity

Error

SDC_232

Both -net_delay and object_list together cannot be specified

Language

Verilog, VHDL

Cause

Both `-net_delay` and `object_list` together cannot be specified for the `set_timing_derate` command.

Example

```
set_timing_derate -max -early 0.8 -net_delay U1
```

Options `-net_delay` and `object_list` (`U1`) cannot be combined; use one of them.

Message

```
Both -net_delay and object_list together cannot be specified
```

Severity

Error

SDC_233

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

Object specified in constraint file either does not exist or is not of required type, allowed types for the SDC command are leaf cells, instances or library cells.

Example

```
set_timing_derate -max -late 1.2 U1
```

The object `U1` either does not exist or not of required type; it should be leaf cells, instances, or library cells.

Message

```
Object: <obj-name> specified in constraint file either does not
exist or is not of required type (allowed types for <obj> =
leaf cells, instances or library cells)
```

Severity

Error

SDC_234

Library not specified with the ‘-library’ option for the command

Language

Verilog, VHDL

Cause

Library not specified with the `-library` option for the `set_wire_load_model` command; so using the `wire_load_model` from the library specified by using the `set_option enable_gateslib_autocompile yes` command in the project file.

Example

```
set_wire_load_model -name “10x10”
```

No library is specified with `-library` option, so using the `wire_load_model` from the library specified by using the `set_option enable_gateslib_autocompile yes` command in the project file.

Message

```
Library not specified with the ‘-library’ option, using
wire_load_model ”<wlm-name>” from the ”<lib-name>” library
specified by sglib/gateslib option
```

Severity

Info

SDC_235

No library specified with the ‘-library’ option

Language

Verilog, VHDL

Cause

No library specified with the `-library` option for `wire_load_model`.

Example

```
set_wire_load_model -name “10x10” -library lsi_10k
```

Option `-library` should be specified.

Message

```
No library specified with the ‘-library’ option for
wire_load_model
```

Severity

Error

SDC_236

Wire load model not found in default library

Language

Verilog, VHDL

Cause

Wire load model not found in default library specified by using the `set_option enable_gateslib_autocompile yes` command in the project file.

Example

```
set_wire_load_model -name “10x10”
```

The wire_load model `10x10` not found in the library specified by using the `set_option enable_gateslib_autocompile yes` command in the project file.

Message

```
Wire load model not found in default library specified by
sglib/gateslib option
```

Severity

Error

SDC_237

Do not use absolute path declaration when -library option is used.

Language

Verilog, VHDL

Cause

Do not use absolute path declaration when `-library` option is used.

Example

```
set_wire_load_model -name “10x10” -library /u/ur/lsi_10k
```

The name `/u/ur/lsi_10k` specified with `-library` name should not be absolute path.

Message

```
The object code library <lib-name> is specified using -library
option. Do not use absolute path declaration when -library
option is used
```

Severity

Warning

SDC_238

Cannot specify -true_threshold without -true for the get_timing_paths command.

Language

Verilog, VHDL

Cause

Option `-true_threshold` must be specified with `-true` option for the `get_timing_paths` command.

Example

```
get_timing_paths -true_threshold -nworsts 1000
```

The above command is incorrect; use `-true` option also.

Message

```
Cannot specify -true_threshold without -true
```

Severity

Error

SDC_239

Option -true specified with incorrect options.

Language

Verilog, VHDL

Cause

Option `-true` cannot be specified along with `-max_paths(>1)`, `-nworst(>1)`, `-lesser`, `-greater`, and `-delay_type` (path type other than max) options for the `get_timing_paths` command.

Example

```
get_timing_paths -true -nworsts 1000
```

Option `-true` cannot be used with greater than 1 value of `-nworsts` option.

Message

```
Option -true cannot be specified along with ‘<option-name>’
option(s)
```

Severity

Error

SDC_240

Options -early / -late cannot be specified without -source option.

Language

Verilog, VHDL

Cause

Options `-early`/`-late` cannot be specified without `-source` option for `set_clock_latency` command.

Example

```
set_clock_latency 0.8 -source -early [get_clocks CLK1]
```

The option `-early` is specified with `-source` option.

Message

```
Option(s) ‘<option-name>’ cannot be specified without -source
option
```

Severity

Error

SDC_241

If some option/attribute is given which is not valid with the currently set pt mode, then this error is flagged.

Language

Verilog, VHDL

Cause

When the `pt` parameter is set to `no`, SpyGlass behaves in sync with the DesignCompiler, and when set to yes/not set, then in the prime time mode. SpyGlass reports violation when incompatible options are specified.

Example

```
get_attribute -class port in1 name
```

The option `-class` is invalid when the `pt` parameter is set to `no`; to use this option set the `pt` parameter to `yes`.

Consider another example, as given below:

```
get_attribute -class port lsi_10k/IV/A capacitance
```

In the above example, the attribute, `capacitance`, is invalid when the `pt` parameter is set to `no`. To use this option, set the `pt` parameter to `yes` or use `pin_capacitance` instead of capacitance.

Message

```
'<Option | Attribute>' : '<Option-name | Attribute-name>' is
invalid when 'set_parameter pt' is set to '<yes | no>'
```

Severity

Error

SDC_242

Invalid value specified for the -class option.

Language

Verilog, VHDL

Cause

The specified value is not valid for the `-class` option of `get_attribute` command.

Valid values are `design`, `port`, `cell`, `pin`, `net`, `lib`, `lib_cell`, `lib_pin`, `clock`, and `generated_clock`.

Example

```
get_attribute -class rtl in1 ref_name
```

`rtl` is not a valid value for `-class` option.

Message

```
'<val-name>' is not a valid class name
```

Severity

Error

SDC_243

‘-from’, ‘-fall_from’, and ‘rise_from’ options cannot be specified together in the same get_timing_paths command.

Language

Verilog, VHDL

Cause

The `-from`, `-fall_from`, and `rise_from` options are mutually exclusive options of the `get_timing_paths` commands. Hence, you can specify only one of them at one time.

Example

```
get_timing_paths -true_threshold -nworsts 1000 -from {in1} -
fall_from {F1/Q}
```

Options `-from` and `-fall_from` cannot be specified together.

Message

```
Options '<option-name-list>' cannot be combined
```

Severity

Error

SDC_244

‘-to’, ‘-fall_to’, and ‘rise_to’ options cannot be specified together in the same get_timing_paths command.

Language

Verilog, VHDL

Cause

The `-to`, `-fall_to`, and `rise_to` options are mutually exclusive options of the `get_timing_paths` commands. Hence, you can specify only one of them at one time.

Example

```
get_timing_paths -true_threshold -nworsts 1000 -to {in1} -
fall_to {F1/Q}
```

Options `-to` and `-fall_to` cannot be specified together.

Message

```
Options '<option-name-list>' cannot be combined
```

Severity

Error

SDC_245

wire_load not specified

Language

Verilog, VHDL

Cause

The option `wire_load` should be specified for the `set_wire_load` command.

Example

```
set_wire_load “10x10” -library CORE9GPLL
```

Option `wire_load` (`10x10`) should be specified for the `set_wire_load` command.

Message

```
wire_load not specified
```

Severity

Error

SDC_246

Option -port_list cannot be combined with -mode, -cluster, or -selection_group options.

Language

Verilog, VHDL

Cause

The option `-port_list` cannot be combined with `-mode`, `-cluster`, or `-selection_group` options for the `set_wire_load` command.

Example

```
set_wire_load “10x10” -cluster CACHE -port_list {IN1}
```

Option `-port_list` cannot be combined with `-cluster` option.

Message

```
Option -port_list cannot be combined with ”<option-name-list>” option(s)
```

Severity

Error

SDC_247

Option -selection_group can only be specified with “enclosed” wire load mode.

Language

Verilog, VHDL

Cause

The option `-selection_group` can only be specified with `enclosed` wire load mode; other wire load mode {`top` `segmented`} cannot be specified for the `set_wire_load` command.

Example

```
set_wire_load
  “10x10” -selection_group 2layermetal -mode top
```

Option `-selection_group` cannot be specified with `top` wire load mode.

Message

```
Option -selection_group cannot be specified with ”<mode>” wire
load mode
```

Severity

Error

SDC_248

Both -min and -min_block_size options cannot be specified together

Language

Verilog, VHDL

Cause

Both `-min` and `-min_block_size` options cannot be specified together for the `set_wire_load` command.

Example

```
set_wire_load “10x10” -min_block_size 20 -min
```

Options `-min` and `-min_block_size` cannot be specified together; so use one of them.

Message

```
Both -min and -min_block_size options cannot be specified
together
```

Severity

Error

SDC_249

The specified references are not found in the design.

Language

Verilog, VHDL

Cause

The references specified in the constraint file are not found.

Hint

Check for name mismatch with the reference; it can be a typo mistake.

Example

```
set_dont_touch [get_references blk]
```

The reference `blk` should exist in the design.

Message

```
References: <ref-name> specified in constraint file not found
in design
```

Severity

Error

SDC_250

Cannot specify -master_clock without -add when the -pt option is set to “yes/supermode”.

Language

Verilog, VHDL

Cause

When `-pt` option is set to `yes` or supermode, SpyGlass behaves in sync with prime time.

The option `-master_clock` cannot be specified without the `-add` option for `create_generated_clock` command when the `-pt` option is set to `yes` or `supermode`; either set the option `-pt` to `no` or use the `-add` option.

Example

```
create_generated_clock -name GCLK -source clka [get_pins abc]
-master_clock CLK
```

Option `-master_clock` cannot be specified without `-add` option.

Message

```
Cannot specify -master_clock without -add
```

Severity

Error

SDC_251

Cannot specify -add without -master_clock when the -pt option is set to “yes/supermode”.

Language

Verilog, VHDL

Cause

When `-pt` option is set to `yes` or `supermode`, SpyGlass behaves in sync with prime time.

The option `-add` cannot be specified without the `-master_clock` option for `create_generated_clock` command when the `-pt` option is set to `yes` or `supermode`; either set the option `-pt` to `no` or use the `-master_clock` option.

Example

```
create_generated_clock -name GCLK -source clka [get_pins abc] -add
```

Option `-add` cannot be specified without `-master_clock` option.

Message

```
Cannot specify -add without -master_clock
```

Severity

Error

SDC_252

check_value not specified

Language

Verilog, VHDL

Cause

The value of the option `check_value` should be specified for the `set_annotated_check` command.

Example

```
set_annotated_check -setup 2.1 -from u1/ff12/CP -to u1/ff12/D
```

Value (`2.1`) for the `check_value` option should be specified for the `set_annotated_check` command.

Message

```
check_value not specified
```

Severity

Error

SDC_253

Option -from not specified

Language

Verilog, VHDL

Cause

The option `-from` should be specified for the command.

Example

```
set_annotated_check -setup 2.1 -from u1/ff12/CP -to u1/ff12/D
```

Option `-from` should be specified for the `set_annotated_check` command.

Message

```
Option -from not specified
```

Severity

Error

SDC_254

Option -to not specified

Language

Verilog, VHDL

Cause

The option `-to` should be specified for the command.

Example

```
set_annotated_check -setup 2.1 -from u1/ff12/CP -to u1/ff12/D
```

Option `-to` should be specified for the `set_annotated_check` command.

Message

```
Option -to not specified
```

Severity

Error

SDC_255

One of -setup, -hold, -recovery, -removal, -nochange_high, or -nochange_low must be specified

Language

Verilog, VHDL

Cause

Either of the options `-setup`, `-hold`, `-recovery`, `-removal`, `-nochange_high`, or `-nochange_low` must be specified for the `set_annotated_check` command to specify the type of the timing check.

Example

```
set_annotated_check 2.1 -from u1/ff12/CP -to u1/ff12/D
```

In the above mentioned command one of options {`-setup`, `-hold`, `-recovery`, `-removal`, `-nochange_high`, or `-nochange_low`} should be specified.

Message

```
One of -setup, -hold, -recovery, -removal, -nochange_high, or -
nochange_low must be specified
```

Severity

Error

SDC_256

Only one of -setup, -hold, -recovery, -removal, -nochange_high, or -nochange_low should be used

Language

Verilog, VHDL

Cause

Only one of the options from `-setup`, `-hold`, `-recovery`, `-removal`, `-nochange_high`, or `-nochange_low` should be specified for the `set_annotated_check` command.

Example

```
set_annotated_check 2.1 -from u1/ff12/CP -to u1/ff12/D -setup
-recovery
```

Options `-setup` and `-recovery` cannot be combined; use one of them.

Message

```
Only one of -setup, -hold, -recovery, -removal, -nochange_high,
or -nochange_low should be used
```

Severity

Error

SDC_257

Clock specified by -master_clock option does not exist in the fan-in of object specified by -source option.

Language

Verilog, VHDL

Cause

Clock specified by `-master_clock` option does not exist in the fan-in of object specified by `-source` option.

Example

```
create_generated_clock gclk1 -edges {1 3 5} -source clka
[get_pins foo2] -master_clock clk1
```

Clock `clk1` should be in the fan-in of the object `clka` specified by `-source` option.

Message

`Clock ″<clk-name>″ specified by -master_clock option does not exist in the fan-in of ″<obj-name>″ specified by -source option <reason>`

Where, *<reason>* indicates any of the following:

- All paths were traversed but the master clock was not found.
- Path traversal got blocked because some node had issues, such as the presence of a generated clock.

Severity

Error

SDC_258

Clock is not declared on the object (or no clock pin in its fanout, if parameter pt is set to yes)

Language

Verilog, VHDL

Cause

Clock is not declared on the object (or no clock pin in its fan-out, if the `pt` parameter is set to `yes` or `supermode`) specified for the `set_propagated_clock` command.

Example

```
set_propagated_clock [get_pins u1/A]
```

No clock is declared on the object (or no clock pin in its fan-out, if the `pt` parameter is set to `yes` or `supermode`) `u1/A`.

Message

```
Clock not declared on object '<obj-name>' (or no clock pin in
its fanout, if "set_parameter pt" is set to yes/supermode)
```

Severity

Error

SDC_259

Both -all and port_pin_list options cannot be specified together

Language

Verilog, VHDL

Cause

Both `-all` and *port_pin_list* options cannot be specified together for the remove_case_analysis command.

Example

```
remove_case_analysis in1 -all
```

Here, options `-all` and *port_pin_list* (`in1`) cannot be specified together; so use one of them.

Message

```
both -all and port_pin_list cannot be specified together
```

SDC_260

The clock edge must be in increasing order in the -waveform specification of the derive_clock

Language

Verilog, VHDL

Cause

The clock edge must be in increasing order in the -waveform specification of the `derive_clock`.

Example

```
derive_clock -period 10 -waveform {1 4 3 5}
```

Here, the edges `{1 4 3 5}` should be in increasing order.

Message

```
Edge specifications must be in increasing order in the
-waveform option of derive_clock command
```

Severity

Error

SDC_261

Cannot find the specified cell in the design.

Language

Verilog, VHDL

Cause

Cannot find the specified cell in the design.

Example

```
current_instance abc
```

Here, cell (`abc`) should exist in the `current_design`.

Message

```
Cannot find cell '<cell>' in design '<design>'
```

Severity

Error

SDC_262

Must specify one of these options: '-setup' or '-hold' or '-high' or '-low'.

Language

Verilog, VHDL

Cause

You must specify one of these options: `-setup` or `-hold` or `-high` or `-low` for the `set_clock_gating_check` command.

Example

```
set_clock_gating_check -setup 0.75 -hold 0.5 [get_design test]
```

One option must be specified from the {'-setup' or '-hold' or '-high' or '-low'} options for the `set_clock_gating_check` command.

Message

```
Must specify one of these options: '-setup' or '-hold' or '-high' or '-low'
```

Severity

Error

SDC_263

Cannot specify -add without -name

Language

Verilog, VHDL

Cause

The `-add` option cannot be specified without the `-name` option; the command should have both the options.

Example

```
create_clock -name C1 -period 10 [get_ports clk] -add
```

The `-add` option cannot be specified without the `-name` option.

Message

```
Cannot specify -add without -name
```

Severity

Error

SDC_264

Both -analysis_type and -object_list together cannot be specified

Language

Verilog, VHDL

Cause

Both `-analysis_type` and `-object_list` options cannot be specified together for the specified command.

Example

```
set_operating_conditions
  -min BCCOM -max WCCOM
  -analysis_type bc_wc -object_list F1/q
```

Here, the `-analysis_type` and `-object_list` options cannot be combined; use one of them.

Message

```
Both -analysis_type and -object_list together cannot be
specified
```

Severity

Error

SDC_265

target is not specified

Language

Verilog, VHDL

Cause

Target not specified for the `redirect` command.

Example

```
redirect t.out {set a xyz}
```

Here, target`{t.out}` is mandatory for the command.

Message

```
target is not specified
```

Severity

Error

SDC_266

command_string is not specified

Language

Verilog, VHDL

Cause

command_string is not specified for the `redirect` command.

Example

```
redirect t.out {set a xyz}
```

Here, *command_string* {"`set a xyz`"} is mandatory for the command.

Message

```
command_string is not specified
```

Severity

Error

SDC_267

Both -file and -variable together cannot be specified

Language

Verilog, VHDL

Cause

Both `-file` and `-variable` options together cannot be specified for the specified command.

Example

```
redirect -file -variable t.out {set a xyz}
```

Here, options `-file` and `-variable` cannot be combined; use one of them.

Message

```
Both -file and -variable together cannot be specified
```

Severity

Error

SDC_268

This rule has been deprecated.

With the new architecture, the SDC parser does not comment a line if a back slash (\) is specified at the end of its previous line.

Therefore, the *SDC_268* rule is not valid anymore as it reported scenarios in which a line got commented due to a back slash at the end of its previous line.

SDC_269

No timing arcs exist between -from pin and -to pin.

Language

Verilog, VHDL

Cause

No timing arcs exist between the pin specified with the `-from` option and `-to` option. The cell might not have been read or it might not be a Synopsis Liberty (.lib) cell. The `set_disable_timing` command supports liberty cells only. It is recommended to replace any HDL models with liberty models if `set_disable_timing` is to be used.

Example

```
set_disable_timing
  -from X -to Z [get_lib_cells lsi_10k/AN2]
```

Here, no timing arcs exist from pin `X` to pin `Z`.

```
set_disable_timing
  -from A -to B [get_cells AND]
```

Here, the cell `AND` is not read because it is not a liberty cell.

Message

```
There are no timing arcs from pin '<pin1>' to pin '<pin2>' on
cell '<cell1>'. The cell '<cell2>' might not have been read
```

Severity

Error

SDC_270

bus_name not specified

Language

Verilog, VHDL

Cause

The option `bus_name` should be specified for the `create_bus` command.

Example

```
create_bus {d1 d2} BUS1
```

Attribute name `bus_name` should be specified for the `get_attribute` command.

Message

```
bus_name not specified
```

Severity

Error

SDC_272

Options -regexp and -exact cannot be combined together for the command

Language

Verilog, VHDL

Cause

Options `-regexp` and `-exact` cannot be combined together, use one of them

Example

```
get_ports -regexp -exact cl*
```

Option `-regexp` and `-exact` cannot be combine together.

Message

```
Options -regexp and -exact cannot be combined together for the
"<command>" command
```

Severity

Error

SDC_273

Cannot specify -leaf option without -of_objects option for the get_pins command

Language

Verilog, VHDL

Cause

Cannot specify `-leaf` option without `-of_objects` option for the `get_pins` command, use `-of_objects` option.

Example

```
get_pins -leaf A1/a
```

Option `-leaf` cannot be used without `-of_objects` option

Message

```
Cannot specify -leaf option without -of_objects option for the
get_pins command
```

Severity

Warning

SDC_274

Cannot specify -reference_pin option with -network_latency_included or -source_latency_included option for the set_input_delay

Language

Verilog, VHDL

Cause

Cannot specify `-reference_pin` option with `-network_latency_included` or `-source_latency_included` option for the `set_input_delay`, don't use `-network_latency_included` or `-source_latency_included` option with `-reference_pin`

Example

```
set_input_delay 4 -clock clk1 -reference_pin clka -
network_latency_included -max -rise -add_delay data1
```

Option `-reference_pin` cannot be used with `-network_latency_included` option

Message

```
Cannot specify -reference_pin option with
-network_latency_included or -source_latency_included option
for the set_input_delay
```

Severity

Error

SDC_275

Objects specified in constraint file not found in design

Language

Verilog, VHDL

Cause

The object specified in the constraint file is not found in the design; the allowed types for the SDC command are port and leaf pin.

Example

```
set_input_delay 4 -clock clk1 -reference_pin clka -
network_latency_included -max -rise -add_delay data1
```

Object specified with `-reference_pin` option should be port or leaf pin.

Message

```
Object(s): <hierarchical-name-of-object> specified in
constraint file not found in design (allowed types for
<constraints-specified-in-input-SDC-file>
'<constraint-argument>' = port and leaf pin)
```

Severity

Error

SDC_276

No clock in the transitive fan-in of the pin/port given by reference_pin for the set_input_delay command/set_output_delay

Language

Verilog, VHDL

Cause

No clock in the transitive fan-in of the pin/port given by `reference_pin` for the `set_input_delay`/`set_output_delay` command.

Example

```
set_input_delay 15 -reference_pin F2/in1 -max -rise data1
```

In the above mentioned command, there is no clock in the transitive fan-in of F2/in1.

Message

```
No clock in the transitive fan-in of the pin/port given by
reference_pin for the set_input_delay command/set_output_delay
```

Severity

Error

SDC_277

Cannot find the clock specified by -clock option in the transitive fan-in of the pin/port given by reference_pin for the set_input_delay/set_output_delay command

Language

Verilog, VHDL

Cause

Cannot find the clock specified by `-clock` option in the transitive fan-in of the pin/port given by `reference_pin` for the `set_input_delay`/ `set_output_delay` command.

Example

```
set_input_delay 15 -clock clk1 -reference_pin F2/in1 -max -
rise data1
```

In the above mentioned command, clock `clk1` is not in the transitive fan-in of `F2/in1`.

Message

```
Cannot find the clock <clock-name> specified by -clock option
in the transitive fan-in of the pin/port given by reference_pin
for the set_input_delay command/set_output_delay
```

Severity

Error

SDC_278

Inappropriate class and collection given with get_attribute command

Language

Verilog, VHDL

Cause

Inappropriate class and collection given with `get_attribute` command

Example

```
get_attribute -class pin [get_ports data1] full_name
```

`data1` is a port, but class given as `pin`

Message

```
Inappropriate class ("<class>") for collection ("<collection>")
given with get_attribute command
```

Severity

Error

SDC_279

Incorrect collection type for the command

Language

Verilog, VHDL

Cause

Specified collection is incorrect for this command

Example

```
get_ports -of_objects [get_ports in]
```

Collection (of ports) is incorrect for the `-of_objects`, it should be net or pin

Message

```
Incorrect collection type (<type>) for option "<option>" in
"<command>" command
```

Severity

Error

SDC_280

Option -rule not specified with set_level_shifter_strategy command

Language

Verilog, VHDL

Cause

Option `-rule` not specified with `set_level_shifter_strategy` command, specify rule with using `set_level_shifter_strategy` command

Example

```
set_level_shifter_strategy -rule all
```

Option `-rule` should be specified for the `set_level_shifter_strategy` command.

Message

```
Option -rule not specified with set_level_shifter_strategy command
```

Severity

Error

SDC_281

cell_list not specified

Language

Verilog, VHDL

Cause

cell_list should be specified for the command

Example

```
create_voltage_area -name ABC u1
```

cell_list (`u1`) should be specified for `create_voltage_area` command.

Message

```
cell_list not specified for the "<command>" command
```

Severity

Error

SDC_282

Inappropriate format for the 'bus_naming_style' variable

Language

Verilog, VHDL

Cause

Inappropriate format for the `bus_naming_style` variable, value should contain appropriate characters, and separated by some character.

Example

```
set bus_naming_style %s_%d
```

Message

```
Inappropriate format for the 'bus_naming_style' variable
```

Severity

Error

SDC_283

cannot use attribute 'is_hierarchical' along with -tc_mantle='yes'.

Language

Verilog, VHDL

Cause

cannot use attribute `is_hierarchical` when the `tc_mantle` command is set to `yes`.

NOTE: *This rule has been deprecated and will be removed in a future release.*

Example

```
set a [get_pins -hier * -filter "is_hierarchical == yes"]
```

The above command will result in a violation message because the attribute `is_hierarchical` cannot be used when the `tc_mantle` command is set to `yes`.

Message

```
cannot use attribute 'is_hierarchical' along with -
tc_mantle='yes'
```

Severity

Error

SDC_284

Wildcard not allowed in context of vectors when -tc_mantle='yes'.

Language

Verilog, VHDL

Cause

Wildcard characters not allowed in context of vectors when the `tc_mantle` command is set to `yes`.

NOTE: *This rule has been deprecated and will be removed in a future release.*

Example

```
puts [get_ports in1*]
```

where, `in1` is a vector port in the design

The above command will result in a violation message because wildcard characters cannot be used in context of vectors when `tc_mantle` command is set to `yes`.

Message

```
Wildcard not allowed in context of vectors when -
tc_mantle='yes'.
```

Severity

Error

SDC_286

set_clock_groups overwrites false paths for the clocks

Language

Verilog, VHDL

Cause

`set_clock_groups` overwrites false paths for the clocks.

Examples

The following examples illustrates when the SDC_286 rule reports violation messages.

Example 1

In the following example, one `set_false_path` has been provided after `set_clock_groups`. The SDC_286 reports a Warning violation message, which indicates that both `set_false_path` and `set_clock_groups` have been defined between the clocks and the tool will retain `set_clock_groups`.

```
set_clock_groups -asynchronous -group {clk1} -group {clk2}
set_false_path -from clk1 -to clk2
```

Example 2

In the following example, two set_false_paths have been provided after `set_clock_groups`. The SDC_286 rule reports a Warning violation message for each `set_false_path`. Each violation message because both `set_false_path` and `set_clock_groups` have been defined between the clocks and the tool retains `set_clock_groups`.

```
set_clock_groups -asynchronous -group {clk1} -group {clk2}
set_false_path -from clk1 -to clk2
set_false_path -from clk2 -to clk1
```

Message

```
set_clock_groups overwrites false paths for the clocks
```

'<clock-name1>' & '<clock-name2>' (file:<file-name> line:<line-number>)

Severity

Warning

SDC_287

Clock exists in more than one group

Language

Verilog, VHDL

Cause

In the `set_clock_groups` command, a clock exists in more than one group.

Example

```
set_clock_groups -asy -group {c1 c2} -group {c1 c3}
```

The above command results in violation because clock `c1` cannot be specified for more than one groups for `set_clock_groups` command.

Message

```
Clock '<clock-name>' exists in more than one group
```

Severity

Error

SDC_288

Clock groups with these clocks are already set

Language

Verilog, VHDL

Cause

Clock groups with the clocks given in the `set_clock_groups` command are already set.

Example

```
set_clock_groups -asy -group c1 -group c2
set_clock_groups -asy -allow_paths -group c1 -group c2
```

In this example, the second `set_clock_groups` command results in violation because clock groups with `c1` and `c2` clocks are already set in the previous line.

Message

```
Clock groups with '<clock-name1>' and '<clock-name2>' clocks
are already set (file:<file-name> line:<line-number>)
```

Severity

Warning

SDC_289

The specified from-pin is not the input/inout pin of the specified cell.

Language

Verilog, VHDL

Cause

The specified from-pin should be the input/inout pin of the cell.

Example

```
set_driving_cell -lib_cell AND2 -clock CLK1 -clock_fall {IN1}
-pin Z -from_pin A.
```

In this example, the pin `A` specified as `-from_pin` option should be an input/inout pin of the `AND2` cell.

Message

```
Pin: <pin> is not the input/inout pin of the specified cell
```

Severity

Error

SDC_290

The specified pin of hierarchical cell resolved to valid start point.

Language

Verilog, VHDL

Cause

The specified pin of hierarchical cell resolved to a unique valid start point. This corresponds to clock pin of a connected sequential cell.

Example

```
set_false_path -from "u_from/out1"
```

In this example, `set_false_path` defined at clock pin, `u_from/out1`, may be resolved.

Message

```
The specified pin of hierarchical cell "<cell>" resolved to
valid start point "<start-point>"
```

Severity

Info

SDC_291

The specified pin of hierarchical cell resolved to valid endpoint.

Language

Verilog, VHDL

Cause

The specified pin of hierarchical cell resolved to a unique valid endpoint. This corresponds to input pin (except clock and enable pins) of a sequential cell.

Example

```
set_false_path -from "u_from/out1"
```

In this example, `set_false_path` defined at clock pin, `u_from/out1`, may be resolved.

Message

```
The specified pin of hierarchical cell "<cell>" resolved to
valid end point "<end-point>"
```

Severity

Info

SDC_292

The specified hierarchical cell resolved to valid through-point(s).

Language

Verilog, VHDL

Cause

The specified hierarchical cell resolved to valid through-point(s). This corresponds to output pins of the specified hierarchical cells.

Example

```
set_false_path -from "u_from/out1"
```

In this example, `set_false_path` defined at clock pin, `u_from/out1`, may be resolved.

Message

```
The specified hierarchical cell "<cell>" resolved to valid
through point(s)
```

Severity

Info

SDC_293

One of -from, -rise_from, or -fall_from options should be given.

Language

Verilog, VHDL

Cause

One of the `-from`, `-rise_from`, or `-fall_from` options should be given for the `set_data_check` command.

Example

```
set_data_check 0.123 -hold -to [get_ports out]
```

In this example, one of the `-from`, `-rise_from`, or `-fall_from` options should be specified.

Message

```
One of -from, -rise_from, or -fall_from should be given
```

Severity

Error

SDC_294

One of -to, -rise_to, or -fall_to options should be given.

Language

Verilog, VHDL

Cause

One of the `-to`, `-rise_to`, or `-fall_to` options should be given for the `set_data_check` command.

Example

```
set_data_check 0.123 -setup -from [get_ports in]
```

In this example, one of the `-to`, `-rise_to`, or `-fall_to` options should be specified.

Message

```
One of -to, -rise_to, or -fall_to should be given
```

Severity

Error

SDC_295

Check_value not specified for the set_data_check command.

Language

Verilog, VHDL

Cause

The `check_value` specifying the setup/hold time for the `set_data_check` command must be specified.

Example

```
set_data_check -setup -from [get_ports in] -to [get_ports out]
```

In this example, the `check_value` option must be specified.

Message

```
Check_value not specified for the set_data_check command
```

Severity

Error

SDC_296

Option -edge_shift should be used with option -edges.

Language

Verilog, VHDL

Cause

The `-edge_shift` option is populated only when specified with the `-edges` option for the `create_generated_clock` command. Otherwise, the `-edge_shift` option is ignored.

Example

```
create_generated_command -edge_shift {0 2.5 0 2.5 0} -
divide_by 10 -source [get_pins F1/CP] [get_pins F1/CP]
```

In this example, the `-edge_shift` option will be ignored because it is not being used with the `-edges` option.

Message

```
Option -edge_shift should be used with option -edges
```

Severity

Warning

SDC_297

Inout port specified in port list for set_fanout_load command.

Language

Verilog, VHDL

Cause

Inout port is specified in the port list of the `set_fanout_load` command. An output port should be specified in the port list of the `set_fanout_command`.

Example

```
set_fanout_load 123.4 [get_ports INOUT]
```

In this example, `INOUT` is an inout port.

Message

```
Inout port '<port>' specified in port list for set_fanout_load
command
```

Severity

Warning

SDC_298

The specified mode does not exist

Language

Verilog, VHDL

Cause

The mode specified in the command was not created using the `define_design_mode_group` command.

Example

```
map_design_mode DM123 READ [get_cells MEM_CELLS*]
```

In the above command, the design mode, `DM123`, should be defined with a separate `define_design_mode_group` command to avoid reporting of this error.

Message

```
Design mode "<design-mode>" specified in constraint file does
not exist
```

Severity

Error

SDC_299

The specified mode already exists

Language

Verilog, VHDL

Cause

The specified mode has already been created by the `define_design_mode_group` command.

Example

```
define_design_mode_group {DM1 DM2 DM3}
```

In the above command, if the design mode, `DM1`, has already been created previously, this violation will get reported.

Message

```
Design mode "<design-mode>" already exists
```

Severity

Error

SDC_300

The specified cell-mode is not valid for given cell instance.

Language

Verilog, VHDL

Cause

The cell-mode specified in mode list of the `map_design_mode` command does not exist for a cell instance.

Example

```
map_design_mode DM1 {MODE_NOTDEF} [get_cells F1]
```

In the above command, the design mode, `MODE_NOTDEF`, is not defined for cell, `F1`.

Message

```
Cell-mode "<cell-mode>" specified for instance "<instance>"
does not exist
```

Severity

Warning

SDC_301

Couldn't change working directory.

Language

Verilog, VHDL

Cause

The target directory for the `cd` command was not found.

Example

```
cd temp/
```

In this example, `temp/` directory does not exist.

Message

```
Couldn't change working directory to '<directory>': no such
directory
```

Severity

Error

SDC_302

Incorrect value for parameter -type specified. Valid values are "design" and "cell".

Language

Verilog, VHDL

Cause

Incorrect value for the `-type` parameter is specified. Valid values are `design` and `cell`.

Example

```
set_mode -type abc [get_cells F1/q]
```

In the above command, value `abc` cannot be specified with `-type`.

Message

```
Incorrect value for parameter -type for '<command>' command.
Valid values are design|cell
```

Severity

Error

SDC_303

Depending on different tools and versions, the priority of -combinational w.r.t. the three options '-multiply_by', '-edges' and '-divide_by' may change.

Language

Verilog, VHDL

Cause

The use of `-combinational` option with any of the `-multiply_by`, `-edges` or `-divide_by` options for `create_generated_clock` constraint may be interpreted differently by different tools and versions. In SpyGlass, the waveform characteristics of the generated clock is determined by the `-multiply_by`, `-edges` or `-divide_by` option and not by the `-combinational` option.

Example

```
create_generated_clock -name gclk1 -combinational -divide_by 3 [get_pins abc]
```

In this example, the time period of the generated clock will be three times that of the source clock that is, the factor specified by the `-divide_by` option. There will be no effect of the `-combinational` option in deciding the waveform characteristics

Message

```
Depending on different tools and versions, the priority of -combinational w.r.t. the three options '-multiply_by', '-edges' and '-divide_by' may change.
```

Severity

Info

SDC_304

Mode value being set by set_case_analysis would be overridden by value set through set_mode

Language

Verilog, VHDL

Cause

Mode value being set by `set_case_analysis` would be overridden by value set through `set_mode`.

Example

```
set_mode -type cell {MODE_1} F1
set_case_analysis 1 F1/enable
```

In the above example, `set_case_analysis` is not allowed to set a mode that is different from `MODE_1` on instance `F1`.

Message

```
Mode value being set by set_case_analysis would be overridden
by value set through set_mode
```

Severity

Warning

SDC_305

SpyGlass currently does not support all the attributes with get_attribute command.

Language

Verilog, VHDL

Cause

This message indicates that the attribute specified with the `get_attribute` command is currently not supported/understood by SpyGlass.

For complete list of supported attributes, see the *SpyGlass Constraints Rules Reference*.

Example

```
get_attribute out1 load
```

In this example, the `load` attribute on ports is currently not understood by SpyGlass.

Message

```
SpyGlass currently does not support attribute
'<attribute-name>' for <object-type> '<object-name>'
```

Where, *<object-type>* is the type of object, such as pin, net, cell, or clock for which the reported attribute is referred.

Severity

Info

SDC_306

Although Magma Tool supports "force timing break" without any argument, SpyGlass requires at least one option to be specified.

Language

Verilog, VHDL

Cause

As per the Magma tool, all the options are optional for the "force timing break" command. However, if you do not specify any option, the command is not be populated for SpyGlass. Hence, it is required to specify at least `-from` or `-to` option.

NOTE: *This rule has been deprecated and will be removed in a future release.*

Message

```
Although Magma Tool supports "force timing break" without any
argument, SpyGlass requires at least one option to be specified
```

Severity

Error

SDC_307

-rise and -fall_to options cannot be specified together

Language

Verilog, VHDL

Cause

The `-rise` and `-fall_to` options cannot be specified together. If you specify them together, the condition on end points becomes contradictory, and the command is not populated. Change the `-fall_to` option to either `-to` or `-rise_to` option.

Example

```
set_false_path -rise -from in1 -fall_to out1
```

In this example, `-rise` and `-fall_to` options should not be specified together.

Message

```
The options -rise and -fall_to cannot be specified together
```

Severity

Warning

SDC_308

-fall and -rise_to options cannot be specified together

Language

Verilog, VHDL

Cause

The `-fall` and `-rise_to` options cannot be specified together. If you specify them together, the condition on end points becomes contradictory, and the command is not populated. Change the `-rise_to` option to either `-to` or `-fall_to` option.

Example

```
set_false_path -fall -from in1 -rise_to out1
```

In this example, `-fall` and `-rise_to` options should not be specified together.

Message

```
The options -fall and -rise_to cannot be specified together
```

Severity

Warning

SDC_309

Incorrect argument when parameter pt is set to yes/supermode

Language

Verilog, VHDL

Cause

The given argument is incorrect for the command specified in the constraint file when the `pt` parameter is set to `yes` or `supermode`.

Example

```
set_dont_touch_network clk -no_propagate
```

In this example, the `-no_propagate` option is invalid when the `pt` parameter is set to `yes` or `supermode`.

Message

```
incorrect argument "<argument>" for "<command>" while
'set_parameter pt' is set to yes/supermode
```

Severity

Error

SDC_310

Option not given

Language

Verilog, VHDL

Cause

Option should be given for the command.

Example

```
set_dft_signal -view spec -port in1
```

In this example, the `-type` option is mandatory for the `set_dft_signal` command.

Message

```
"<option>" option not given
```

Severity

Error

SDC_312

The given option(s) needs to be specified with another option(s)

Language

Verilog, VHDL

Cause

For a given command, option(s) <*option1*> is/are mandatory with option(s) <*option2*>.

For example, hookup pins need to be specified with the `-hookup_sense` option for the `set_dft_signal` command.

Example

```
set_dft_signal -view existing_dft -type Reset -hookup_sense
inverted
```

In this example, option `hookup_pin` is mandatory with the `-hookup_sense` option.

```
set_scan_group grp1 -access {ScanDataIn out1}
-include_elements {u_to} -serial_routed false
```

In this example, option `-serial_routed true` is mandatory with the `-access` option.

Message

```
Option(s) "<option1-name(s)>" is/are mandatory with option(s)
"<option2-name(s)>"
```

Severity

Error

SDC_313

Both these options cannot be specified together

Language

Verilog, VHDL

Cause

The `-timing` option cannot be used with the `-view spec` for the `set_dft_signal` command. You should specify it with `-view existing_dft`.

Example

```
set_dft_signal -view spec -port in1 -type Reset -timing {10 20}
```

In this example, option `-timing` can be specified with `-view existing_dft` only for the `set_dft_signal` command.

Message

```
Both these options <option1> & <option2> cannot be specified together
```

Severity

Error

SDC_314

Option "-period" can come with only the option "-type RefClock" for the set_dft_signal command.

Language

Verilog, VHDL

Cause

The `-period` option can be specified only with the signal type, `RefClock`, for the `set_dft_signal` command.

Example

```
set_dft_signal -view existing_dft -port in1 -type Reset -
timing {10 20} -period 10
```

In this example, option `-period` can be specified with option `-type RefClock` only for the `set_dft_signal` command.

Message

```
-period option can come with only -type RefClock
```

Severity

Error

SDC_316

name of the scan group not specified

Language

Verilog, VHDL

Cause

The `scan_group_name` option should be specified for the `set_scan_group` command.

Example

`set_scan_group`

In this example, the `scan_group_name` option should be specified for the `set_scan_group` command.

`remove_scan_group`

In this example, the `scan_group_name` option should be specified for the `remove_scan_group` command.

Message

`name of the scan group not specified`

Severity

Error

SDC_317

-include_elements not specified

Language

Verilog, VHDL

Cause

The `-include_elements` option should be specified and is mandatory for the `set_scan_group` command.

Example

```
set_scan_group grp1
```

In this example, the `-include_elements` option should be specified for the `set_scan_group` command.

Message

```
-include_elements not specified
```

Severity

Error

SDC_318

-access should not be specified with -serial_routed false

Language

Verilog, VHDL

Cause

The `-access` option should be specified only when `-serial_routed` is set to `true` for the `set_scan_group` command.

Example

```
set_scan_group grp1 -access {ScanDataIn out1}
-include_elements {u_to} -serial_routed false
```

In this example, the `-access` option should be specified only with `-serial_routed true` for the `set_scan_group` command.

The support for the `-access` option is available only when `-serial_routed` is set to `true` for the `set_scan_group` command. You should either provide `-serial_routed true` or remove the `-access` option.

Message

```
-access should not be specified with -serial_routed false
```

Severity

Error

SDC_319

scan_group_name not specified

Language

Verilog, VHDL

Cause

The `scan_group_name` option should be specified for the `remove_scan_group` command.

Example

```
remove_scan_group
```

In this example, the `scan_group_name` option should be specified for the `remove_scan_group` command.

Message

```
scan_group_name not specified
```

Severity

Error

SDC_320

The command cannot be used when the "set_parameter pt" is set to yes.

Language

Verilog, VHDL

Cause

The `set_signal_type` command cannot be used with the `pt` parameter set to `yes`.

Message

```
The command "set_signal_type" cannot be used with the
set_parameter pt set to "yes"
```

Severity

Error

SDC_321

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type. The allowed types for the SDC command are cell.

Example

```
set_scan_group grp1 -access {ScanDataIn out1}
-include_elements {MB1} -serial_routed true
```

In this example, the object, `MB1`, either does not exist or is not of the required type. It should be a cell.

Message

```
Object: <object> specified in constraint file either does not
exist or is not of required type (allowed types = Cell)
```

Severity

Error

SDC_322

Access pins cannot be associated with more than one signal type

Language

Verilog, VHDL

Cause

Segment access list has multiple signal types for the same hookup point. All but the first is discarded.

Example

```
set_scan_group grp1 -access {ScanDataIn out1 ScanClock out1}
-include_elements {u_to} -serial_routed true
```

In this example, either the ScanDataIn type or the ScanClock type should be specified for the hookup point, `out1`, in the `set_scan_group` command.

Message

```
Access pin "<pin>" cannot be associated with more than one
signal type. All but the first is discarded
```

Severity

Warning

SDC_323

Scan groups cannot have more than one ScanDataIn or ScanDataOut access pin.

Language

Verilog, VHDL

Cause

Segment access list has multiple hookup points for the ScanDataIn/ ScanDataOut signal type. All but the first is discarded.

Example

```
set_scan_group grp1 -access {ScanDataIn out1 ScanDataIn out2}
-include_elements {u_to} -serial_routed true
```

In this example, the hookup point, `out1` or `out2`, should be specified for the ScanDataIn type in the `set_scan_group` command.

Message

```
Scan groups cannot have more than one "<ScanDataIn |
ScanDataOut>" access pin. All but the first is discarded
```

Severity

Warning

SDC_324

Object specified in constraint file either does not exist or is not of required type

Language

Verilog, VHDL

Cause

The object specified in the constraint file either does not exist or is not of the required type. The allowed types for the SDC command are pins.

Example

```
set_dft_signal -view existing_dft -type Reset -hookup_pin in1
-port {in1 clk}
```

In this example, the port `in1` cannot be specified as `hookup_pin`.

Message

```
Object: "<object>" specified in constraint file either does not
exist or is not of required type (allowed types are pins)
```

Severity

Error

SDC_325

The cells cannot belong to more than one scan group

Language

Verilog, VHDL

Cause

Scan groups cannot have common cells.

Example

```
set_scan_group grp1 -access {ScanDataIn out1}
-include_elements {u_to} -serial_routed true
```

```
set_scan_group grp2 -access {ScanDataIn out1}
-include_elements {u_to} -serial_routed true
```

In this example, scan groups, `grp2` and `grp1`, have common elements `u_to`.

Message

```
The Scan Groups "<group1>" and "<group2>" cannot have common
element "<element>"
```

Severity

Error

SDC_326

The direction of port specified in the SDC command is not consistent with the signal type.

Language

Verilog, VHDL

Cause

The direction of port/pin specified in the SDC command is not correct since it is not consistent with signal type.

Example

```
set_dft_signal -view spec -port in1 -type ScanDataOut
```

In this example, port `in1` cannot be specified as ScanDataOut. Either use output/inout pin or specify `test_scan_in` for the `set_scan_signal` command.

Message

```
The port "<port>" with direction "<direction>" cannot be
specified as "<ScanDataOut>" in command set_dft_signal
```

Severity

Error

SDC_327

The given signal type is not a valid scan_signal_type

Language

Verilog, VHDL

Cause

The `-access` option must be a pair of `scan_signal_type` and `port_pin` {`scan_signal_type` port/pin} in the `set_scan_group` command. The value specified in this command is not one of the valid `scan_signal_type` such as ScanDataIn/ScanDataOut/ ScanMasterClock/ScanSlaveClock/ScanEnable/ScanClock.

Example

```
set_scan_group grp1 -access {out1 in1 ScanDataIn in1
ScanDataIn in1} -include_elements {u_to} -serial_routed true
```

In this example, `out1` is not a valid `scan_signal_type`.

Message

```
"<signal-type>" is not a valid scan signal type
```

Severity

Warning

SDC_328

nets can be specified only with -no_propagate option in the command set_ideal_network

Language

Verilog, VHDL

Cause

The `set_ideal_network` command accepts nets only when the `-no_propagate` option is specified.

Example

```
set_ideal_network [get_nets temp]
```

In this example, nets can be specified only with the `-no_propagate` option in the `set_ideal_network` command.

Message

```
nets can be specified only with -no_propagate option in the
command set_ideal_network
```

Severity

Warning

SDC_329

The given SpyGlass DFT solution signal in the remove_dft_signal command does not exist

Language

Verilog, VHDL

Cause

The `remove_dft_signal` is trying to remove a command which was not specified using `set_dft_signal` command earlier. The `remove_dft_signal` command only removes the earlier `set_dft_signal` commands in the same view and on same ports.

Example

```
set_dft_signal -port {in1} -type Reset
remove_dft_signal -view spec -port {clk}
```

In this example, `set_dft_signal` not found for the object, `clk`, for the corresponding `remove_dft_signal` specified.

Message

```
set_dft_signal not found for the object "<object>" for the
corresponding remove_dft_signal specified
```

Severity

Error

SDC_330

The given scan_group in the remove_scan_group command does not exist

Language

Verilog, VHDL

Cause

Scan group specified in the `remove_scan_group` command does not exist. Hence `remove_scan_group` does not work.

Example

```
set_scan_group grp1 -access {ScanDataIn out1}
-include_elements {u_to} -serial_routed true
remove_scan_group grp2
```

In this example, scan group `grp2` specified in the `remove_scan_group` command does not exist.

Message

```
scan_group "<group>" does not exist hence remove_scan_group
will not work
```

Severity

Error

SDC_331

The given scan_group overwrites the earlier specification of set_scan_group

Language

Verilog, VHDL

Cause

Scan group already exists. Therefore, overwriting with the new command.

Example

```
set_scan_group grp1 -access {ScanDataIn out1}
-include_elements {u_to} -serial_routed true
set_scan_group grp1 -access {ScanDataIn in1}
-include_elements {u_to} -serial_routed true
```

In this example, scan group `grp1` is specified twice, and therefore, latest one overwrites the earlier specification.

Message

```
The given scan group "<group>" overwrites the earlier
specification of set_scan_group
```

Severity

Warning

SDC_332

The given signal type is not a valid signal type

Language

Verilog, VHDL

Cause

The `-type` option must be followed by a valid signal type such as Reset/Constant/TestMode/TestData/ScanDataIn/ScanDataOut/ScanMasterClock/ScanSlaveClock/ScanEnable/InOutControl/MasterClock/SlaveClock/Oscillator/RefClock/ScanClock/tdi/tdo/tck/tms/trst/bsd_shift_en/bsd_capture_en/bsd_capture_dr/bsd_update_en/bsd_update_dr/capture_clk/update_clk/inst_enable/bist_enable/bist_clk/bsd_reset.

Example

```
set_dft_signal -view spec -port in1 -type ScanClock1
```

In this example, `ScanClock1` is not a valid signal type.

Message

```
"<signal-type>" is not a valid signal type
```

Severity

Warning

SDC_334

name of the rule not specified

Language

Verilog, VHDL

Cause

SpyGlass flags this error when the option name_rules is not given. It is mandatory for the define_name_rules command.

Example

```
define_name_rules
```

This is an invalid command. The name of the rule is mandatory.

```
define_name_rules verilog_rule_temp
```

This is a valid command. The name of the rule is specified in this case.

Message

```
name of the rule not specified
```

Severity

Error

SDC_335

-rules/-rule not specified

Language

Verilog, VHDL

Cause

SpyGlass flags this error when the `-rules/-rule` option is not specified. It is mandatory for the `change_names` command.

Example

```
change_names
```

This is an invalid command. It is mandatory to specify the `-rules` option.

```
change_names -rules verilog_rule_temp
```

This is a valid command, as the `-rules` option is specified.

```
change_names -rule verilog_rule_temp
```

This is a valid command, as the `-rule` option is specified.

Message

```
-rules/-rule not specified
```

Severity

Error

SDC_336

Value for -rule/-rules not defined earlier

Language

Verilog, VHDL

Cause

The change_names command should be followed with `-rules/-rule` <*rule-name*> where <*rule-name*> is already defined with the `define_name_rules` command.

Example

```
change_names -rule vhdl_old
```

This is an invalid command. Here, the value for the `-rule` option should be defined earlier.

```
define_name_rules vhdl_old -type port
change_names -rule vhdl_old
```

This is a valid command.

Message

```
Rule "<rule-name>" not defined earlier with define_name_rules
command
```

Severity

Error

SDC_337

The given define_name_rules <ruleName> overwrites the earlier specification of define_name_rules for the type <typeName>

Language

Verilog, VHDL

Cause

The define_name_rules <rule-name> specification already exists. Hence, it is overwritten with the given define_name_rules command.

Example

```
define_name_rules verilog_out -type port -first_restricted
"_ 0-8" -prefix "D"
```

```
define_name_rules verilog_out -type port -first_restricted
"_ 0-8" -prefix "E"
```

Here, the latest define_name_rules verilog_out specification overwrites the earlier specification of define_name_rules for port.

Message

```
The given define_name_rules "<rule-name>" overwrites the
earlier specification of define_name_rules for "<type-name>
```

Severity

Warning

SDC_338

name_rules <ruleName1> and <ruleName2> applied across sdc_data are different and hence not allowed.

Language

Verilog, VHDL

Cause

All the `sdc_data` constraints are parsed with a single rule name only. It can be either specified in a single constraint (in multiple SDC files belonging to that constraint, in which the change names will be applied in the same order in which they appear) or repeatedly specified identically in multiple constraints.

Example

Consider the following specifications in different SDC files:

In bar0.sdc:

```
// NO define_name_rules is specified
```

In bar.sdc:

```
define_name_rules rule1 -allowed "c"
change_names -rules rule1
```

In bar1.sdc:

```
define_name_rules rule1 -allowed "a"
change_names -rules rule1
```

In bar2.sdc:

```
define_name_rules rule2 -allowed "b"
```

Now, consider the following cases:

Case1

```
sdc_data -file bar0.sdc
```

```
sdc_data -file bar.sdc bar1.sdc
sdc_data -file bar2.sdc
```

SpyGlass flags an error in this case as there is a conflict between `rule1` and `rule2`.

Case2

```
sdc_data -file bar0.sdc
sdc_data -file bar2.sdc
```

This is a valid specification. In this case, first `Rule0` will be applied and then `Rule2` thereafter.

Case3

```
sdc_data -file bar.sdc bar1.sdc
sdc_data -file bar2.sdc
```

SpyGlass flags an error in this case as there is a conflict between `rule1` and `rule2`.

Message

```
name_rules "<rule-name1>" and "<rule-name2>" applied across
schemas are different and hence not allowed
```

Severity

Error

SDC_339

Unrecognized option, SpyGlass supports only selective options for define_name_rules and change_names

Language

Verilog, VHDL

Cause

The options supported for `define_name_rules` and `change_names` are shown in the following table:

define_name_rules	-allowed	-replacement_char
	-prefix	-type
	-last_restricted	-max_length
	-first_restricted	-target_bus_naming_style
	-map	
change_names	-rules	-hierarchy

Example

```
define_name_rules vhdl_old -case_insensitive
```

The `-case_insensitive` option is not supported in the above command.

```
change_names -rules vhdl_old -verbose
```

The `-verbose` option is not supported in the above command.

```
define_name_rules vhdl_old -allowed "a-z"
```

The `-allowed` option is supported in the above command.

```
change_names -rules vhdl_old
```

The -rules option is supported in the above command.

Message

Unrecognized option, SpyGlass supports only selective options for define_name_rules and change_names: for define_name_rules { -allowed, -replacement_char, -prefix,-type,-first_restricted,-max_length, -target_bus_naming_style, -map } and for change_names {-rules and -hierarchy }

Severity

Info

SDC_340

Inappropriate value for the switch '-target_bus_naming_style' in the command define_name_rules

Language

Verilog, VHDL

Cause

SpyGlass flags this error when an inappropriate value is specified for the `-target_bus_naming_style` switch in the define_name_rules command. Value should contain '%s' and '%d', separated by some characters.

Example

```
define_name_rules verilog_out_name_rule
-target_bus_naming_style %s%d
```

This is an invalid command.

```
define_name_rules verilog_out_name_rule
-target_bus_naming_style %s_%d
```

This is a valid command.

Message

```
Inappropriate value for the switch '-target_bus_naming_style'
in the command define_name_rules
```

Severity

Error

SDC_341

Commands supported by SpyGlass constraints in latest version but not supported in the currently set older sdc_version.

Language

Verilog, VHDL

Cause

SpyGlass flags this message if the commands that are supported by SpyGlass constraints in latest version are not supported in the currently set older `sdc_version`. In such cases, the support for these commands may not be present in one or more older versions.

Message

```
Commands supported by spyglass constraints in latest version
but not supported in the currently set older sdc_version. For
report please look into file <file-name>
```

Severity

Info

SDC_342

name of the scan path not specified

Language

Verilog, VHDL

Cause

SpyGlass flags this message if the name of the scan path is not specified.

Example

```
set_scan_path -include_elements F1
```

Message

```
Name of scan path not specified which is a mandatory option
```

Severity

Error

SDC_343

Pin specified in hookup pin List is not a valid hookup pin.

Language

Verilog, VHDL

Cause

SpyGlass flags this message if the pin specified in the list of pins is not a valid hookup pin. A pin in the list specified by using the `-hookup` option is a valid hookup pin only if the following conditions hold true:

- It is specified as a hookup pin by using the `set_dft_signal` command with view spec.
- In `set_scan_path` command, the view given is spec only.

Example

```
set_dft_signal -type Reset -hookup_pin F1/cp
-port {in1} -hookup_sense inverted
```

```
set_dft_signal -type Reset -hookup_pin F1/cp
-port {in1} -hookup_sense inverted -view existing_dft
```

```
set_scan_path chain1 -class bsd -hookup {F1/cp F1/d}
-exact_length 2
```

```
set_scan_path chain2 -class bsd -hookup {F1/cp}
-exact_length 2 -view existing_dft
```

In the above example, for `chain1`, `F1/cp` is a valid hookup pin but `F1/d` is not due to the first reason (that is, the pin is specified as a hookup pin by using the `set_dft_signal` command with view spec).

For `chain2`, the same `F1/cp` becomes invalid due to the second reason (that is, in `set_scan_path` command, the view given is spec only).

Message

```
Pin: "<pin-name>" not a valid hookup pin
```

Severity

Error

SDC_344

This scan chain should have at least one element with -complete true option.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that if you specify `-complete true`, the specified scan path should have at least one valid element specified by using various options, that is, `-include/head/tail/ordered` elements.

Example

```
set_scan_path chain1 -include_elements {not_present}
-complete true
set_scan_path chain2 -complete true
```

In the above example, `chain1` has an element specified that does not actually exists, and `chain2` has no element specified. Hence, SpyGlass discards both these scan path specifications.

Message

```
Scan chain cannot be empty with -complete option set
```

Severity

Error

SDC_345

Given element is given multiple times in the same scan chain.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that an element is specified repeatedly in the same scan path.

Example

```
set_scan_path chain1 -include_elements {F1}
-tail_elements {F2 F1}
```

In the above example, `F1` is overlapping. Hence, SpyGlass flags the SDC_345 message in this case.

Message

```
Element "<object-name>" given for option "<option-name>" is
already present in this chain under a different option
```

Severity

Error

SDC_346

Two scan chains cannot have common elements. In such a case, the earlier specified chain exists and the recent one is discarded.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that scan paths cannot have common elements.

Example

```
set_scan_path chain1 -include_elements {F1}
-tail_elements {F2 }

set_scan_path chain2 -include_elements {F2}
```

In the above example, `F2` is common in `chain1` and `chain2`. Therefore, `chain2` is discarded.

Message

```
Scan-path "<first-scan-chain-name>" is being discarded due to
common element "<object-name>" with scan-path "<second-scan-
chain-name>"
```

Severity

Error

SDC_347

The given scan_path overwrites the earlier specification of set_scan_path

Language

Verilog, VHDL

Cause

SpyGlass flags this message if two scan chains have similar names. In such cases, SpyGlass overwrites the previous specification with the latest one.

Example

```
set_scan_path chain1 -include_elements {F1}
set_scan_path chain1 -include_elements {F2}
```

In the above example, the latest `chain1` specification overwrites the previous one.

Message

```
This scan chain "<scan-chain-name>" is overwriting the earlier specification due to same scan_chain_names
```

Severity

Warning

SDC_348

The specified object is not a valid ScanDataOut/ScanDataIn/ScanEnable.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that the port specified as value for `scan_data_out/in/enable` option should be declared prior to the type, `ScanEnable/DataOut/DataIn`, respectively, by using the `set_dft_signal` command.

Example

```
set_dft_signal -view spec -port out1 -type ScanDataOut
```

```
set_scan_path chain1 -include_elements {F2}
-scan_data_in w1 -scan_data_out out1
```

In the above example, `out1` is a valid `scan_data_out` port, but `w1` is not a valid `scan_data_in` port.

Message

```
Port : "<port-name>" not of the required type "<ScanDataOut |
ScanDataIn |ScanEnable>"
```

Severity

Error

SDC_349

-chain not specified

Language

Verilog, VHDL

Cause

SpyGlass flags this message if the name of scan chain in not specified by using the `-chain` option. The `-chain` option is a mandatory option.

Example

```
remove_scan_path -view spec
```

In the above example, the `-chain` option is not specified. Hence, SpyGlass flags the SDC_349 message.

Message

```
-chain not specified which is a mandatory option
```

Severity

Error

SDC_350

The given scan_path in the remove_scan_path command does not exist

Language

Verilog, VHDL

Cause

SpyGlass flags this message if the scan path specified in the `remove_scan_path` command does not exist. In such cases, the `remove_scan_path` does not work.

Example

```
set_scan_path chain1 -include_elements {F2}
remove_scan_path -chain chain2
```

In the above example, scan path `chain2` specified in the `remove_scan_path` command does not exist.

Message

```
scan_path "<scan-chain-name>" does not exist hence
remove_scan_path will not work
```

Severity

Error

SDC_351

The value of -view option should be same set in the chain which is being removed.
The value of -test_mode option should be same set in the chain which is being removed.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate the following:

- The value specified by the `-view` option in the `remove_scan_path` command should match with the value specified in the `set_scan_path` command for the specified scan chain.
- The value specified by the `-test_mode` option in the `remove_scan_path` command should match with the value specified in the `set_scan_path` command for the specified scan chain.

If any of the above conditions does not hold true, the `remove_scan_path` command does not work.

Example

```
set_scan_path chain1 -include_elements {F2} -view spec

remove_scan_path -chain chain1 -view existing_dft

set_scan_path chain1 -include_elements {F2}
-test_mode all_dft

remove_scan_path -chain chain1 -test_mode Inserted_scan
```

Message

```
Remove_scan_path does not work due to mismatch in "<view |
test_mode>" values for chain "<scan-chain-name>"
```

Severity

Error

SDC_353

The object given should be a port but it is a pin.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that the value given should be a port for validity, and not a pin.

Example

```
set_scan_path chain1 -scan_enable {w1 F1/CP}
```

In the above example, the object, `F1/CP`, is invalid. Hence, SpyGlass flags the SDC_353 message.

Message

```
Port should be given as an argument but "<pin-name>" is a pin
```

Severity

Error

SDC_354

The object given should be a sequential cell.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that the specified value should be a sequential cell.

Example

```
set_scan_path chain1 -scan_enable {w1 F1/CP}
```

In the above example, the object, `F1/CP`, is invalid. Hence, SpyGlass flags the SDC_354 message.

Message

```
Cell: "<cell-name>" either not present in the design or is not
of sequential type
```

Severity

Warning

SDC_355

When -bsd_style is mentioned for set_scan_path, ordered_list cannot have any elements.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that when the `-bsd_style` option is specified for the `set_scan_path` command, the ordered list cannot have any elements.

Example

```
set_scan_path chain1 -class scan -ordered_elements F1
-bsd_style global
```

```
set_scan_path chain1 -class bsd -hookup F2/cp
-exact_length 5 -ordered_elements w1
-bsd_style synchronous
```

In the above example, the `-bsd_style` option is specified for the `set_scan_path` command. In addition, valid element is also specified in the `-ordered_elements` option. Hence, SpyGlass flags the SDC_355 message in this case.

Consider another example, as given below:

```
set_scan_path chain1 -ordered_elements not_present
-bsd_style global
```

In the above example, the `-ordered_elements` option does not specify any valid element. Hence, SpyGlass does not flag the SDC_355 message in this case.

Message

```
When -bsd_style is mentioned for set_scan_path,ordered_list
cannot have any elements
```

Severity

Error

SDC_356

Number of pipeline stages should match for all the defined scan-chains

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that the number of pipeline stages should match for all the defined scan chains.

The number is set as soon as one scan path, with either of pipeline head registers or pipeline tail registers, is specified.

Example

```
set_scan_path chain1 -class scan -ordered_elements F1
-bsd_style global
```

Message

```
Number of pipeline stages should match for all the defined
scan-chains
```

Severity

Error

SDC_357

The specified port is not of output type.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that the port specified in the constraint file should be of output type.

Example

```
set_scan_path chain1 -include_elements {F2}
-scan_data_out w1
```

In the above example, the `w1` port should be an output port.

Message

`Port: <port-name> specified in constraint file not of type output`

Severity

Error

SDC_358

Domain information could not be extracted for this clock due to conflicting domain information found from its derived clocks.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that the domain information cannot not be extracted for a particular clock due to conflicting domain information found from its generated clocks.

Message

```
Domain information could not be extracted for this clock
'<clock-name>' due to conflicting domain information found from
its generated clocks '<clock1>' (File: <file1> Line:
<line-num1>) and '<clock2>' (File: <file2> Line: <line-num2>)
```

Severity

Error

SDC_362

Cannot specify the two options together.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate the options that cannot be specified together for a particular command, such as `all_registers`.

For example, in the `all_registers` command, you can specify either `-rise_clock`, `-fall_clock`, or `-clock` option at a time. Else, SpyGlass reports an error, as in the following case:

```
create_clock -name CLK -period 10 {in1}
all_registers -rise_clock CLK -fall_clock CLK
all_registers -fall_clock CLK -clock CLK
```

Message

```
'<option1>' cannot be specified with '<option2>'
```

Severity

Error

SDC_363

Must specify one of the two options.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate the use of at least one option for the given command, such as `set_voltage`.

For example, in the `set_voltage` command, you must specify either the `-object_list` option or the `-cell` and `-pg_pin_name` options. If none of them are specified, SpyGlass reports this error message, as in the following case:

```
set_voltage 1 -min 1.188
```

Message

```
Must specify one of these options: '<option1>' or '<option2>'
```

Severity

Error

SDC_364

Cannot specify '<option1>' without '<option2>'

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate the use of a specific option if some particular option is specified in a command.

For example, in the `set_timing_derate` command, if you specify the `-static` option, then you must also specify the `-net_delay` option. If you do not specify the `-net_delay` option, SpyGlass reports an error, as in the following case:

```
set_timing_derate -static -net_delay -early 0.7 [get_nets n1]
```

Message

```
Cannot specify '<option-name1>' without '<option-name2>'
```

Severity

Error

SDC_365

The value specified by '<option1>' cannot be greater than the value specified by '<option2>'

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that the value specified in a particular option should be less than the value specified in the other option of a given command.

For example, in the `set_voltage` command, the value specified by the `-min` option should be less than the value specified by the `-max_voltage` option. If the value specified `-min` option is greater than `-max_voltage` option, SpyGlass reports an error, as in the following case:

```
set_voltage 0.88 -min 1.188 -cell B2 -pg_pin_name in1
-dynamic 1.0 -min_dynamic 1.0
```

Message

```
The value specified by '<option-name1>' cannot be greater than
the value specified by '<option-name2>'
```

Severity

Error

SDC_366

A generic information message for the user

Language

Verilog, VHDL

Cause

This is a generic information message which reports cases that are not captured by the existing `SDC_*` messages.

For example, this message gets reported when you specify the `set_option debug_proc yes` project file command and there are procedure calls in an SDC file that contains errors.

Message

```
For more information on procedure call trace on some of the SDC
messages generated, please look into file '<file-name>'
```

Severity

Info

SDC_367

Number of edges and number of edge_shifts should be equal.

Language

Verilog, VHDL

Cause

SpyGlass flags this message to indicate that the numbers specified in the `-edges` and `-edge_shift` options do not match. These numbers must be exactly equal.

For example, the SDC_367 rule reports a violation in the following case, because the numbers specified by the `-edges` and `-edge_shift` options do not match:

```
create_generated_clock -name gclk2
-source [get_pins A1/Z] [get_pins A2/A]
-edges {1 3 6} -edge_shift {2 5 3 1 1}
```

Message

```
The number of edges specified by '-edges' does not match the
number of edge_shifts specified by '-edge_shift'
```

Severity

Info

SDC_370

There are multiple clocks reaching the source of the generated clock.

Language

Verilog, VHDL

Cause

SpyGlass reports this message to indicate that multiple clocks are reaching to the source of the generated clock.

If you do not specify the `-master_clock` argument of the `create_generated_clock` command, the fan-in traversal starts from the source point. If there are multiple clocks in the fan-in, the first clock is considered as the master clock.

If you want to create generated clocks for the remaining clocks, perform one of the following actions:

- Run the *SDC_GenerateIncr* rule of the SpyGlass Constraints solution.
- Edit your SDC file to specify the generated clocks.

Message

```
There are multiple clocks reaching the source of generated
clock. Since no master clock is specified, "<clk-name>" will be
taken as the master clock.
```

Severity

Info

SDC_372

set_load nets cannot be specified with either rise or fall options

Language

Verilog, VHDL

Cause

SpyGlass reports this error when the `set_load` command is specified with either `rise` or `fall` on the nets. For example, in the following code snippet, the SDC_372 rule reports an error violation message.

```
set_load 2 -rise -min [get_nets in2]
```

To resolve this violation message, use both `rise` and `fall`, as shown in the following:

```
set_load 2 -rise -fall [get_nets in2]
```

Message

```
For set_load: "rise|fall" cannot be used with nets
```

Severity

Error

SDC_378

During parse_proc_arguments, the options passed to the procedure should be of the type mentioned in define_proc_attributes

Language

Verilog, VHDL

Cause

The *SDC_378* rule reports a violation if the argument passed to the `parse_proc_arguments` command is not of the correct type.

The argument type should match with the type defined by the `define_proc_attributes` command.

Consider the following example:

```
proc my_create_gen_clock { args } {
   parse_proc_arguments -args $args results
}

define_proc_attributes  my_create_gen_clock -info "Procedure
to create a clock" \
  -define_args  {
  {-name          "see create_clock"   name    string   optional}
  {-period        "see create_clock"   period   int      required}
  {source_objects  "see create_clock"  source_objects   list optional}
  {s2             "see create_clock"   s2                list   required}
  {iv            "see create_clock"    iv                int    optional}
  {s3            "see create_clock"    s3                list   optional}
```

```
my_create_gen_clock -name ITSMYCLK -name MLKOP -period acv
-period 100 clk out1 20
```

In the above example, the *SDC_378* rule reports a violation for the `acv` value specified to the `-period` argument of the `my_create_gen_clock` command. This is because the `-period`

argument accepts an integer value (as defined by the `define_proc_attributes` command in this example). However, a non integer value (`acv`) is specified to the `-period` argument of the `my_create_gen_clock` command.

Message

```
value '<value>' for option '<option>' not of type '<type>
```

Severity

Error

SDC_379

During define_proc_attributes, 2 fields are at least required in argument definition

Language

Verilog, VHDL

Cause

The *SDC_379* rule reports a violation if minimum of two fields are not specified for an argument definition in the `define_proc_attributes` command.

Consider the following example:

```
proc my_create_gen_clock { args } {
   parse_proc_arguments -args $args results
}

define_proc_attributes  my_create_gen_clock -info "Procedure to create a
clock" \
    -define_args  {
       {-name          "see create_clock" name          string optional}
       {-period        "see create_clock" period         int   required}
       {source_objects}
}
```

For the above example, the *SDC_379* rule reports a violation because the minimum of two arguments are not specified for `source_objects` argument definition in the `define_proc_attributes` command.

Message

```
Need at least 2 fields in argument definition <argument> for
proc '<procedure>'
```

Severity

Error

SDC_380

The value specified in the command is negative value.

Language

Verilog, VHDL

Cause

The *SDC_380* rule reports a violation when a negative value is specified to the `set_min_delay`, `set_max_delay`, `set_input_delay`, and `set_output_delay` commands.

For example, this rule reports a violation in the following cases:

```
set_max_delay -55.0
set_input_delay -2215 -clock clk1 -max -rise data1
```

Message

```
negative value for "<command>"
```

Severity

Info

SDC_382

Certain commands applied on the output pins which are hanging are rejected.

Language

Verilog, VHDL

Cause

The *SDC_382* rule reports a violation when SDC commands are specified on the output pins that are hanging, that is, the pins with the 0 load.

A violation appears if such pins are specified to the following arguments:

- `-through` argument of the `set_multicycle_path`, `set_false_path`, `set_min_delay`, and `set_max_delay` commands
- `-port_pin_list` argument of the `create_clock`, `create_generated_clock`, `set_disable_timing`, and `set_case_analysis` commands

Example

Consider that the `u_sub/OUT2` pin is hanging. In this case, specifying the following command results in the *SDC_382* rule violation:

```
set_multicycle_path 2 -through [get_pins u_sub/OUT2] -setup
```

Message

```
Command "<SDC-command>" with "<output-pin>" in the "<command-argument>" option is rejected as this output pin is hanging
```

Severity

Error

SDC_391

The allow_non_standard_sdc feature cannot be used when "set_parameter pt yes" is specified.

Language

Verilog, VHDL

Causes

When the `pt` parameter is set to `yes` (`set_parameter pt yes`), SpyGlass behaves in sync with prime time.

However, the `allow_non_standard_sdc` command works only if the `pt` parameter is set to any of the following:

```
set_parameter pt supermode
```

or

```
set_parameter pt no
```

Message

```
The allow_non_standard_sdc feature only works with
set_parameter pt=no/supermode hence the value pt=yes is changed
to pt=supermode
```

Severity

Error

PLIB Parsing Built-In Rules

Overview

The rules of this category report violations based on the parsing done on the PLIB files.

PLIBSTX_1

Unexpected token encountered

Language

Verilog, VHDL

Rule Description

Unexpected token encountered

Message Details

```
Parse error on or before `<token>'
```

Severity

Syntax

PLIBWRN_1

Unsupported construct encountered

Language

Verilog, VHDL

Rule Description

The construct is not supported in the current version

Message Details

```
Unknown Attribute '<attribute>' in group '<group>'
```

Severity

Warning

PLIBWRN_2

Unexpected token encountered

Language

Verilog, VHDL

Rule Description

This construct is well supported but not inside group where it has been found

Message Details

```
Construct '<construct>' is not supported inside '<group>' group
```

Severity

Warning

PLIBWRN_3

Duplicate macro

Language

Verilog, VHDL

Rule Description

This macro is repeated in PLIB file

Message Details

```
Ignoring repeated macro '<macro>' in library '<library>'
```

Severity

Warning

PLIBWRN_4

Pin_type of pin is missing

Language

Verilog, VHDL

Rule Description

Pin_type is mandatory feature of pin to get power ground information

Message Details

```
Pin type of pin '<pin>' of macro '<macro>' is missing in PLIB
file, assuming pin type as 'signal'
```

Severity

Warning

PLIBWRN_5

Pin direction of pin is missing

Language

Verilog, VHDL

Rule Description

Pin direction is mandatory feature of pin to get electrical direction of pin

Message Details

```
Pin direction of pin '<pin>' of macro '<macro>' is missing in
PLIB file, assuming pin direction as 'inout'
```

Severity

Warning

Commands and Rule Parameters Sanity Checking Rules

This section describes the command-checking rules. These rules identify issues pertaining to user input. In addition, the rules also check for software configuration or version related problems.

Generic Command Checks

This section describes the built-in generic command checks.

CheckCelldefine

Reports if Verilog 'celldefine module is present in the current run which is skipped during rule-checking

Language

Verilog, VHDL

Rule Description

The CheckCelldefine rule flags if a Verilog `'celldefine` module is present in the current run and is skipped during rule-checking.

By default, the Verilog `'celldefine` modules are not checked by SpyGlass as they are leaf-level library cells, and any independent checks (such as style checks like indentation) on these cells are not desired by most of the users.

However, if you want to check the `'celldefine` modules, specify the `set_option check_celldefine yes` command in the project file.

NOTE: *SpyGlass always performs hook-up checks involving* `'celldefine` *modules even if you do not specify the* `set_option check_celldefine yes` command in the project file*.*

Message Details

```
Rule-checking skipped on Verilog celldefine modules present in the current run. If you want to check these modules, please specify 'check_celldefine' option
```

Severity

Info

CheckCMD_existence

Checks existence of design object specified as value in an option or Rule Parameter

Language

Verilog, VHDL

Rule Description

The checkCMD_existence is a generic rule that checks the existence of design objects specified as value in a command or a rule parameter.

The checkCMD_existence rule reports a failure in the following scenarios:

- If a given net, instance, port, sub-hierarchy module, or instance is not present in the design
- The specified object has been specified as a wildcard and it does not match any of the design objects in the design
- For a given rule parameter, an invalid object type may be specified. For example, consider the following rule parameter specification:

  ```
  set_parameter design_port top.p1
  ```

  The checkCMD_existence rule will flag if the port `top.p1` specified as value of the rule parameter `design_port` does not exist in the design.

Message Details

`Option <option-name>: <message>`

Where, *<message>* is a message about non existence of a value.

Severity

Error

Suggested Fix

Do the following:

- Check if the given string has some typo and correct it.
- The design specification may be incomplete. You need to provide the definition of missing modules.

- One of the modules in the hierarchy may not be synthesizable. You are required to correct the synthesis errors.
- Check for elaboration errors in the hierarchy and fix them.

checkCMD_wildcardMatch

Checks Parameter values containing design objects specification in wildcard format

Language

Verilog, VHDL

Rule Description

The checkCMD_wildcardMatch rule reports improper wildcard specification given as rule parameter value.

This rule reports a failure if specified wildcard string:

- does not match any of the objects in the design
- matches too many strings where only a single match is expected

The checkCMD_wildcardMatch rule flags through the following set of rules:

checkCMD_wildcardMatch01, *checkCMD_wildcardMatch02*, *checkCMD_wildcardMatch03*

These rules flag under different conditions as explained in the following sections.

Message Details

```
Option '<option>': Wildcard value '<value>' is incorrect
```

Severity

Error

Suggested Fix

Check the following:

- Check if the given wildcard string has some typo and correct it.
- Check if given wildcard string has some typo and correct it. If none of the above holds true, failure could be due to invalid object type specified for a given field. For example, an instance name specified in parameter `unexpected_ckcells_file`.

- If wildcard matches too many objects (where only a single match is expected), the wildcard may need to be pruned to match exactly one object. For example, in the following parameter value specification a violation is reported if the specified wildcard value does not match any cell in the design:

  ```
  set_parameter unexpected_ckcells_file module.<cll*>
  ```

checkCMD_wildcardMatch01

Checks if no wildcard match exists for design objects specification

Language

Verilog, VHDL

Rule Description

The checkCMD_wildcardMatch01 rule flags if a given wildcard string does not match any of the objects in the design.

Message Details

```
Option '<option>': Wildcard value '<value>' matched no design object
```

Severity

Fatal

Suggested Fix

Check if the given wildcard string has some typo and correct it.

checkCMD_wildcardMatch02

Checks if no wildcard match exists for design objects specification

Language

Verilog, VHDL

Rule Description

The checkCMD_wildcardMatch02 rule flags if a given wildcard string does not match any of the objects in the design.

Message Details

```
Option '<option>': Wildcard value '<value>' matched no design object
```

Severity

Warning

Suggested Fix

Check if the given wildcard string has some typo and correct it.

checkCMD_wildcardMatch03

Checks if multiple wildcard match exists for design objects specification

Language

Verilog, VHDL

Rule Description

The checkCMD_wildcardMatch03 rule violates if the given wildcard string matches multiple objects in the design and also reports some of the matched elements.

Message Details

```
Option '<option-name>': Wildcard value '<value>'[<option-type>]
for scalar parameter matched many ('<number-of-matches>')
elements in the design. Some of the matched values are
(<matched-values>)
```

Where, <option-type> can have values, such as net, instance, or terminal.

Severity

Error

Suggested Fix

```
Recheck the wildcard specification to match exactly one object.
```

checkCMD_value

Checks non-design objects specified as parameter values to ensure that these values confirm the parameter semantic

Language

Verilog, VHDL

Rule Description

The checkCMD_value rule is a generic check to validate parameters where the user specifies values other than design objects. The rule validates a variety of parameter values, including:

- Basic data type values like integer, float, string, identifier etc
- Enumeration type values to ensure that the given value falls within the enumeration range
- File type field values to ensure that the given file is read/write okay

NOTE: *All the checks mentioned above do not depend on the design.*

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message about invalid value specification.

Severity

Error

Suggested Fix

Do the following:

- Check the valid value set specified by the displayed message and accordingly update the given values. For example, consider the following rule parameter specification:

  `set_parameter strict_sync_check abc`

  The rule parameter `strict_sync_check` can be set to `yes` or `no`. In this case the rule will flag because the given value ‘abc’ is not one of allowed values.

- Fix the value as per the allowed integer range.

checkCMD_deprecate

Reports for deprecated SpyGlass Options and Rule Parameters.

Language

Verilog, VHDL

Rule Description

The checkCMD_deprecate rule flags if the given SpyGlass option or rule parameter has been deprecated.

The checkCMD_deprecate rule also reports the SpyGlass version in which an option or rule parameter has been deprecated. If already deprecated, the rule reports alternate available options.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that an option or parameter is deprecated and its corresponding alternative is available.

Severity

Warning

checkCMD_deprecate01

Reports for deprecated SpyGlass Options and Rule Parameters.

Language

Verilog, VHDL

Rule Description

The checkCMD_deprecate01 rule flags if the given SpyGlass option or rule parameter has been deprecated. For example, it flags with the `du` option.

If an option or rule parameter is going to be deprecated, this rule reports the SpyGlass version in which it would be deprecated, and if it has been already deprecated, the rule reports alternate options available.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that an option or parameter is deprecated its corresponding alternative is available.

Severity

Fatal

checkCMD_deprecate02

Reports for deprecated SpyGlass Options and Rule Parameters.

Language

Verilog, VHDL

Rule Description

The checkCMD_deprecate02 rule flags if the given command or rule parameter has been deprecated.

If a command or rule parameter is going to be deprecated, this rule reports the SpyGlass version in which it would be deprecated, and if it has been already deprecated, the rule reports alternate commands available.

NOTE: *Currently, this rule is not being used. But in future it can be used for a command.*

Message Details

`Option '<command-name>': <message>`

Where, *<message>* is a message indicating that a command or parameter is deprecated its corresponding alternative is available.

Severity

Severity

checkCMD_deprecate03

Reports for deprecated SpyGlass Options and Rule Parameters.

Language

Verilog, VHDL

Rule Description

The checkCMD_deprecate03 rule flags if the given command or rule parameter has been deprecated.

If the command or rule parameter is going to be deprecated, this rule reports the SpyGlass version in which it would be deprecated, and if it has been already deprecated, the rule reports alternate commands available.

NOTE: *Currently, this rule is not being used. But in future it can be used for a command.*

Message Details

`Option '<command-name>': <message>`

Where, *<message>* is a message indicating that a command or parameter is deprecated its corresponding alternative is available.

Severity

Warning

checkCMD_deprecate04

Reports for deprecated SpyGlass Options and Rule Parameters.

Language

Verilog, VHDL

Rule Description

The checkCMD_deprecate04 rule flags if the given SpyGlass option or rule parameter has been deprecated.

If an option or rule parameter is going to be deprecated, this rule reports the SpyGlass version in which it would be deprecated, and if it has been already deprecated, the rule reports alternate options available.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that an option or parameter is deprecated its corresponding alternative is available.

Severity

Info

checkCMD_policyrule

Reports for incorrect product / rule specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_policyrule rule flags if commands or rule parameters take product or rule names as their values.

This rule would flags if:

- Product name or rule name is not a valid alpha numeric value.
- Product passed as value is not enabled in current run
- Rule passed as value is not registered in current run
- Rule passed as value is not registered in selected language but in some other

The checkCMD_policyrule rule flags through the following set of rules:

checkCMD_policyrule01, *checkCMD_policyrule02*, *checkCMD_policyrule03*, *checkCMD_policyrule04*, *checkCMD_policyrule05*

These rules flag under different conditions as explained in the following sections.

Message Details

```
Option '<option>': Invalid Product/Rule specification
'<specification>' passed
```

Severity

Error

checkCMD_policyrule01

Reports for incorrect product / rule specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_policyrule01 rule flags if a product or rule name is not a valid alphanumeric value.

Message Details

```
Option '<option>': Product/Rule specification '<specification>'
not an alphanumeric value
```

Severity

Warning

checkCMD_policyrule02

Reports for incorrect product / rule specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_policyrule02 rule flags if the rule passed as value is not registered in the selected language.

Message Details

```
Option '<option>': Rule/Group '<rule/group>' not defined in
current rule-checking mode
```

```
Option '<option1>': Alternate Rule/Group '<rule/group>' for
Obsolete Rule/Group '<obsolete-rule/group>' specified by
'<option2>' option not defined in current rule-checking mode
```

Severity

Error

checkCMD_policyrule03

Reports for incorrect product / rule specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_policyrule03 flags when SpyGlass options or rule parameters take product or rule names as their values.

The checkCMD_policyrule03 rule flags if a rule passed as value for the `set_goal_option rules <rules>` or `set_goal_option addrules <rules>` command (in project file) is not registered in the current SpyGlass run.

Message Details

```
Option '<option>': Rule/Group '<rule/group>' not registered in
current run
```

```
Option '<option1>': Alternate Rule/Group '<rule/group>' for
Obsolete Rule/Group '<obsolete-rule/group>' specified by
'<option2>' option not registered in current run
```

Severity

Error (if the `set_goal_option ignore_undefined_rules yes` command is specified in the project file)

Fatal (if the `set_goal_option ignore_undefined_rules no` command is specified in the project file)

checkCMD_policyrule04

Reports for incorrect product / rule specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_policyrule04 flags when commands or rule parameters take product or rule names as their values.

The checkCMD_policyrule04 rule flags if a product passed as value is not enabled in the current SpyGlass run.

Message Details

```
Option '<option>': Product '<product>' not enabled in current
run
```

Severity

Error

checkCMD_policyrule05

Reports for incorrect product / rule specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_policyrule05 flags when SpyGlass options or rule parameters take product or rule names as their values.

This rule flags if the rule passed as value to the `set_option ignorerules` *<rule-name>* command (in project file) is not registered in the current SpyGlass run.

Message Details

```
Option '<option>': Rule '<rule>' not registered in current run


Option '<option1>': Alternate Rule/Group '<rule/group>' for
Obsolete Rule/Group '<obsolete-rule/group>' specified by
'<option2>' option not registered in current run
```

Severity

Error

checkCMD_dirfile

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfile rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule rule flags under the following conditions:

- Directory or file does not exist:
  - May have opened in read mode
  - Has to be written to and its parent directory does not have read permissions
- Directory or file exists but:
  - Has to be opened in read mode and does not have read permission
  - Has to be written to and does not have write permission
- Directory passed for a File type of option
- File passed for a directory type of option
- Directory to be read is empty

The checkCMD_dirfile rule flags through the following set of rules:

checkCMD_dirfile01, *checkCMD_dirfile02*, *checkCMD_dirfile03*, *checkCMD_dirfile04*, *checkCMD_dirfile05*, *checkCMD_dirfile06*, *checkCMD_dirfile07*, *checkCMD_dirfile08*, *checkCMD_dirfile09*, *checkCMD_dirfile10*, *checkCMD_dirfile11*, *checkCMD_dirfile12*, *checkCMD_dirfile13*, *checkCMD_dirfile14*, *checkCMD_dirfile15*, *checkCMD_dirfile16*, *checkCMD_dirfile17*

These rules flag under different conditions as explained in the following sections.

Message Details

```
Option '<option>': Invalid <file | directory> specification
'<specification>' passed <file/directory>
```

Severity

Error

checkCMD_dirfile01

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule01 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule01 rule flags if:

- Directory or file does not exist:
  - May have opened in read mode
  - Has to be written to and its parent directory does not have read permissions

Message Details

```
Option '<option>': <file | directory> specification
'<specification>' does not exist <file/directory>
```

Severity

Fatal

checkCMD_dirfile02

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule02 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule02 rule flags if:

- Directory or file does not exist:
  - May have opened in read mode
  - Has to be written to and its parent directory does not have read permissions

Message Details

```
Option '<option>': <file | directory> specification
'<specification>' does not exist <file/directory>
```

Severity

Error

checkCMD_dirfile03

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule03 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule03 rule flags if:

- Directory or file does not exist:
  - May have opened in read mode
  - Has to be written to and its parent directory does not have read permissions

Message Details

```
Option '<option>': <file | directory> specification
'<specification>' does not exist <file/directory>
```

Severity

Warning

checkCMD_dirfile04

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule04 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule04 rule flags if:

- Directory or file does not exist:
  - May have opened in read mode
  - Has to be written to and its parent directory does not have read permissions

Message Details

```
Option '<option>': <file | directory> specification
'<specification>' does not exist <file/directory>
```

Severity

Info

checkCMD_dirfile05

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule05 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule05 rule flags if directory is passed for a file type command.

Message Details

```
Option '<option-name>': '<value>' is a <directory>. File name expected
```

Severity

Error

checkCMD_dirfile06

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule06 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule06 rule flags if file is passed as value for directory type command.

Message Details

```
Option '<option-name>': '<value>' is a <file>. File name
expected
```

Severity

Error

checkCMD_dirfile07

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule07 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule07 rule flags if a directory or file exists and has to be written to but it does not have write permission.

Message Details

```
Option '<option>': Unable to open <file | directory> '<file/
directory>' for writing
```

Severity

Fatal

checkCMD_dirfile08

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule08 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule08 rule flags if: a directory or file exists and it has to be written to but it does not have the write permissions.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>': Unable to open <file | directory> '<file/
directory>' for writing
```

Severity

Warning

checkCMD_dirfile09

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule09 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule09 rule flags if a directory or file exists and it has to be written to but it does not have the write permissions.

Message Details

```
Option '<option>': Unable to open <file | directory> '<file/
directory>' for writing
```

Severity

Info

checkCMD_dirfile10

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule10 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule10 rule flags if: a directory or file exists and it has to be written to but it does not have the write permissions.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>': Unable to open <file | directory> '<file/
directory>' for writing
```

Severity

Fatal

checkCMD_dirfile11

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule11 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule11 rule flags if a directory or file exists and it has to be opened in read mode and does not has the read permissions. For example, it comes with various commands, such as `read_file -type sourcelist <file>` and `set_option enable_gateslib_autocompile yes`.

Message Details

```
Option '<option>': Unable to open <file | directory> '<file/
directory>' for reading
```

Severity

Fatal

checkCMD_dirfile12

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule12 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule12 rule flags if a directory or file exists and it has to be opened in read mode and does not has the read permissions.

Currently, this rule is not being used. But in future it can be used for some commands such as `set_option stopfile` *<file-name>* and `set_option stopdir` *<directory-name>*.

Message Details

```
Option '<option>': Unable to open <file | directory> '<file/
directory>' for reading
```

Severity

Info

checkCMD_dirfile13

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule13 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule13 rule flags if a directory or file exists and it has to be opened in read mode and does not has the read permissions.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>': Unable to open <file | directory> '<file/
directory>' for reading
```

Severity

Info

checkCMD_dirfile14

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule14 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule14 rule flags if a directory or file exists but does not have read permission.

The checkCMD_dirfule14 rule flags if a directory or file exists and it has to be opened in read mode and does not has the read permissions.

The rule is used for commands such as, such as `set_option stopfile <file-name>` and `set_option stopdir <directory-name>`.

Message Details

```
Option '<option>': Unable to open <file | directory> '<file/
directory>' for reading
```

Severity

Warning

checkCMD_dirfile15

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule15 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule15 rule flags if the file or directory to be read is empty. Currently, this rule is not used. But in future it can be used for some commands.

Message Details

```
Option/Rule-Parameter:'<name>' <file | directory> '<value>' is empty
```

Severity

Info

checkCMD_dirfile16

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule16 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule16 rule flags if the directory to be read is empty.

This rule would violate if a directory to be read is empty. For example, it is used for commands, such as `set_option stopfile <file-name>` and `set_option stopdir <directory-name>`.

Message Details

```
Option/Rule-Parameter:'<name>' <file | directory> '<value>' is empty
```

Severity

Warning

checkCMD_dirfile17

Reports for incorrect file / directory specification.

Language

Verilog, VHDL

Rule Description

The checkCMD_dirfule17 rule flags when commands or rule parameters take directories or files as their input values.

The checkCMD_dirfule17 rule flags if the directory to be read is empty. Currently, this rule is not used. But in future it can be used for some commands.

Message Details

```
Option/Rule-Parameter:'<name>' <file | directory> '<value>' is empty
```

Severity

Error

checkCMD_ignore01

Reports if a particular option has been ignored as some other superseding option has been specified

Language

Verilog, VHDL

Rule Description

The checkCMD_ignore01 rule identifies the SpyGlass command or rule parameter that has been ignored by other commands.

For example, the command `set_option report <report-name>` is ignored with the `set_option noreport yes` command.

Message Details

```
Option '<option1>' ignored as another option(s) '<option2>'
have been specified
```

Severity

Info

checkCMD_ignore02

Reports if a particular command has been ignored as it is effective in certain spyglass language mode only

Language

Verilog, VHDL

Rule Description

The checkCMD_ignore02 rule reports commands or rule parameters that are ignored because they are effective in a given language mode only and SpyGlass may not be running in that language.

For example, the `set_option enableSV` <*yes* | *no*> project file command is ignored if no Verilog file has been passed.

Message Details

```
Option '<option1>' ignored as no <file> design file is
specified
```

Severity

Warning

checkCMD_lc

Reports if a particular option can't be specified in library compiler mode

Language

Verilog, VHDL

Rule Description

The checkCMD_lc rule identifies the command or rule parameter that cannot be specified in library compiler mode but has been specified.

For example, the `policy` command cannot be given in library compiler mode.

Message Details

```
Option '<option>' not allowed in spyglass_lc run
```

Severity

Fatal

checkCMD_ignore_case_analysis

Reports if the set_case_analysis constraint is ignored

When to Use

This rule runs by default.

Description

SpyGlass reports a message when the `set_case_analysis` constraint is specified and the `ignore_case_analysis` argument is set.

Language

Verilog, VHDL

Parameter(s)

None

Constraint(s)

None

Messages and Suggested Fix

The following message is reported if the `set_case_analysis` constraint has been ignored because the `ignore_case_analysis` argument is set:

```
[INFO] Constraint 'set_case_analysis' has been ignored as
'ignore_case_analysis' option is set
```

Example Code and/or Schematic

Consider the following SGDC snippet:

```
current_design test
set_case_analysis -name a -value 10000001
set_case_analysis -name b -value 01010101
```

In the above example, SpyGlass reports the following violation message because the `ignore_case_analysis` argument is set:

```
Constraint 'set_case_analysis' has been ignored as
```

'ignore_case_analysis' option is set

Default Severity Label

Info

Related Reports

None

checkCMD_dependpolicyrule

Reports if the products / rules passed as option's values are running in current run or not

Language

Verilog, VHDL

Rule Description

The checkCMD_dependpolicyrule rule reports those SpyGlass options or rule parameters (that accept products or rules as their values), for which the passed product or rule is not enabled in the current SpyGlass run.

The checkCMD_dependpolicyrule rule flags through the following set of rules:

checkCMD_dependpolicyrule01 and *checkCMD_dependpolicyrule02*

These rules flag under different conditions as explained in the following sections.

Message Details

```
Option '<option>' ignored as none of the Polic(ies)/Rule(s)
'<name>' is enabled in current run
```

Severity

Warning

checkCMD_dependpolicyrule01

Reports if the products / rules passed as option's values are running in current run or not

Language

Verilog, VHDL

Rule Description

The checkCMD_dependpolicyrule01 rule reports those SpyGlass options or rule parameters that accept products or rules as their values, for which the passed product or rule is not enabled in the current SpyGlass run.

Message Details

```
Option '<option>' ignored as none of the Polic(ies)/Rule(s)
'<name>' is enabled in current run
```

Severity

Warning

checkCMD_dependpolicyrule02

Reports if the products / rules passed as option's values are running in current run or not

Language

Verilog, VHDL

Rule Description

The checkCMD_dependpolicyrule02 rule reports those commands or rule parameters that accept products or rules as their values, if passed rule is not enabled in the current SpyGlass run.

Message Details

```
Option '<option>' ignored as none of the Polic(ies)/Rule(s)
'<name>' is enabled in current run
```

Severity

Warning

checkCMD_recommended

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended rule flags a warning if a command or rule parameter is not recommended.

The checkCMD_recommended rule flags through the following set of rules:

checkCMD_recommended01, *checkCMD_recommended02*, *checkCMD_recommended03*, *checkCMD_recommended04*, *checkCMD_recommended05*, *checkCMD_recommended06*, *checkCMD_recommended07*, *checkCMD_recommended08*, *checkCMD_recommended09*

These rules flag under different conditions as explained in the following sections.

Message Details

```
Option '<option>' is not recommended option
```

Severity

Warning

checkCMD_recommended01

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended01 rule flags a warning if a command or rule parameter is not recommended (although SpyGlass understands and owns it). The rule provides a list of alternate options, if any.

Message Details

```
Option '<option>' is not recommended option. Alternatives
available : '<alternate-option>'
```

Severity

Warning

checkCMD_recommended02

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended02 rule reports a command or rule parameter that is not recommended, although SpyGlass understands and owns it. The rule also provides a list of alternate options, if any.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>' is not recommended option
```

Severity

Fatal

checkCMD_recommended03

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended03 rule reports a command or rule parameter that is not recommended, although SpyGlass understands and owns it. The rule also provides a list of alternate options, if any.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>' is not recommended option. Alternatives
available : '<alternate-option>'
```

Severity

Fatal

checkCMD_recommended04

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended04 rule reports a command or rule parameter that is not recommended, although SpyGlass understands and owns it. The rule also provides a list of alternate options, if any.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>' is not recommended option
```

Severity

Error

checkCMD_recommended05

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended05 rule reports a command or rule parameter that is not recommended, although SpyGlass understands and owns it. The rule also provides a list of alternate options, if any.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>' is not recommended option. Alternatives
available : '<alternate-option>'
```

Severity

Error

checkCMD_recommended06

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended06 rule reports a command or rule parameter that is not recommended, although SpyGlass understands and owns it. The rule also provides a list of alternate options, if any.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>' is not recommended option
```

Severity

Info

checkCMD_recommended07

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended07 rule reports a command or rule parameter that is not recommended, although SpyGlass understands and owns it. The rule also provides a list of alternate options, if any.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>' is not recommended option. Alternatives
available : '<alternate-option>'
```

Severity

Info

checkCMD_recommended08

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended08 rule reports a command or rule parameter that is not recommended, although SpyGlass understands and owns it. The rule also provides a list of alternate options, if any.

Currently, this rule is not being used. But in future it can be used for some commands.

Message Details

```
Option '<option>' is not recommended option
```

Severity

Warning

checkCMD_recommended09

Option is not a recommended Option

Language

Verilog, VHDL

Rule Description

The checkCMD_recommended09 rule reports a command or rule parameter that is not recommended, although SpyGlass understands and owns it. The rule also provides a list of alternate options, if any.

Message Details

```
Option '<option>' is not recommended option. Alternatives
available : '<alternate-option>'
```

Severity

Warning

checkCMD_together

Two commands/Rule Parameters should be specified together.

Language

Verilog, VHDL

Rule Description

The checkCMD_together rule flags if given commands or rule parameters have to be specified with other commands, but other required command has not been specified.

For example, the `nosavepolicy` project file command works only when the `enable_save_restore` project file command is set to `yes`.

Message Details

```
Option '<option1>' can/should be specified with option(s)
'<option2>' only
```

Severity

Info

checkCMD_together01

Two commands/Rule Parameters should be specified together.

Language

Verilog, VHDL

Rule Description

The checkCMD_together01 rule flags if given commands or rule parameters have to be specified with other commands, but other required command has not been specified.

For example, the `nosavepolicy` project file command works only when the `enable_save_restore` project file command is set to `yes`.

Message Details

```
Option '<option1>' can/should be specified with option(s)
'<option2>' only
```

Severity

Info

checkCMD_together02

Two commands/Rule Parameters should be specified together.

Language

Verilog, VHDL

Rule Description

The checkCMD_together02 rule flags if given commands or rule parameters have to be specified with other commands, but other required command has not been specified.

Currently, this rule is not used. But in future, it can be used for some command.

Message Details

```
Option '<option1>' can/should be specified with option(s)
'<option2>' only
```

Severity

Fatal

checkCMD_together03

Two commands/Rule Parameters should be specified together.

Language

Verilog, VHDL

Rule Description

The checkCMD_together03 rule flags if given commands or rule parameters have to be specified with other commands, but other required command has not been specified.

Currently, this rule is not used. But in future, it can be used for some command.

Message Details

```
Option '<option1>' can/should be specified with option(s)
'<option2>' only
```

Severity

Error

checkCMD_together04

Two commands/Rule Parameters should be specified together.

Language

Verilog, VHDL

Rule Description

The checkCMD_together04 rule flags if given commands or rule parameters have to be specified with other commands, but other required command has not been specified.

Currently, this rule is not used. But in future, it can be used for some command.

Message Details

```
Option '<option1>' can/should be specified with option(s)
'<option2>' only
```

Severity

Warning

checkCMD_unset_option

This rule reports if a particular command/parameter has been unset due to some reasons which can be identified from the message

Language

Verilog, VHDL

Rule Description

The checkCMD_unset_option rule flags if a particular command/parameter has been unset due to some reasons which can be identified from the message.

For example, if you specify `set_option gen_hiersgdc yes` and `set_option enable_save_restore yes` project file commands, all the save-restore will be disabled.

Message Details

```
Option/Parameter '<option/parameter-name>' has been unset.
Reason: <reason>
```

```
Option/Parameter '<option/parameter-name1>' has been unset as
Option/Parameter '<option/parameter-name2>' has been specified
```

```
Option/Parameter '<option/parameter-name>': Feature '<feature>'
is not supported with option '<option>'.Disabling all '<type>'
related options
```

```
Option/Parameter '<option/parameter-name>': Failed to check out
license feature '<feature>'.SpyGlass would disable
'<feature-name>' feature
```

Severity

Warning

CMD_top

Design Unit passed in top option does not exist

Language

Verilog, VHDL

Rule Description

The CMD_top rule flags if the design unit passed with the `set_option top` <*du-name*> project file command not exist.

Message Details

```
top '<du-name>': Design unit not found in the design
```

Severity

Fatal

checkCMD_nottogether03

Given options cannot be specified together

Language

Verilog, VHDL

Rule Description

The checkCMD_nottogether03 rule flags if two commands or rule parameters cannot be specified together.

Message Details

```
Options '<option1>' and '<option2>' can't be specified together
```

Severity

Error

checkCMD_wildcarddirfile

Reports about issues in wildcard pattern specified as values for file / directory type of options

Language

Verilog, VHDL

Rule Description

The checkCMD_wildcarddirfile rule flags if there are any issues in wildcard patterns specified for file/directory type of options.

These can be one of following issues:

- Wildcard pattern does not match any of the files or directories
- Wildcard pattern matches many files or directories

The checkCMD_wildcarddirfile rule flags through the following set of rules:

checkCMD_wildcarddirfile01, *checkCMD_wildcarddirfile02*, *checkCMD_wildcarddirfile03*

These rules flag under different conditions as explained in the following sections.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that a wildcard specification for a file or directory did not match any file or directory, or it matched many files or directories.

Severity

Error

checkCMD_wildcarddirfile01

Reports about issues in wildcard pattern specified as values for file / directory type of options

Language

Verilog, VHDL

Rule Description

The checkCMD_wildcarddirfile01 rule violates if there are any issues in wildcard patterns specified for files or directory type of options.

The checkCMD_wildcarddirfile01 rule flags if the specified wildcard pattern does not match any of the files or directories.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that a wildcard specification for a file or directory did not match any file or directory.

Severity

Fatal

checkCMD_wildcarddirfile02

Reports about issues in wildcard pattern specified as values for file / directory type of options

Language

Verilog, VHDL

Rule Description

The checkCMD_wildcarddirfile02 rule flags if the specified wildcard pattern does not match any of the files or directories.

For example, the rule flags if the specified wildcard pattern does not match any of the specified files using the `read_file -type sourcelist <files>` project file command.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that a wildcard specification for a file or directory did not match any file or directory.

Severity

Warning

checkCMD_wildcarddirfile03

Reports about issues in wildcard pattern specified as values for file / directory type of options

Language

Verilog, VHDL

Rule Description

The checkCMD_wildcarddirfile03 rule flags if the specified wildcard pattern matches many files or directories.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that a wildcard specification for a file or directory matched many files or directories.

Severity

Warning

checkCMD_duplicate01

Multiple Values specified for a Scalar type of option.

Language

Verilog, VHDL

Rule Description

The checkCMD_duplicate01 rule flags when you supply a scalar option multiple times with a different set of values.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that a scalar option is defined multiple times with given values and SpyGlass would pick only one value.

Severity

Warning

checkCMD_duplicate02

Multiple Values specified for a Rule Parameter.

Language

Verilog, VHDL

Rule Description

The checkCMD_duplicate02 rule flags when a rule parameter with a different set of values is specified multiple times.

Message Details

```
Rule-Parameter: '<name>' has been defined multiple times with
values '<values>'. Picked last value in current run
```

Severity

Info

checkCMD_duplicate03

Multiple specifications for a list type of option (File / Directory as of now)

Language

Verilog, VHDL

Rule Description

The checkCMD_duplicate03 rule flags when the same value is supplied multiple times for a list type of file or directory type of options.

For example, the rule flags when you specify:

```
-waiver first.swf -waiver first.swf
```

Message Details

```
Option '<option-name>': <message>
```

Where, *<message>* is a message indicating that a value is specified multiple times for a list type of file or directory option.

Severity

Warning

checkCMD_unknown

Unrecognized Argument specified

Language

Verilog, VHDL

Rule Description

The checkCMD_unknown rule flags when you pass an unknown switch to SpyGlass. It may be a rule parameter that is not enabled in the current SpyGlass run.

Message Details

```
Unrecognized Argument '<argument>' has been ignored
```

Severity

Warning

checkCMD_unused_param01

Parameter specified by product will not be used as none of the dependent rules are running

Language

Verilog, VHDL

Rule Description

The checkCMD_unknown_param01 rule flags when the value of the parameter is going to be ignored as dependent rules are not enabled in the current run.

Message Details

```
Parameter '<parameter>' has been ignored as none of the
dependent rules are enabled
```

Severity

Warning

NoTopDUFound

No Top design unit has been found in the design.

Language

Verilog, VHDL

Rule Description

The NoTopDUFound flags when no top-level design unit is found in the design. This can happen for the following cases:

- If the top design unit is `celldefined
- If the top design unit is stopped. For details, refer to the *Stopping Design Units* topic in *Atrenta Console User Guide*.
- If all the design units in the design are under translate on/off
- If no architecture is found in case of VHDL designs
- If the top design unit has elaboration error(s)

Message Details

```
No top design unit has been found in the design
```

Severity

Error

ReportIncompatibleRules

Reports incompatible rules specification.

Language

Verilog, VHDL

Rule Description

The ReportIncompatibleRules flags when incompatible rules are run together.

Message Details

The following message appears when incompatible rules are run together:

```
Option '<option>': Rules '<parent-rule>' and '<copy-rule>'
cannot run together. Rule '<copy-rule>' will run in place of
rule '<parent-rule>'
```

Severity

Info

ReportRuleNotRun

Reports reason of rule(s)/rule group(s) not being run specified with rules/addrules

Language

Verilog, VHDL

Rule Description

The ReportRuleNotRun rule reports the reason of rule(s) (specified through the `rules/addrules` project file command) not being run.

This rule reports a violation if a rule is specified by using the `rules/addrules` project file command and rule run stage is not active.

For example, consider a case in which you specify `set_goal_option rules SYNTH_5130` command, but synthesis is not happening. In this case, SpyGlass does not report any violation for the SYNTH_5130 rule because synthesis is not happening and this rule is a synthesis rule.

However, the ReportRuleNotRun rule flags an error message in this case specifying that the SYNTH_5130 rule is not run as synthesis is not enabled.

Message Details

The following message appears if the synthesis/analysis/elaboration rule *<rule-name>* is given in the `rules/addrules` project file command, and synthesis/analysis/elaboration is not enabled:

```
Rule '<rule-name>' not run as synthesis | analysis |
elaboration is not enabled.
```

The following message appears if all or some of the synthesis/analysis/elaboration rules of the rule group *<group-name>* are not being run because synthesis/analysis/elaboration is not enabled:

```
Some or all rules of rule group '<group-name>' not run as
synthesis | analysis | elaboration is not enabled
```

Severity

Error

Custom Command Checks

This section describes the built-in custom command checks.

CMD_param01

Parameter/Generic specification not found in the design

Language

Verilog, VHDL

Rule Description

This rule reports a violation if a parameter/generic specified by the `set_option param` *<name>* project file command does not exist in a design.

Message Details

```
'param' specification '<specification>' is not correct as
design unit '<design-unit>' has no parameter with name '<name>'
```

Example

Consider the following module `top`:

```
module top (a1,b1,y);
  parameter width_top = 7;
  parameter width_top_out = width_top+1 ;
  input [width_top-1:0] a1;
  input [width_top-1:0] b1;
  wire [width_top-1:0] a1;
  wire [width_top-1:0] b1;
output [width_top_out-1:0] y;
reg [width_top_out-1:0] y;
    my_tc U1(
  .a(a1),
  .b(b1),
  .c(y)
  );
endmodule
```

Now, consider that you specify the following specification for the `param` project file command:

```
set_option param 'top.width_top_1=2'
```

In this case, the CMD_param01 rule reports the following violation as there is no parameter by the name `width_top_1` in the `top` module.

```
'param' specification 'top.width_top_1=2' is not correct as
design unit 'top' has no parameter/generic with name
'width_top_1'
```

Severity

Fatal

CMD_param02

Parameter/Generic specification specified for a module with elaboration errors

Language

Verilog, VHDL

Rule Description

This rule reports a violation if a design unit for a generic/parameter specified in the `set_option param` <*name*> project file command has elaboration errors.

Message Details

```
Design unit '<design-unit>' being overridden by param
specification '<specification>' has elaboration errors
```

Severity

Warning

CMD_param03

Parameter/Generic specification overrides a top design unit

Language

Verilog, VHDL

Rule Description

This rule reports a violation to indicate that the value of a parameter/generic specified by the `set_option param` <*name*> command has been overridden.

Message Details

```
'param' overrides top-level design unit '<design-unit>'
parameter/generic '<parameter/generic-name>' with value
'<value>'
```

Example

Consider the following module `top`:

```
module top (a1,b1,y);
  parameter width_top = 7;
  parameter width_top_out = width_top+1;
  input [width_top-1:0] a1;
  input [width_top-1:0] b1;
  wire [width_top-1:0] a1;
  wire [width_top-1:0] b1;
  output [width_top_out-1:0] y;
  reg [width_top_out-1:0] y;
    my_tc U1(
  .a(a1),
  .b(b1),
  .c(y)
  );
endmodule
```

Now, consider that you use the following `param` project file command to override the value of the `width_top` parameter:

```
set_option param 'top.width_top=2'
```

In this case, the CMD_param03 rule reports the following informational message:

```
'param' overrides top-level design unit 'top' parameter/generic
'width_top' with value '2'
```

Severity

Info

CMD_param04

Parameter/Generic specification overrides a non top design unit

Language

Verilog, VHDL

Rule Description

This rule reports a violation to indicate that the value of a parameter/generic specified by the `set_option param` *<name>* project file command has been overridden for a module that is not a top-level module.

Message Details

```
'param' overrides non-top design unit '<design-unit>' parameter
'<parameter/generic-name>' with value '<value>'
```

Example

Consider the following design:

```
module my_tc (a,b,c);
  parameter width = 2;
  parameter out_width = width+1;
  input [width-1:0] a;
  input [width-1:0] b;
  wire [width-1:0] a;
  wire [width-1:0] b;
  output [out_width-1:0] c;
  assign c = a & b;
  endmodule


module top (a1,b1,y);
  parameter width_top = 7;
  parameter width_top_out = width_top+1;
  input [width_top-1:0] a1;
  input [width_top-1:0] b1;
  output [width_top_out-1:0] y;
  reg [width_top_out-1:0] y;
```

```
    my_tc U1(
    .a(a1),
    .b(b1),
    .c(y)
    );
endmodule
```

Now, consider that you specify the following `param` project file command:

```
set_option param 'my_tc.width="3"'
```

In this case, the CMD_param04 rule reports the following violation to indicate that the value of the `width` parameter has been overridden for the my_tc module that is not a top-level module.

```
'param' overrides non-top design unit 'my_tc' parameter/generic
'width' with value '3'
```

Severity

Warning

CMD_param05

Parameter/Generic specified for a non existent design unit

Language

Verilog, VHDL

Rule Description

This rule reports a violation if the `set_option param` <*name*> project file command is specified for a parameter/generic of a design unit that does not exist or if there are some elaboration errors in that design unit.

Message Details

```
Option '<option>': Design unit named '<design-unit>' passed
with parameter/generic '<parameter/generic-name>' does not
exist in design or has elaboration errors in it
```

Example

Consider the following design:

```
module my_tc (a,b,c);
  parameter width = 5;
  parameter out_width = width+1;
  parameter condition = "1";
  input [width-1:0] a;
  input [width-1:0] b;
  wire [width-1:0] a;
  wire [width-1:0] b;
  output [out_width-1:0] c;
  reg [out_width-1:0] c;
  always@(*)
  begin
    if (condition == "1")
      c <= a & b;
    else
      c <= a | b;
  end
endmodule
```

Now, consider that you specify the following `param` project file command:

```
set_option param 'my_tc_1.width="3"'
```

In this case, the CMD_param05 rule reports the following violation as the `my_tc_1` module does not exist in the design passed for SpyGlass analysis:

```
Option 'param': Design unit named 'my_tc_1' passed with parameter/generic 'width' does not exist in design or has elaboration errors in it
```

Severity

Warning

CMD_param06

Parameter/Generic value specified does not match the valid syntax allowed

Language

Verilog, VHDL

Rule Description

This rule reports a violation if you assign an invalid value to a parameter/generic in the `set_option param` <*name*> project file command.

Message Details

```
Option '<option>': Invalid value '<value>' passed with
Parameter/generic '<parameter/generic-name>' for Design Unit
<design-unit>
```

Example

Consider the following code snippet:

```
module top (a1,b1,y);
  parameter width_top = 7;
  parameter width_top_out = width_top+1;
  input [width_top-1:0] a1;
  input [width_top-1:0] b1;
  wire [width_top-1:0] a1;
  wire [width_top-1:0] b1;
  output [width_top_out-1:0] y;
  reg [width_top_out-1:0] y;
    my_tc U1(
    .a(a1),
    .b(b1),
    .c(y)
    );
endmodule
```

Now, consider that you specify the following `-param` command:

```
set_option param 'top.width_top="1'b8"'
```

In this case, the CMD_param06 rule reports the following violation as an invalid value is assigned to the `width_top` parameter:

```
Option 'param': Invalid value '1'b8' passed with Parameter/
generic 'width_top' for Design Unit top
```

Severity

Fatal

CMD_report

Report not registered

Language

Verilog, VHDL

Rule Description

The CMD_report rule flags when a report specified by the `set_option report` *<report-name>* project file command is not registered.

NOTE: *The CMD_report rule has been turned off for now.*

Message Details

```
Report '<report-name>' specified by '<option>' option not
registered in current run
```

Severity

Error

CMD_32bit

SpyGlass running in 32 bit mode on 64 bit architecture

Language

Verilog, VHDL

Rule Description

The CMD_32bit rule flags when SpyGlass is run in 32-bit mode on a 64-bit architecture.

NOTE: *The CMD_32bit rule violates under* `-strict_cmd_check` *only.*

Message Details

```
32bit run mode selected on 64bit machine. Memory access limited
to '<mode>
```

Severity

Info

CMD_define_severity01

Severity Label specified with define_severity option is not alphanumeric

Language

Verilog, VHDL

Rule Description

The CMD_define_severity01 rule flags when the severity label specified by the `set_goal_option define_severity` *<label>* project file command is not alphanumeric.

Message Details

```
Option '<option>' ignored as severity '<severity>' is not
alphanumeric
```

Severity

Warning

CMD_define_severity02

Severity Label specified with define_severity has not been used by product specified in define_severity option

Language

Verilog, VHDL

Rule Description

The CMD_define_severity02 rule flags when the severity label specified with the `set_goal_option define_severity <label>` project file command has not been used by the specified product.

Message Details

```
Option '<option>' ignored as severity '<severity>' is not used
by product '<product>'
```

Severity

Warning

CMD_define_severity03

Severity Label specified with define_severity has been registered under DATA severity Class

Language

Verilog, VHDL

Rule Description

The CMD_define_severity03 rule flags when the severity label specified with the `set_goal_option define_severity <label>` project file command has been registered under DATA severity class.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that some severity label is defined with the DATA severity and it is for SpyGlass internal use only.

Severity

Error

CMD_define_severity04

Severity Class specified with define_severity option is not one of the standard severity classes and has not been registered

Language

Verilog, VHDL

Rule Description

The CMD_define_severity04 rule flags when the severity class specified with `set_goal_option define_severity <class>` project file command is not one of the standard severity classes and has not been registered.

Message Details

`Option '<option-name>': <message>`

Where, *<message>* is a message indicating that some severity label is defined with an unregistered severity class and it will not be registered.

Severity

Error

CMD_gateslib01

Use of precompiled libraries through sglib option is preferred rather than text.lib through gateslib option

Language

Verilog, VHDL

Rule Description

The CMD_gateslib01 rule flags when use of precompiled libraries specified through the `read_file -type sglib <file-name>` project file command is preferred rather than text.lib specified by the `read_file -type gateslib <file-name>` project file command.

Message Details

`Option/Rule-Parameter '<name>': <message>`

Where, *<message>* is a message indicating that you should use the `read_file -type sglib <file-name>` project file command instead of the `read_file -type gateslib <file-name>` project file command.

Severity

Info

CMD_gateslib02

No cell has been picked from gateslib in the design

Language

Verilog, VHDL

Rule Description

The CMD_gateslib02 rule flags if no cell has been picked in the design from the given gateslib.

Message Details

```
No Cell has been used in design from gateslib file '<file-name>
```

Severity

Info

CMD_higher_capacity

Warns user that higher_capacity might be not too useful if user is running SpyGlass on 32 bit platform

Language

Verilog, VHDL

Rule Description

The CMD_higher_capacity rule warns that the `higher_capacity` might be not useful if SpyGlass is running on a 32-bit platform.

NOTE: *This rule is OFF for now.*

Message Details

```
Option '<option>': Switch '<switch>' might be not effective as
spyglass is running in 32 bit mode
```

Severity

Info

CMD_ignorelibs01

Reports ignorelibs option is given with v/y options

Language

Verilog, VHDL

Rule Description

The CMD_ignorelibs01 rule reports violation if the `set_option ignorelibs yes` project file command is used with the `set_option v <file-name>` or `set_option y <directory-path>` project file commands.

When the `ignorelibs` command is used with the `v/y` commands, SpyGlass does not perform rule-checking on module definitions specified by the `v/y` commands.

Message Details

```
Rule-checking not done on module definitions picked from v/y
specified libraries as 'ignorelibs' option has been specified
```

Severity

Info

CMD_define01

Macro specified with define option has not been used as no Verilog/Library file has been specified

Language

Verilog, VHDL

Rule Description

The CMD_define01 rule flags a warning when a macro specified by the `set_option define` *<macro-def>*=*<value>* project file command is not used as no Verilog or Library file has been specified.

Message Details

```
Macro specification '<macro>+define' ignored as no Verilog
design/library file is specified
```

Severity

Warning

CMD_define02

Macro not used in current run in any of the Verilog or Include files

Language

Verilog, VHDL

Rule Description

The CMD_define02 rule flags a warning when a macro is not used in the current SpyGlass run in any of the Verilog or Include files.

Message Details

```
Macro specification '<define>+<macro-name>' not used in current run
```

Severity

Warning

CMD_define03

Macro used in current run is in different case than specified

Language

Verilog, VHDL

Rule Description

The CMD_define03 rule flags a warning when a macro used in the current SpyGlass run is in a different case than specified.

Message Details

```
Macro '<macro>' differs in case from that used in file '<file>
```

Severity

Warning

CMD_define04

Macro used in current run in one of the precompiled files

Language

Verilog, VHDL

Rule Description

The CMD_define04 rule flags a warning when a macro is used in precompiled files.

Message Details

```
Macro '<macro>' used in precompiled design units but not in
currently given RTL files. Macro would be applied to currently
specified RTL files only
```

Severity

Warning

CMD_define05

Macro specified multiple times, SpyGlass would pick the first specified value

Language

Verilog, VHDL

Rule Description

The CMD_define05 rule flags a warning when a macro is specified multiple times. In that case SpyGlass will pick the first specified value.

Message Details

```
Macro '<macro>' specified multiple times. Picked first
specification in current run
```

Severity

Warning

CMD_sglib01

Sglib being used has been compiled from a different version of spyglass

Language

Verilog, VHDL

Rule Description

The CMD_sglib01 rule flags if sglib being picked has been compiled from a different SpyGlass Version than with which it is being run.

Message Details

`Option 'sglib': <message>`

Where, *<message>* is a message indicating that the given sglib is compiled with a different version of SpyGlass.

Severity

Warning

CMD_sglib02

Sglib Not compatible with current version of spyglass or file format not recognized

Language

Verilog, VHDL

Rule Description

The CMD_sglib02 rule flags if sglib is not compatible with the current version of SpyGlass or the file format is not recognized.

Message Details

`Option 'sglib': <message>`

Where, *<message>* is a message indicating that no cell is used from the given sglib.

Severity

Fatal

CMD_sglib03

No cell has been picked from sglib in the design

Language

Verilog, VHDL

Rule Description

The CMD_sglib03 rule flags if no cell has been picked in the design from the given sglib.

Message Details

```
No Cell has been used in design from sglib file '<file-name>
```

Severity

Info

CMD_sglib04

Sglib Not compiled with 'include_opt_data' option

Language

Verilog, VHDL

Rule Description

The CMD_sglib04 rule reports if an sglib file is not compiled with the `set_option include_opt_data` *<true|false|yes|no>* project file command. Recompiling the sglib file with this option gives better performance and early error detection during power estimation of the design.

Message Details

```
No opt data found in sglib '<sglib_path>'. Please use
'include_opt_data' option to re-compile the sglib using
SpyGlass Library Compiler for better performance and early
error detection.
```

Severity

Info

CMD_minus_f01

File specified in f option is not a command file

Language

Verilog, VHDL

Rule Description

The CMD_minus_f01 rule flags if a file specified using `read_file -type sourcelist` *<file-list>* project file command is not a command file.

Message Details

```
File '<command> <file-name>' is not a command file
```

Severity

Error

CMD_minus_f02

File specified with f option has been specified multiple times or recursively

Language

Verilog, VHDL

Rule Description

The CMD_minus_f02 rule flags if a file specified by using the `read_file -type sourcelist` *<file-name>* project file command has been specified multiple times or recursively.

Message Details

```
Command file '<file-name>' is recursively specified
```

Severity

Info

CMD_incdir01

Directory specified with incdir option not used

Language

Verilog, VHDL

Rule Description

The CMD_incdir01 rule flags when an incdir specification could not be used due to the following reasons:

- None of the include files are found in include directory
- All the include files in the given include directory have been already found specified in some other incdir specification

Message Details

```
Option '<option>': Incdir specification '<specification>' not
used in current run
```

Severity

Warning

CMD_incdir02

Some Verilog files are present in an include directory which has not been used

Language

Verilog, VHDL

Rule Description

The CMD_incdir02 rule flags when some Verilog files present in an include directory which has not been used.

Message Details

```
Option '<option>': Incdir '<include-directory>' contains some
Verilog files in it or inside it's sub-directories
```

Severity

Warning

CMD_incdir03

Incdir specification has been ignored as no Verilog file or Library file has been specified

Language

Verilog, VHDL

Rule Description

The CMD_incdir03 rule flags a warning when incdir specification is ignored because no Verilog or Library file has been specified.

Message Details

```
Incdir specification 'incdir+<value>' ignored as no Verilog
design/library file is specified
```

Severity

Warning

CMD_libext01

Libext option not specified with 'y' option

Language

Verilog, VHDL

Rule Description

The CMD_libext01 rule flags if the `set_option libext` <*`list-of-extensions`*> project file command is not specified with the `set_option y` <*`path-names-of-directories`*> project file command.

By default, SpyGlass picks the libext extension as empty.

Message Details

```
No 'libext' specification given with '<command>' option, By
default SpyGlass has picked 'libext' value as empty.
```

Severity

Info

CMD_minus_y01

Directory specified with y option remained unused

Language

Verilog, VHDL

Rule Description

The CMD_minus_y01 rule flags if the directory specified with the `set_option y` *<directory-path>* project file command is not used.

This may be because none of the files inside the Verilog library directory matched the extensions specified in the `set_option libext` *<text>* project file command.

Message Details

```
No file inside Verilog library directory specified by y
'<option>' option matched to the extensions specified in libext
```

Severity

Warning

CMD_minus_y02

Incomplete / Incorrect libext specification

Language

Verilog, VHDL

Rule Description

The CMD_minus_y02 rule flags if the directory specified by the `set_option y` *<directory-path>* project file command contains some files with their names starting with name of a black box in the design.

In this case, the `set_option libext` *<text>* project file command might be incorrect or incomplete.

Message Details

```
Verilog Library '<library>' specified with y option contains
files having name starting with name of following black boxes
in the design '<design>'. Your libext specification might be
incorrect/incomplete
```

Severity

Warning

CMD_minus_y03

Incorrect y specification

Language

Verilog, VHDL

Rule Description

The CMD_minus_y03 rule flags if the directory specified by the `set_option y <directory-path>` project file command contains some directories that have some files having their names starting with the name of a black box in the design and have the same extension as those specified by the `set_option libext <text>` project file command.

In such cases, probably the `set_option y <directory-path>` project file command is incorrect

Message Details

```
Verilog Library '<library>' specified with y option contains
files having name starting with name of following black boxes
in the design '<design>' in it's sub-directories. Your y
specification might be incorrect
```

Severity

Warning

CMD_minus_y04

Incorrect y specification and incorrect/incomplete libext specification

Language

Verilog, VHDL

Rule Description

The CMD_minus_y04 rule flags if the directory specified by the `set_option y` *<directory-path>* project file command:

- Contains some files having their names starting with the name of a black box in the design, and
- Contains some directories which have some files with their names starting with the name of a black box in the design and have the same extension as specified with the `set_option libext` *<text>* project file command.

Message Details

```
Verilog Library '<library>' specified with y option contains
files having name starting with name of following black boxes
in the design '<design1>' inside it and contains sub-
directories which contain files having name starting with name
of following black boxes in the design '<design2>'. Your libext
specifications might be incorrect/incomplete as well as your y
specification might be incomplete
```

Severity

Warning

CMD_lvpr01

Incorrect lvpr format

Language

Verilog, VHDL

Rule Description

The CMD_lvpr01 rule flags if the `set_option lvpr <format>` project file command specifies an incorrect format.

Message Details

```
Incorrect specification '<specification>' with lvpr option.
Expected <expected-value>
```

Severity

Error

CMD_lvpr02

Value specified with lvpr option is not an integer

Language

Verilog, VHDL

Rule Description

The CMD_lvpr02 rule flags if a value specified by the `set_option lvpr <value>` project file command is not an integer.

Message Details

```
Non-Integer Value '<value>' passed for rule '<rule>' in lvpr
specification '<specification>
```

Severity

Error

CMD_lvpr03

Rule specified with lvpr option has not been registered in current run

Language

Verilog, VHDL

Rule Description

The CMD_lvpr03 rule flags if the rule specified by the `set_option lvpr <rule>` project file command is not registered in the current SpyGlass run.

Message Details

```
Rule '<rule-name>' passed with lvpr specification
'<specification>' is not registered
```

Severity

Warning

CMD_overload

No product with overload extension specified with overload found in I paths

Language

Verilog, VHDL

Rule Description

The CMD_overload rule flags a warning if a product with overload extension, specified using the `set_goal_option overload <named-overload>` project file command, is not found in the paths specified by the `set_option I <directory-name>` project file command.

Message Details

`Option 'overload': <message>`

Where, *<message>* is a message indicating that the product specified with the `set_goal_option overload` *<named-overload>* project file command is not found in the path specified by the `set_option I` *<directory-name>* project file command.

Severity

Warning

CMD_overloadPolicy

Overload specified for product not running in current run

Language

Verilog, VHDL

Rule Description

The CMD_overloadPolicy rule flags if the overload specified for a product is not running in the current SpyGlass run.

Message Details

`Option 'overload': <message>`

Where, *<message>* is a message indicating that the `set_goal_option overload` *<named-overload>* project file command is specified for a product that is not used in the current SpyGlass run.

Severity

Error

CMD_overloadrule01

Overload ignored as rule not registered in run language

Language

Verilog, VHDL

Rule Description

The CMD_overloadrule01 rule flags a warning when an overload is ignored because the rule is not registered in the run language.

Message Details

```
Overload Specification for rule '<rule-name>' has been ignored
```

Severity

Warning

CMD_overloadrule02

Overload specification tried to downgrade Fatal or Error severity

Language

Verilog, VHDL

Rule Description

The CMD_overloadrule02 rule flags if an overload specification attempts to downgrade a rule with fatal or error severity.

Message Details

```
Overload for rule '<rule-name>' (language: '<language>')
downgrades its severity from FATAL (severity label:
'<severity-label1>') to <severity> (severity label:
'<severity-label2>')
```

Severity

Error

CMD_register_severity

Severity Label not registered

Language

Verilog, VHDL

Rule Description

The CMD_register_severity rule flags if a severity label specified in rule registration has not been registered under any of the severity classes. It is registered by default under severity ‘Error’.

Message Details

`<define_severity>: <message>`

Where, *<message>* is a message indicating that the given severity label is not registered with any severity and it will be registered as `ERROR`.

Severity

Warning

CMD_param01

Parameter/Generic specification not found in the design

Language

Verilog, VHDL

Rule Description

The CMD_param01 rule flags if a rule parameter or generic specification is not found in the design.

Message Details

```
'<Parameter | Generic>' specification '<specification>' is not
correct as design unit '<design-unit>' has no parameter with
name '<name>'
```

Severity

Fatal

CMD_param02

Parameter/Generic specification specified for a module with elaboration errors

Language

Verilog, VHDL

Rule Description

The CMD_param02 rule flags if a rule parameter or generic specification has been specified for a module with elaboration errors.

Message Details

```
Design unit '<design-unit>' being overridden by -param
specification '<specification>' has elaboration errors
```

Severity

Warning

CMD_param03

Parameter/Generic specification overrides a top design unit

Language

Verilog, VHDL

Rule Description

The CMD_param03 rule flags if a rule parameter or generic specification overrides a top design unit.

Message Details

```
'<Parameter | Generic>' overrides top-level design unit
'<design-unit>' parameter '<parameter>' with value '<value>'
```

Severity

Info

CMD_param04

Parameter/Generic specification overrides a non top design unit

Language

Verilog, VHDL

Rule Description

The CMD_param04 rule flags if a rule parameter or generic specification overrides a non-top design unit.

Message Details

```
'<Parameter | Generic>' overrides non-top design unit
'<design-unit>' parameter '<parameter>' with value '<value>'
```

Severity

Warning

CMD_param05

Parameter/Generic specified for a non existent design unit

Language

Verilog, VHDL

Rule Description

The CMD_param05 rule flags if a rule parameter or generic specification is given for a non-existent design unit.

Message Details

```
Option '<option>': Design unit named '<design-unit>' passed
with parameter '<parameter>' does not exist in design or has
elaboration errors in it
```

Severity

Warning

CMD_param06

Parameter/Generic value specified does not match the valid syntax allowed

Language

Verilog, VHDL

Rule Description

The CMD_param06 rule flags if a rule parameter or specified generic value does not match the valid syntax allowed.

Message Details

```
Option '<option>': Invalid value '<value>' passed with
Parameter '<parameter>' for Design Unit <design-unit>
```

Severity

Warning

CMD_sortrule

Reports issues in sortrule specification

Language

Verilog, VHDL

Rule Description

The CMD_sortrule rule reports issues in the `set_option sortrule <options>` project file command.

Message Details

```
Option '<option-name>' : More than one option specified for the argument '<argument-name>' in <option-name>. Only First will be considered
```

```
Option '<option-name>' : More than one option specified for the file '<file-name>' in '<option-name>'. Only First will be considered
```

```
Option '<option-name>' : More than one option specified for the line '<line-number>' in '<option-name>'. Only First will be considered
```

Severity

Warning

CMD_dump_mode

Reports if Verilog precompile options have been incorrectly specified

Language

Verilog, VHDL

Rule Description

The CMD_dump_mode rule reports if Verilog precompile options have been incorrectly specified.

For example, the rule will flag if the `set_option dump_all_modes yes` project file command is specified without the `set_option enable_precompile_vlog yes` command.

Message Details

```
option '<option1>' should be specified with option '<option2>'
to do a Precompile of Verilog Files
```

Severity

Warning

CMD_cell_define_messages

Options no_celldefine_messages or no_rcheck_celldefine have been ignored

Language

Verilog, VHDL

Rule Description

The CMD_cell_define_messages rule reports that the `no_celldefine_messages` or `no_rcheck_celldefine` commands have been ignored because no Verilog source file or library file is specified while running in Verilog/Mixed or VHDL mode, and no sglib file has been specified.

Message Details

```
option '<option>' ignored as no Verilog design/library file is specified
```

Severity

Warning

CMD_dnc_param01

Parameter/generic specification not found in the design

Language

Verilog, VHDL

Rule Description

The *CMD_dnc_param01* rule reports a violation if the parameter/generic specified to the `dnc_param` command does not exist in the design.

Message Details

```
'dnc_param' specification '<specification>' is not correct as
design unit '<design-unit-name>' has no parameter with name
'<param-name>'
```

Severity

Fatal

Example

Consider that you specify the following command:

```
set_option dnc_param sfifo.INT
```

If the `sfifo` design unit does not contain any parameter of the name `INT`, the following violation appears:

```
'-dnc_param' specification 'sfifo.INT' is not correct as design
unit 'sfifo' has no parameter with name 'INT'
```

CMD_dnc_param02

Parameter / generic specification not found in the design

Language

Verilog, VHDL

Rule Description

The *CMD_dnc_param02* rule reports a violation the design unit specified to the `dnc_param` command either does not exist in the current design or it contains elaboration errors.

Message Details

```
Option '<option>': Design unit named '<design-unit-name>'
passed with parameter '<parameter-name>' does not exists in
design or has elaboration errors in it
```

Severity

Warning

Example

Consider that you specify the following command:

```
set_option dnc_param reorderbit.abc
```

If `reorderbit` does not exist in the current design, the following violation appears:

```
Option '-dnc_param': Design unit named 'reorderbit' passed with
parameter 'abc' does not exists in design or has elaboration
errors in it
```

HdlLibDuCheck

Reports about usability/issues related to the specification of option 'hdllibdu' and 'add_hdllibdu_lexical_checks'

Language

Verilog, VHDL

Rule Description

The HdlLibDuCheck rule flags usability/issues related to the specification of the `hdllibdu` and `add_hdllibdu_lexical_checks` project file commands.

The HdlLibDuCheck rule flags through the following set of rules:

HdlLibDuCheck_01, *HdlLibDuCheck_02*, *HdlLibDuCheck_03*

These rules flag under different conditions as explained in the following sections.

Severity

Info

HdlLibDuCheck_01

To run RTLDU/RTLDULIST/RTLALLDULIST/LEXICAL rules on precompiled designs set option 'hdllibdu'

Language

Verilog, VHDL

Rule Description

The HdlLibDuCheck_01 rule flags usability/issues related to the specification of `set_option hdllibdu` <*yes* | *no*> project file command.

This rule flags if the `hdllibdu` command is not set to `yes` to run RTLDU/RTLDULIST/RTLALLDULIST type of rules on precompiled designs.

To run RTLDU/RTLDULIST/RTLALLDULIST type of rule on precompiled designs, specify `set_option hdllibdu yes` project file command.

Message Details

```
"To run RTLDU/RTLDULIST/RTLALLDULIST/LEXICAL rules on
precompiled designs set option 'hdllibdu'
```

Severity

Warning

HdlLibDuCheck_02

Reports that 'hdllibdu' is not required if no RTL/LEXICAL rules are running.

Language

Verilog, VHDL

Rule Description

The HdlLibDuCheck_02 rule flags if the `set_option hdllibdu yes` project file command is used if no RTL/LEXICAL rules are running.

This command enables RTL/LEXICAL level checks on precompiled designs. This command is not required if no RTL/LEXICAL rules are running.

Message Details

```
No need to specify '<command>' since <rule-type> type of rules
are not running
```

Severity

Info

HdlLibDuCheck_03

Reports that 'hdllibdu' is not required if no precompiled design unit is used in current run.

Language

Verilog, VHDL

Rule Description

The HdlLibDuCheck_03 rule flags if the `set_option hdllibdu yes` project file command is used if no precompiled design unit is used in current.

This command enables RTL/LEXICAL level checks on precompiled designs. This command is not required if no RTL/LEXICAL rules are running.

Message Details

```
No need to set option '<command>' since no precompiled design
unit is being used in current run
```

Severity

Info

SpyGlass Informational Messages

Overview

This section describes the following:

- The *Info_Case_Analysis* rule
- The informational messages generated during SpyGlass run. See *Informational Messages Generated during SpyGlass Run*.

Info_Case_Analysis

Highlights case-analysis settings

Rule Description

Use this rule to view constant values, such as values set by using the `set_case_analysis` SGDC constraint on a terminal/net, supply values, or ground values propagating through a design.

This rule generates information to highlight case analysis values and power/ground information in a schematic. This power/ground information is inferred from the design.

Information generated by this rule is useful for observing value propagation in a design.

It is recommended to run this rule with other rules as this rule provides valuable debug aid to see how case values are propagating through the design.

Prerequisites

Specify case analysis signals by using the `set_case_analysis` constraint. Based on this information, SpyGlass simulates a design and annotates the simulation value (0 or 1) for each accessible net.

Performing Value Propagation

If you specify the `set_option enable_const_prop_thru_seq yes` command in the project file, the *set_case_analysis* values propagate beyond sequential elements. Constant propagation from flop-D happens only if one of the following conditions is true:

- Flip-flop does not have preset/clear pin.
- Data is tied to 0, and flip-flop has only clear pin.
- Data is tied to 1, and flip-flop has only preset pin.

While performing value-propagation, SpyGlass generates the following message for each top-level design unit (*<top-du-name>*) where case analysis information has been processed:

```
Information for set_case_analysis value propagation for "<top-
```

du-name>" is displayed

Viewing Case Analysis Settings Along With Rule Violations

While debugging a violation of a rule in the *Incremental Schematic* window, you can view case analysis settings along with the violation of other rules.

To view case analysis settings, perform any of the following actions:

- Select the rule violation.
- Double-click on the rule violation message of the *Info_case_analysis* rule while pressing the <*Ctrl*> key.
- Select the rule violation and open the *Incremental Schematic* window.
- Click the *Edit -> Show Case Analysis* menu option in the *Incremental Schematic* window

  For more details, refer to *Atrenta Console User Guide*.

Example

Consider the following test case on which the *Ac_unsync01* rule of SpyGlass CDC solution and *Info_Case_Analysis* rules are run:

```
// test.v

module testme(a, b, s, c, d, clk1, clk3);
    input a, b, s, clk1, clk3;
    output c, d;
    ff FA(b, fa_out, clk1);
    internal INT(a, fa_out, s, fc_in);
    ff FC(fc_in, c, clk3);
endmodule

module internal (a, b, s, c);
    input a,b,s;
    output c;
    assign c = s ? a : b;
endmodule

module ff(in, out, clk);
    input in, clk;
    output reg out;
    always @ (posedge clk)
        out = in;
endmodule



// constraints.sgdc

current_design testme
clock -name clk1
clock -name clk3
set_case_analysis -name testme.INT.s -value 1
set_case_analysis -name a -value 1
```

Once SpyGlass analysis is complete, double-click on the violation of the *Propagate_Clocks* rule of SpyGlass CDC solution and open the *Incremental Schematic* window.

This schematic is shown in the following figure:

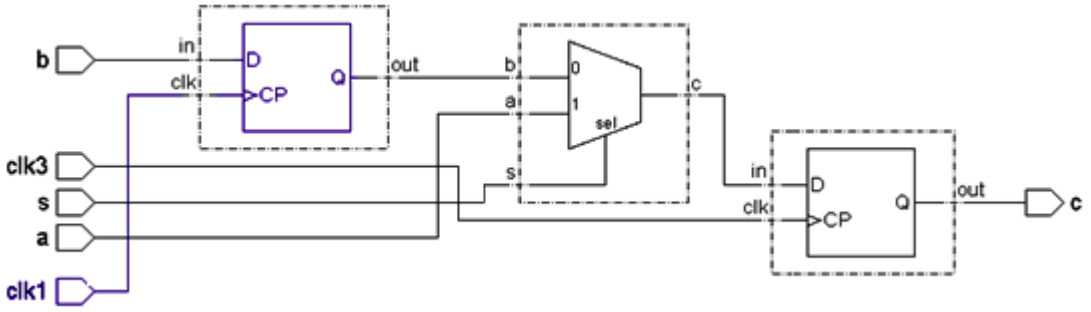


FIGURE 1.

In the above figure, notice that there is a crossing that is not being reported.

Now select the *Edit -> Show Case Analysis* menu option in the *Incremental Schematic* window. The schematic now appears as shown in the following figure:

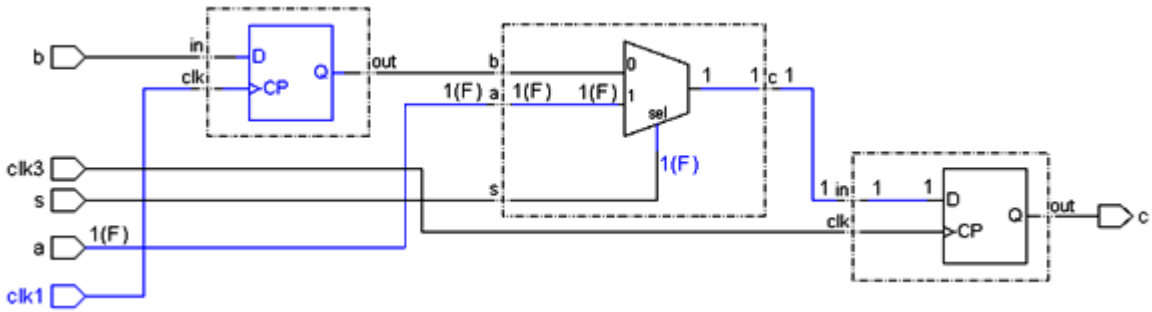


FIGURE 2.

In the above schematic, notice that the `set_case_analysis` value on the select line of the MUX is blocking the crossing path. As a result, the path is not getting reported by the *Ac_unsync01* rule.

Schematic Details

The *Info_Case_Analysis* rule highlights power ground simulation values and `set_case_analysis` constraints propagated through combinational logic.

Severity

Info

Informational Messages Generated during SpyGlass Run

These messages are printed in the *Session Log* page of the *Results* pane in Atrenta Console GUI or at the stdout while working in the batch mode.

The following table lists the informational messages:

Message Number	Message
[1]	"Rule registration requires non-null rule name";
[2]	"'<string>', Rule registration requires non-null init or exec function name for the first time";
[3]	"Group registration requires non-null group name";
[4]	"Shared C Library `<name>' cannot be opened: <reason>.";
[5]	"Shared C Library specified as NULL";
[6]	"Work Directory `<dir-name>' does not exist.";
[7]	"Environment variable SPYGLASS_HOME not specified. Set environment variable SPYGLASS_HOME and then rerun spyglass.";
[8]	"No <file-name> file specified.";
[9]	"License feature <license-feature> checkout failed.";
[10]	"No command file name specified after 'f' option";
[11]	"'<string>' Duplication/Recursion of command file (ignoring....)";
[12]	"No rule argument specified after 'rules' option";
[13]	"No script file name specified after 'script' option";
[14]	"No report generator name specified after 'report' option";
[15]	"No report file name specified after 'reportfile' option";
[16]	"No violation DB file name specified after 'vdbfile' option";
[17]	"No code specified to suppress banner";
[18]	"No top design unit name specified after 'top' option";
[19]	"No work directory path specified after 'work' option";
[20]	"Need two arguments for <option-name> option";
[21]	"No du-name specified after 'du' option";
[22]	"'<message>': error opening command file (Reason: <reason>)";

Message Number	Message
[23]	"Cannot handle string containing more than 4096 characters on one line inside command file `<file-name>'";
[24]	"End of Command File `<file-name>' before comment termination";
[25]	"Following Rules/Rule Groups either do not exist or are not available for the selected language mode <language-mode>";
[26]	"'<message>': Cannot open violation DB file for writing (<string>)";
[27]	"'<message>': Can't open VHDL file";
[28]	"NULL init/exec function name specified for error handler";
[29]	"NULL init/exec function name specified for report generator";
[30]	"'<message>': Report Generator is not registered";
[31]	"<message>': Unable to open help file for reading";
[32]	"'<message>': No such file or directory";
[33]	"'<message>': Permission denied";
[34]	"'<message>': Path name is too long";
[35]	"'<message>': Symbolic link loop";
[36]	"'<message>': is a <string>. <name> name expected.";
[37]	"'<message>': the link has been severed";
[38]	"<message>";
[39]	"Rule `<rule-name>' is registered, but switched off";
[40]	"Logical Library `<lib-name>' not defined";
[41]	"Library `<lib-name>' is not readable.";
[42]	"File <file-name>, Line <line-num>: Environment Variable `<var-name>' is not defined.";
[43]	"No file specified after 'stopmodule_list' option";
[44]	"No file specified after 'topmodule_list' option";
[45]	"No library file specified after 'v' option";
[46]	"No library directory specified after 'y' option";
[47]	"No name specified after 'synth_prefix' option";
[48]	"Missing quote (\") for string `<string>'";
[49]	"Debug Level not Specified";
[50]	"No code specified to suppress custom rule authentication";

Message Number	Message
[51]	"Shared library `<lib-name>' has a format which is not recognized by SpyGlass. Use spygenlib to generate a shared library.";
[52]	"SpyGlass Option `<option>' not available with license feature `<feature>'.";
[53]	"Cannot open lint file `<file-name>' for reading";
[54]	"Cannot unpack shared library `<lib-name>' in the <dir-name> directory: <message>";
[55]	"Max Violation Per Rule not Specified";
[56]	"Argument `<arg-name>' for Rule `<rule-name>' not recognized";
[57]	"Cannot open shared library `<lib-name>' for reading (Reason: <reason>)";
[58]	"Memory threshold value not specified";
[59]	"The perl `-e' option requires a following 'command'";
[60]	"`spyglass check' only allows you a limited number of `I' options";
[61]	"`spyglass check' only allows you a limited number of `m' or `M' options";
[62]	"`spyglass check' only allows you a limited number of `e' options";
[63]	"Too many files are specified with `glib' option";
[64]	"No file name specified after `glib' option";
[65]	"The show synthesis syntax is `showsynth' or `showsynth=filename'";
[66]	"Could not open the `showsynth' output file `<file-name>' because `<reason>'";
[67]	"The show flattened syntax is `showflat' or `showflat=filename'";
[68]	"Could not open the `showflat' output file `<file-name>' because `<reason>'";
[69]	"+vlogarg argument not specified";
[70]	"+vhdlarg argument not specified";
[71]	"No file name specified after `gateslibfile' option";
[72]	"No file type specified after `gateslibtype' option";
[73]	"Too many files are specified with `gateslib' option";
[74]	"+syntharg argument not specified";
[75]	"Creating the Work Directory `<dir-name>' for <dump> precompiled dump.";
[76]	"Using `<dir>' as the Work Directory for <dump> precompiled dump.";
[77]	"Physical path '<path>' does not exist.";
[78]	"Shared C Library `<lib-name>' not found in <path> or `I' paths";
[79]	"'<message>': Cannot open Log file for writing (<message>)";

Message Number	Message
[80]	"No Log file name specified after `logfile' option";
[81]	"Write failed in <message>: <message>";
[82]	"Need an argument following <option-name> option";
[83]	"Design unit <du-name> has no generic/param named <name>";
[84]	"Too many policies specified with option 'policy'";
[85]	"'policy/policies' option is not supported with 'script' option";
[86]	"OEM policy `<policy>' missing in environment variable `<var-name>'.";
[87]	"Cannot open OEM policy `<policy>' for reading (Reason: <reason>) ";
[88]	"Cannot load shared library `<lib-name>' on '<platform>' platform. It is built for '<platform>' platform only.";
[89]	" File '<file-name>', Line '<line-num>': It is not allowed to load a shared object (<obj-name>) in an overload file in OEM mode.";
[90]	"Cannot load OEM policy `<policy>'. It is not valid OEM policy.";
[91]	"It is not allowed to register a rule (<rule-name>) in an overload file for policy '<policy>' in OEM mode.";
[92]	"Illegal language used in '<rule/group>' Rule/Group Registration";
[93]	"Illegal value '<value>' passed with 'opt_level'; Allowed value is between 0 and 3. Defaulting to '0'...";
[94]	"Incorrect argument passed to 'param'. Correct usage is : 'set_parameter param <key>=<value>', where <key> is either <ent>.<gen> or <mod>.<param>";
[95]	"Environment Variable '<var-name>' not found";
[96]	"Design unit named '<du-name>' passed to param option does not exist in design.";
[97]	"Cannot use 'top' and 'du' simultaneously.";
[98]	"No Directory specified after 'wdir' option.";
[99]	"Rule <rule-name>: <message>.";
[100]	"Memory Allocation Failed. Exiting ...";
[101]	"Unable to open <file> for lib2v operation";
[102]	"<message>: No such Verilog Library directory (passed with 'y' option)";
[103]	"Unrecognized Option '<option-name>'. Ignoring ...";
[104]	"'<message>': Cannot open Report file for writing (Reason: <reason>)";
[105]	"No file name specified with 'sgdc' option";

Message Number	Message
[106]	"Elaboration error(s) found in design. RULE CHECKING ABORTED.";
[107]	"Elaboration error(s) found. No synthesis will be done";
[108]	"Unsupported option with <option-name> : <message>";
[109]	"Missing mode specific switch 'vhdl' or 'verilog' or 'mixed' or 'def'. Use of mode specific switch is mandatory.";
[110]	"Rule/Group <rule/group> is already registered with 'verilog+vhdl' language.";
[111]	"Rule/Group <rule/group> registered in 'VHDL/Verilog' is replaced with registration in 'MIXED'";
[112]	"<message>";
[113]	"No design unit is given with 'stop' option";
[114]	"'mixed' option will be read in with higher priority and override '<option-name>' option";
[115]	"'<message>' : Unable to open directory for writing (<message>)";
[116]	"Design contains a verilog module ('<string>' defined in file '<file-name>') and a vhdl entity ('<string>' defined in file '<file-name>') with same name.";
[117]	"Cannot use 'higher_capacity' and 'disallow_view_delete' simultaneously.";
[118]	"Disabling following rules that do not support large design processing";
[119]	"Working in Large Design Processing mode";
[120]	"Missing quote (\') for string `<string>'";
[121]	"Failed to find Init/Exec/Exit functions of Rule '<rule-name>'";
[122]	"Analyzer's Initialization failed";
[123]	"Running '<rule-name>' rule instead of obsolete '<obsolete-name>' rule.\n(Please update your script/goal)";
[124]	"Ignoring obsolete '<obsolete-name>' rule.\n(Please update your script/goal)";
[125]	"Following Black Box(es) has/have been found:<message> Note: This error is reported because 'nobb' option was specified.";
[126]	"Rule '<rule-name>' passed with 'lvpr' specification '<option>=<value>' is not registered";
[127]	"Failed to find <string> function '<func-name>' of '<policy-name>' policy";
[128]	"Policy '<policy-name>' requires SpyGlass Version <ver1> (or > <ver2>)";
[129]	"Policy '<policy-name>' is already registered";
[130]	"Could not open '<file-name>' file from '<path>' path because '<reason>'.";
[131]	"top <top-name> : No such design unit";

Message Number	Message
[132]	"Following Design units specified with <option-name> option were not found <string> (Reason: <reason>)";
[133]	"du <du-name> : No such design unit";
[134]	"Rule '<rule-name>' uses obsolete primitive '<primitive> Check c-primitive documentation on how to upgrade this rule";
[135]	"`<option-name>' option requires an argument";
[136]	"Unable to read configuration file: '<file-name>' [<string>] ";
[137]	"Cannot handle string containing more than 4096 characters on line <num> inside configuration file `<file-name>'";
[138]	"'<name>' should not be passed as argument for '<option>' option";
[139]	"Unrecognized configuration key `<key>' on line <num> inside configuration file `<file-name>'";
[140]	"Syntax error near `<string>' on line <num> inside configuration file `<file-name>'";
[141]	"Missing <assignment> assignment to a value on line <num> inside configuration file `<file-name>'";
[142]	"Syntax error - missing `=' on line <num> inside configuration file `<file-name>'";
[143]	"`<key>' key redefined on line <num> inside configuration file `<file-name>'";
[144]	"Incorrect value specified for `<string>' on line <num> inside configuration file `<file-name>' Use one of <string>";
[145]	"Missing PhysicalPath for LogicalLibName `<name>' on line <num> inside configuration file `<file-name>'";
[146]	"PhysicalPath redefined for library `<lib-name>' on line <num> inside configuration file `<file-name>'";
[147]	"Unable to open helpfile '<file-name>': <message>";
[148]	"Only simple configurations (specification of architecture for a top level entity) are supported for synthesis. So ignoring binding of instances in configuration '<name>' at line Number '<num>' in file '<file-name>";
[149]	"File <file-name>, Line <num>: <message>";
[150]	"Rule <rule-name>: File <file-name>, Line <num>: <message>";
[151]	"<message> : Unable to open <string> <file-name> for reading (Reason: <reason>)";

Message Number	Message
[152]	"Unable to load libraries specified by 'gateslib' option";
[153]	"Unable to load default gate lib - <name>";
[154]	"'<message>' : Unable to create directory (<dir-name>)";
[155]	"'<message>': is not a Directory. Directory name expected.";
[156]	"Same '<string>' file '<file-name>' specified multiple times. Ignoring the duplicate specification.";
[157]	"Negative or very-large value (<num> bytes) passed to '<function>' function";
[158]	"<message> : Unable to locate goal file in following search paths - <paths>";
[159]	"<message> : Unable to open goal file for reading (Reason: <reason>)";
[160]	"'handlefloatparam' specification is redundant. Floating parameters are automatically handled by SpyGlass.";
[161]	"Same file '<file-name>' specified multiple times.";
[162]	"'<message1>' : <message2>.";
[163]	"Files '<file1>' and '<file2>' are identical.";
[164]	"File '<file-name>', Line '<num>': High-Profile Rule '<rule-name>' specified in 'spyRegisterPolicy', not found in Policy '<policy-name>'";
[165]	"'<file-name>' is not a Verilog HDL source file. Please specify only Verilog HDL source files for 'verilog' language mode";
[166]	"'<file-name>' is not a VHDL source file. Please specify only VHDL source files for 'vhdl' language mode";
[167]	"'<option>' option is not supported on current platform (<platform>)";
[168]	"Overriding default memory manager for SpyGlass application to support perflog option. This option is not recommended to be used in production environment";
[169]	"Only '<string>' port retention specification is applicable for module '<module>'. Ignoring port retention specification for this module.";
[170]	"Please select only one design hierarchy for entity '%s'.";
[171]	"Most recent architecture '<architecture>' hierarchy is selected and rest all architectures of entity '<entity>' are ignored.";
[172]	"All configurations for the entity '<entity>' are ignored and architecture '<architecture>' hierarchy is selected.";
[173]	"Most recent configuration '<configuration>' hierarchy is selected for entity '<entity>'.";
[174]	"Configuration '<configuration>' hierarchy is selected for entity '<entity>'.";

Message Number	Message
[175]	"Option 'du' is not supported in the current SpyGlass Release.";
[176]	"'<option>s' option is not supported by spyglass_lc";
[177]	"'<message>' : Unable to open file (<file-name>)";
[178]	"sglib File `<file-name>' has a format which is not recognized by SpyGlass. Use spyglass library compiler to generate sglib file .";
[179]	"Top design Unit `<du-name>' cannot be specified as an ILM .";
[180]	"File '<file-name>', Line '<num>': Policy '<policy-name>' for file-label '<label> not registered.";
[181]	"File '<file-name>', Line '<num>': File-label '<label>' is already registered.";
[182]	"File '<file-name>', Line '<num>': File-label '<label>' not registered.";
[183]	"File '<file-name>', Line '<num>': File-label '<label>' cannot be registered for policy '<policy1>' in policy '<policy2>'";
[184]	"The file '<file-name>' is neither a valid verilog nor a valid vhdl file. Please specify only VHDL or Verilog source files to library compiler to provide cell definition externally";
[185]	"File '<file-name>', Line '<num>': Multiple files cannot be opened for the file-label '<label>'";
[186]	"File <file-name>, Line <num>: spyOverload ignored as rule '<rule-name>' is not registered";
[187]	"Rule <rule-name> has created a new vhdl attribute table handle, but has not restored back the original one.";
[188]	"'<message>' : Unable to open directory for reading (<string>)";
[189]	"Command-file '<file-name>' is empty.";
[191]	"Re-mapping of Standard VHDL Library '<lib-name>' is not allowed";
[192]	"Illegal key '<key>' specified for 'sortrule' option";
[193]	"Option 'psl' has been deprecated. Use option(s) 'psl_verilog' and/or 'psl_vhdl' to specify your Verilog and/or VHDL psl files respectively.";
[194]	"WEIGHT '<weight>' specified for overload of RULE '<rule-name>' is invalid.";
[195]	"Syntax error <string><string>.";
[196]	"Syntax error while specifying rule overload information for RULE '<rule-name>', <string> '<string>'.";
[197]	"Syntax error while defining new severity label for POLICY '<policy>', <string>";

Message Number	Message
[198]	"Syntax error while defining new severity label using 'define_severity'";
[199]	"Invalid severity class for POLICY '<policy>'.";
[200]	"User Configuration file '<file-name>' already read from SPYGLASS_HOME, user's HOME, CWD or custom path. Reading again may cause problems.";
[201]	"Configuration file '<file-name>' inappropriate.";
[202]	"'<message>': Cannot open config file for reading (<string>).";
[203]	"Rule '<rule-name>': PI 'sgPatternMatch' called with NULL name (First parameter) for type '<type>'.";
[204]	"PI 'sgPatternMatch' called with NULL name (First parameter) for type '<type>'.";
[205]	"Rule '<rule-name>': PI 'sgPatternMatch' called with NULL routine-name.\n The sgPatternMatch function parses the first token from the ruleInfo\n argument list (that is, the 9th argument to the spyRegisterRule function,\n and calls the PERL function named by that token.";
[206]	"PI 'sgPatternMatch' called with NULL routine-name.";
[207]	"Rule '<rule-name>': Undefined sub routine '<sub-routine>' passed to PI 'sgPatternMatch'.";
[208]	"Undefined sub routine '<sub-routine>' passed to PI 'sgPatternMatch'.";
[209]	"Rule <rule-name>, cannot use 'spyglass_static_auxi_rule' and 'spyglass_auxi_rule' simultaneously.";
[210]	"Duplicate registration for <string> '<name>' in policy '<policy>', earlier registered as <name> in policy '<policy>'. This registration will be ignored.";
[211]	"spyCreateRuleByCopy: Recursive calls detected. <string>'";
[212]	"Option 'opt_level' has been deprecated.";
[214]	"Invalid Hierarchy Separator Specified in SGDC file <file-name> at Line Number <num>.";
[215]	"Too many policies specified with option 'savepolicy'";
[216]	"Disabling save-restore as save-restore is not allowed with policy 'auto-verify";
[217]	"Disabling <name> as <string> is not allowed with option '<option>'";
[218]	"Option '<option>' is deprecated. Please set option 'enable_save_restore' to perform design save/restore";
[219]	"<string>: No such <message>";
[220]	"Incorrect CPRIM definition for CPRIM '<name>' found: DEPEND_RTL=<value> is not allowed for this type of CPRIM.";

Message Number	Message
[221]	"Incorrect value specified for `<name>' on line <num>d inside configuration file `<file-name>'";
[222]	"Policy '<policy-name>' - Severity label ('<label>') cannot have space in it.";
[223]	"Save/Restore feature is not supported with SGDC command '<command>'. Please remove Save/Restore related options from current run options.";
[224]	"Specified option '<option>' is not valid <string> SpyGlass option '<option>'";
[225]	"Rule '<rule-name>': Duplicate mapping for label '<label>'.";
[226]	"Rule '<rule-name>': Syntax error near '<name>', while defining message group use in rule's argument list.";
[227]	"Rule '<rule-name>': Missing argument after '<location>' in rule's argument list.";
[228]	"Rule '<rule-name>': '<name1>' cannot be used without '<name2>'.";
[229]	"<name> Template search directory '<dir>' doesn't exist";
[230]	"<name> Template search directory '<dir>' doesn't have read permission";
[231]	"Following template search path(s) contain 'templates' directory -\n\n%s";
[232]	"End of Command File `<file-name>' before =template(=cut) help termination";
[233]	"The 'filter' option is not applicable in the batch mode; use it for the GUI mode only.";
[234]	"Saving design database in directory '<directory>' ...";
[235]	"Restoring design database from directory '<directory>' ...";
[236]	"Following rules not being run (design database restored & 'skip_rules_for_fast_restore' is set):";
[237]	"Following rules can not run on restored design database. Hence, HDL being re-read:";
[238]	"Restoring design database is disabled as following rules need re-synthesis:";
[239]	"<string> file '<file-name>' contains <string>. It is recommended to define Waiver and other SGDC constraints in\ separate files because use-model for Waiver and other SGDC constraints are different.";
[240]	"SGDC file '<file-name>' contains <string>. It is recommended to define Waiver and other SGDC constraints in separate files because use-model for Waiver and other SGDC constraints are different. Please also note, waiver files are specified using 'waiver' option.";

Message Number	Message
[241]	"<string> file '<file-name>' is specified with option '<option>' as well. It is recommended to define Waiver and other SGDC constraints in separate files because use-model for Waiver and other SGDC constraints are different.";
[242]	"Could not open intermediate file `<file-name>' for writing. Reason: `<reason>'";
[243]	"No sglib specified in current run. At least one sglib should be specified.";
[244]	"Ignoring option '<option>' as it has no effect in current run mode.";
[245]	"Options 'est_mode' and '<option>' cannot be used together.";
[246]	"Multiple license checkout calls by a rule are not supported. Rule '<rule-name>' has already checked out '<string>' (version <version>, and is trying to checkout '<string>' (version <version>).";
[247]	"File: '<file-name>' Line: '<num>' - Invalid mode ('<mode>') specified for policy '<policy>'.";
[248]	"Policy set '<set>' (CLASSIC synthesis mode compatible) can not run with policy set '<set>' (EST mode compatible).";
[250]	"Recompiling the <dump> precompile dump in Work Directory '<path>/<dir>'['remove_work' option specified].";
[251]	"Updating the current <dump> precompile dump in Work Directory '<path>/<dir>'.";
[253]	"SpyGlass requires .sglib (or .lib) for given '<name>' for performing checks on post-layout netlist. Please provide appropriate SpyGlass gates library.";
[254]	"Key not specified in file '<file-name>' at lineNo <num>";
[255]	"Invalid Key '<key>' is specified in file '<file-name>' at lineNo <num>";
[256]	"Value is not specified for the Key '<key>' in file '<file-name>' at lineNo <num>";
[257]	"'{' is expected in file '<file-name>' at lineNo <num>";
[258]	"Unable to read File '<file-name>' [<string>] specified in '<name>' at lineNo <num>d";
[259]	"'sgdc' is not a valid SpyGlass Library Compiler option. If you want to apply waivers, please use 'waiver' option.";
[260]	"For a post-layout netlist, please set option 'enable_pgnetlist' along with <name> to perform SpyGlass analysis on it. Any <name> specified with a post-layout netlist is ignored without 'enable_pgnetlist' option.";
[261]	"Following '<specification>' specifications matches none of the design files passed to SpyGlass analysis (Please check for any typing mistakes) <string>";

Message Number	Message
[262]	"'stopdir <dir-name>' has been converted to 'stopfile '<file-name>'";
[263]	"Switch '<switch>' is redundant in a BATCH invocation";
[264]	" '<name>' is no longer supported";
[265]	"'nosavepolicy' option contains policies '<policies>' used by other goals of '<methodology>' methodology" "Other goals of '<methodology>' methodology having these policies would not be able to use "saved DB.";
[266]	"Unable to checkout license features '<license-features>'." "Other goals of '<methodology>' methodology requiring this feature would not be able to use "saved DB".";
[267]	"Too many options, total - <num>. Please check if options specified for SpyGlass analysis is correct.";
[268]	"<string1> '<string2>' is obsolete. Please run alternate <string3> '<string4>' of '<policy>' policy.";
[269]	"<string>: Invalid logical name specified in '<option>' option (it should resemble a valid verilog/vhdl name)";
[270]	"Multiline Comment found in file '<file-name>' from line <num1> to <num2>";
[271]	"Non Printable character found in file '<file-name>' at line <num>";
[272]	"Multiple specifications found for '<option1>' <option2> as <reason>. All specifications except the first one are ignored";
[273]	"Option '<option>' has been deprecated. Use 'option-name>' to specify physical libraries along with option 'enable_pgnetlist'.";
[274]	"The value '<value1>' for option '<option>' is no longer supported. Please use '<value2>' value instead.";
[275]	"'dump64bit' option would be removed in future release. Please use 'dump_all_modes' option instead. <string>";
[276]	"'<string>' <command> System Command could not be executed. SpyGlass Internal System Error: <message>";
[277]	"'<message>': Unable to open file (<file-name>). <string>";
[278]	"Verilog2001 will be enabled by default from 3.9.0 and therefore the '<switch>' switch will be withdrawn ";
[279]	"'<switch>' switch will be withdrawn in 3.9.0";
[280]	"'<file-name>' is not a valid vdb file (or corresponding smdb in spysch directory may be invalid/incompatible)";
[281]	"could not find a old vdb file for incremental mode";

Message Number	Message
[282]	"'}' is expected in file '<file-name>' at the end";
[283]	"<string> file '<file-name>' contains <value>. Waiver commands from this file will not be migrated because use-models for Waiver and other SGDC commands are different. Please use '-import' field of 'waive' command to migrate waive commands in actual rule checking mode.";
[284]	"Option '<option>' is not valid in DEF mode";
[285]	"<feature> feature is not supported with option '<option>'. Disabling all <options> related options.";
[286]	"File "<file-name>", Line <num>: Incorrect LICENSE_QUEUING_INTERVALS_IN_SECS values("<values>") specified. There is a possibility of following errors: 1. License Retry Interval is greater than License timeout. 2. License Retry Interval or/and License timeout is/are negative.";
[287]	"<string1> '<string2>' <string3> not allowed in DEF mode ";
[288]	"Option '<option>' will be ignored in DEF mode ";
[289]	"Option '<option>' has been deprecated.%s";
[290]	"Both 'verilog' and 'vhdl' language mode specified in current run. Please select appropriate language depending on RTL language. If RTL is a mix of verilog and vhdl languages, then please select 'mixed' language mode";
[291]	"Both 'policy='none' (<value>) and policy='<value1>' (<value2>) are used in the current run. Please choose one of the two as desired.";
[292]	"The option 'sgsyn_rtl_opt_level' should be passed with EST or ESYNTH mode only.";
[293]	"Rules corresponding to multiple synthesis mode selected in current run as follows -" "'<value1>' (CLASSIC synthesis mode)\n" "'<value2>' (EST mode)\n" "'<value3>' (ESYNTH mode)\n" "Please select rules of only single mode in a given run. To force one of the above modes, please ignore rules of other mode";
[294]	"Rules corresponding to multiple synthesis mode selected in current run as follows -" "'<string>' (<mode> mode)\n" " '<string>' (<mode> mode)\n" "Please select rules of only single mode in a given run. To force one of the above modes, please ignore rules of other mode";

Message Number	Message
[295]	"Following rule(s) ignored in current run as their mode (<mode>) is not compatible with selected mode (<mode>) -\n" " <message>";
[296]	"spyRegisterRule: Invalid mode ('<mode>') specified for rule '<rule-name>'.";
[297]	"Illegal or no argument format specified for message argument";
[298]	"Illegal or no value passed in bus merging information";
[299]	"No arguments provided for the intended bus merging";
[300]	"Invalid number of arguments(> <num>) passed for bus merging";
[301]	"No value specified for <string> in bus-merging";
[302]	"Invalid information passed with <value> in spyOverload";
[303]	"New message label <label> passed for spyOverload";
[304]	"Unable to create directory '<directory>'.\n" " (Reason:'<reason>')\n" " Related option : '<option>' specified at:\n" " File Name: <file-name>\n" " Line Number:<num>\n" " Please correct any typo or resolve the problem and then try again.";
[305]	"<message> : Value passed in 'libhdlfiles'/'libhdlf' option for library '<lib-name>' is a <string>. <name> name expected.";
[306]	"<message> : Unable to open <type> <file-name> passed in 'libhdlfiles'/'libhdlf' option for library '<library>' for reading.\n" " (Reason: <reason>)";
[307]=	"'<message1>' <message2> : File passed in 'libhdlfiles'/'libhdlf' option for library '<library-name>' is not a Verilog HDL source file." " Please specify only Verilog HDL source files for 'verilog' language mode";
[308]	"'<message1>' <message2> : File passed in 'libhdlfiles'/'libhdlf' option for library '<library-name>' is not a VHDL source file.\n" " Please specify only VHDL source files for 'vhdl' language mode";
[309]	"No template/policy is specified in current run." " Default template '<none>' i.e. option policy value 'none' selected for current run.";
[310]	"No template/policy is specified in current run." "Default template(s) <template> selected for current run.";
[311]	"Obsolete rule '<obsolete-rule>' has been replaced by built-in rule '<built-in-rule>'. Please update your script/goal accordingly.";

Message Number	Message
[312]	"Missing option '<option1>'. This option must be specified if '<option2>' option is being used";
[313]	"<string1> '<string2>' <string3> not supported in 'Dual Design Read' mode";
[314]	"In '<mode>' mode only a single top is allowed. Please specify top design unit using option 'top'";
[316]	"Auto-compilation of gates libraries failed. Please refer to `<file-name>' for details";
[319]	"Memory Allocation Failed.Running software in 64bit mode may solve the issue. Exiting ...";
[320]	"Waiting for lock for file <file-name>";
[321]	"Could not read dependency information from physical path <path>" "Reason: <reason>.";
[322]	"Could not write dependency information for physical path <path>" "Reason: <reason>.";
[323]	"Following built-in checks are performed only on libraries" " compiled in the current run as these checks require design" " to be re-parsed:<string>" " If you want to perform these checks on all RTL libraries," " please specify option 'force_compile' which recompiles" " all the libraries.";
[324]	"Policy '<policy>' optionally picked from '<methodology>' methodology for saving would not be saved" " Reason: Policy is not compatible with current set of user specified policies." " Goals of '<methodology>' methodology having this policy would not be able to use "saved DB".";
[325]	"Some SGDC sanity check rules running on rtl view have been disabled as they do not have restore cprims and 'no_rtl_sgdc_check' option is provided:";
[326]	"SpyGlass Design Database '<database-name>' can not be restored because:" " <reason>" " Saving Database again";
[327]	"File "<file-name>", Line <num>: Value '<value>' for configuration key '<key>' is no longer supported.";
[329]	"<string> SpyGlass will now run in backward compatibility mode. Please refer to InfoSglibVersionSummary violation for details";
[330]	"File "<file-name>", Line <num>: White-space found after backslash";

Message Number	Message
[332]	"Mixed mode policies being saved in current run";
[333]	"Unable to open file <string><file-name>in mode : '<mode>' (Reason: <reason>)";
[334]	"Unable to open directory <string><directory> (Reason: <reason>)";
[335]	"<string>: Option 'include_goal'/'inherit_goal' is supported inside a goal only. Current specification would be ignored";
[336]	"File <file-name>, Line <num>: Goal '<goal>' specified by <option> option is not of the same language as current language mode.";
[337]	"File <file-name>, Line <num>: Goal '<goal>' specified by <option> option can't be opened for reading (Reason: <reason>)";
[338]	"File <file-name>, Line <num>: Option '<option>' is supported inside user specified goal only and not inside an included/inherited goal. Please remove the specification";
[339]	"File <file-name>, Line <num>: File specified recursively";
[340]	"Multiple 'inherit_goal' specification found in goal <goal-name> as <specification> All specifications except the first one are ignored";
[341]	"Parameter '<param-name>' specified multiple times at following locations - <specifications> All specifications except the last would be ignored.";
[342]	"Rule/Group '<Rule/Group>' specified at <string> has been ignored due to the following 'ignorerule'(s) specifications - <specifications>";
[343]	"Rule/Group '<Rule/Group>' specified at <string> has been ignored as some of its parent group is ignored at following places - <specifications>";
[344]	"Some of the following rule(s)/group(s) of ruleGroup '<rule-group>' specified at <location> has been ignored due to following 'ignorerule'(s) specifications - <specifications>";
[345]	"Severity for '<rule-name>' defined multiple times for policy '<policy>' at following places <specifications> Only last severity specification would be used.";
[347]	"Multiple severity overload specifications found for rule '<rule-name>' (registered in language '<language>') in included/inherited goal and parent goal (<goal>) at following places <specifications> Only last severity class would be used.";

Message Number	Message
[348]	"Following incompatible severity overload specifications for rule '<rule-name>' (registered in language '<language>') in included/inherited goal and parent goal (<goal>) have been ignored -<specifications>.";
[349]	"Multiple specifications of 'include_goal' '<specifications>' found in parent goal (<goal>) at following places-<specifications> All specifications except the first would be ignored.";
[350]	"Option '<string>'<option> supported in Console/TCL shell only and not in classical SpyGlass flow";
[351]	"Option '<option1>' specified, so setting option '<option2>'";
[352]	"Option '<option1>' specified, but not setting option '<option2>', as already set";
[353]	"Option '<option>': Invalid value '<invalid-value>' specified, allowed values - <allowed-values>";
[354]	"<lib-name>: Logical library name specified in '<option>' option. Intermediate logical lib name is expected";
[355]	"libmap <specification>: Intermediate to physical mapping via option 'lib' is expected for intermediate library <lib-name>";
[356]	"Option '<option>' is not allowed inside '<file-name>' command file. Provide design read specific options only in 'af'/'bf' command files";
[357]	"Option '<option>' specified with true value but ignored, as only verilog files specified" and option 'enable_precompile_vlog' to enable verilog precompile dump is not specified";
[358]	"Option '<option>' specified with true value but ignored, as running without any design file";
[359]	"Following builtin rules were not enabled during design save but are enabled now by 'rules'/'addrules' options\n" " Violations for these rules (if any) would not be flagged" " To see such violation in restore, please enable these rules and save the database again" "<message>";
[360]	"Unsetting '<option>' option as no tech library specified with 'gateslib' option";
[361]	"Few builtin rules flagged certain violations on the design in save run." " These violations would not be flagged currently in restore run as option 'enable_save_restore_builtin' not specified. To see these violations, please specify option 'enable_save_restore_builtin'";

Message Number	Message
[362]	"Ignoring following include/inherit goal specification in goal <goal> : <specifications> As its already added in this goal at:<string>";
[363]	"<message: No such Verilog Library file (passed with 'v' option)";
[364]	"Restoring design database is disabled due to architectural difference";
[365]	"M-Bist policy cannot be run with any other policy. (<message>)";
[366]	"FLATDU2_WOL type rule-checking has been deprecated in 4.7 release." " Following rules registered with FLATDU2_WOL mode have been moved to FLATDU2_WL mode\n";
[367]	"LE Oscillation Count Limit <limit> reached for net <net-name> in cycle <cycle>;
[368]	"LE Oscillation Count Limit '<limit>' reached for some nets (perform a run with setting option 'DEBUG' with value 'le' to get these nets). These oscillating nets could be part of long chain of combinational elements OR combinational loops. If these nets are not part of combinational loops, use option 'net_osc_count_limit <new_limit>' to increase the oscillation count limit.\nNOTE: If there are combinational loops, then increasing this limit is not recommended as it might hit runtime.";
[369]	"LE Oscillation Count Limit '<limit>' reached for some nets (refer INFO <num>] for theses nets). These oscillating nets could be part of long chain of combinational elements OR combinational loops. If these nets are not part of combinational loops, use_option 'net_osc_count_limit <new_limit>' to increase the oscillation count limit. NOTE: If there are combinational loops, then increasing this limit is not recommended as it might hit runtime.";
[370]	"One or more design statistics assumption has failed. This might lead to unpredictable behavior. Please refer to "Design Statistics Table" in spyglass.log file for more details";
[371]	"File `<file-name>' is not present in the directory specified by option `<option>'. Please create a file with this name containing list of ignorerules for builtins to be ignored";
[372]	"'param' option specified multiple times for the entity/module “<entity/module>” generic/param “<generic/param>” using the last value: '<value>'";
[373]	"Running in OEM mode, can not run following non-oem policies - <policies>";
[374]	"Unsetting incremental message reporting as could not open old vdbfile (or corresponding smdb in spysch directory) - <string>";
[375]	"Unsetting incremental message reporting as no old vdb file given or found (or corresponding smdb in spysch directory may be absent)";

Message Number	Message
[376]	"No '<file-name>' file found, hence picking '<goal>' goal file" "All the rules will be registered in mixed language";
[377]	"In '<mode>' mode option 'top' is mandatory. Please specify single top design unit using option 'top'.";
[378]	"Report '<report-name>' has been specified with 'report' as well 'disable_report' options. SpyGlass has ignored 'disable_report' specification.";
[379]	"Report '<report-name>' is a default or crucial report. It can not be disabled.";
[380]	"Option 'print_sortoder_only' specified, SpyGlass will exit after printing sorted VHDL files.";
[381]	"Error while creating report '<report-name>': Could not open file '<file-name>'";
[382]	"Calling Perl function for report '<report-name>' as SMDB could not be connected";

Flattening Rules

Overview

This section describes the flattening warning rules. These rules report violations for the cases, such as the following:

- If a flattener API is called with NULL as an argument
- If a combinational loop is detected during simulation

FLAT_502

Flattener API is called with NULL as an argument

Language

Verilog, VHDL

Rule Description

SpyGlass flags this warning if you specify NULL as an argument to a flattener API.

Message Details

`(<API-name>): called with null object. Please contact Atrenta Support.`

Severity

INTERNAL_WARNING

FLAT_503

Unstable combination loop detected during simulation

Language

Verilog, VHDL

Rule Description

SpyGlass reports this warning if a combinational loop is detected during simulation.

The CombLoop rule detects combinational loops in the design. To run the CombLoop rule, run any goal that contains the CombLoop rule. You must remove these loops before running the rules that trigger simulation.

Message Details

```
Unstable combinational loop detected (in simulation cycle -
<cycle-num>) with one of the offending net being '<net-name>'.
Please fix all combinational loops in your design
```

Severity

Warning

FLAT_504

Macro logic evaluation done partially as its size crosses the threshold limit

Language

Verilog, VHDL

Rule Description

SpyGlass reports this warning when the size of a synthesis macro exceeds the limit specified by the following command:

```
set_option define_cell_sim_depth <threshold-limit>
```

During logic evaluation, such macros are partially simulated till the threshold limit is reached. The default threshold limit is 10, which is the data bus size of 1024.

To completely simulate such huge macros for improving accuracy of logic simulation, increase the threshold limit by using the `define_cell_sim_depth` command.

You can increase the threshold limit during evaluation for the following synthesis macros by using the `define_cell_sim_depth` command:

- `M_RTL_ARITH_SHIFT`
- `M_RTL_MUX_N`
- `M_RTL_PRIM_MUX`
- `M_RTL_LSHIFT`
- `M_RTL_RSHIFT`
- `M_RTL_SHIFT`
- `M_RTL_ROTATE`

Message Details

```
Logic evaluation for macro instance '<inst-name>' (master:
<master-name>) inside module '<module-name>' done partially as
its input(s) data bus size '<bus-size>' exceeds the threshold
limit of '<limit>'. To improve the accuracy of logic evaluation
```

'set_option define_cell_sim_depth %d'

Severity

Warning

SDC-to-SGDC Translation Built-In Rules

Overview

The rules of this category report violations during the stage of translating design attributes from the SDC format to the SGDC format by using the `sdc2sgdc` command.

The translated SGDC attributes are used during SpyGlass analysis.

SDC_Mapping01

Reports SDC command mapping with the corresponding command in the generated SGDC

Language

Verilog, VHDL

Rule Description

The SDC_Mapping01 rule generates a spreadsheet that shows a mapping between SDC commands and their corresponding command in the generated SGDC file.

Message Details

`Mapping of SDC Commands Generated`

Severity

Info

Example

The following figure shows the example of a spreadsheet when you double-click on a violation of this rule:

	SDC Command	SDC FileName:LineNumber	Mapped SGDC Command
21	create_clock	clock.sdc:18	clock
22	create_clock	clock.sdc:23	clock
23	create_clock	clock.sdc:24	clock
24	set_case_analysis	clock.sdc:57	set_case_analysis
25	set_input_delay	clock.sdc:50	abstract_port
26	set_input_delay	clock.sdc:51	abstract_port
27	set_output_delay	clock.sdc:54	output
28	set_clock_sense	clock.sdc:60	clock_sense
29	set_clock_group -logically_exclusive	clock.sdc:44	cdc_false_path

FIGURE 1. Spreadsheet of the SDC_Mapping01 Rule

SDC2SGDC_WRN01

Multiple clocks defined on an object in SDC file without using -add option.

Language

Verilog, VHDL

Rule Description

The SDC2SGDC_WRN01 rule flags if multiple clocks are defined on an object in the SDC file without using the `-add` option.

In this case, all the clocks are overwritten by the last specified clock.

Message Details

The following message appears if multiple clocks are defined on an object in the SDC file without using the `-add` option:

```
Multiple clocks defined for object '<obj-name>'. Continuing the analysis with the last specified clock at line no. <line-num> of file <file-name>
```

Severity

Warning

SDC2SGDC_STX01

Multiple clock modes defined on an object in SDC file using -add option.

Language

Verilog, VHDL

Rule Description

The SDC2SGDC_STX01 rule flags if multiple clock modes are defined on an object using the `-add` option in the SDC file.

To perform deterministic clock-domain crossing analysis, a single operational mode must be used. Specify a single-mode SDC file instead of multi-mode SDC file.

Message Details

The following message appears if multiple clock modes are defined on an object using the `-add` option in the SDC file:

```
Multiple clock modes defined for object '<obj-name>' using -add
in conjunction with 'create_clock' constraint in SDC file
```

Severity

Error

SGDC_sdcschema01

Setup rule to check multiple modes for sdc_data constraints

Language

Verilog, VHDL

Rule Description

The SGDC_sdcschema01 rule flags if multiple modes are specified for the current design through the `sdc_data` constraint.

In such cases where you specify multiple modes, you can specify the mode to be translated into SGDC by using the `sdc2sgdc_mode` project file command. If you do not specify this command, the first mode specified by the `sdc_data` constraint is translated into SGDC and the remaining modes are ignored.

Message Details

The following message appears if multiple modes are specified for the current design through the `sdc_data` constraint:

```
Mode "<current-mode>" is different from previous "<previous-mode>" mode (specified at file "<SGDC-file-name>", line <line-num>) for the same current_design "<design-name>"
```

Severity

Error

Domain_Conflict01

Reports domain conflicts in the SDC-to-SGDC flow

Language

Verilog, VHDL

Rule Description

The *Domain_Conflict01* rule reports the clocks for which a conflict occurred during domain computation in the SDC-to-SGDC flow.

For cyclic clocks, use the `assign_domain_conflict_clocks` parameter to assign the same domain in the `sta_compliant` mode.

To debug the violation of this rule, refer to the reports in the *spyglass_reports/sdc2sgdc* directory.

Message Details

```
Clocks "<clock-name-1>" and "<clock-name-2>" are assumed
asynchronous because set_clock_groups overrides
set_clock_uncertainty


For design <top-name>, <number-of-cyclic-dependency> cyclic
dependency exists between the clocks, same domain is assumed
(sta mode)


For design <top-module-name>, <number-of-cyclic-dependency>
cyclic dependency exists between the clocks, no domain is
assumed (pessimistic mode)
```

Severity

Error

Domain_Matrix01

Generates a spreadsheet to show the domains inferred for clocks

Language

Verilog, VHDL

Rule Description

The *Domain_Matrix01* rule generates a spreadsheet showing clocks relationship and the inferred domain depending on the mode specified by the `sdc_domain_mode` parameter.

NOTE: *In the domain matrix spreadsheet, both rows and columns represent clocks and clock domains in a symmetrical manner. Set the show_domain_in_matrix_view parameter to 'no' to show the clock domains in the Domain column only.*

To debug the violation of this rule, refer to the reports in the *spyglass_reports/sdc2sgdc* directory.

Message Details

```
Domain Matrix for design <top-design-name>, based on the
individual constraints defined in the SDC as inferred in the "
<Sta | Pessimistic | Strict> mode"
```

Severity

Error | Fatal | Data

Domain_Missing01

Report clocks for which domain could not be computed from SDC constraints

Language

Verilog, VHDL

Rule Description

The *Domain_Missing01* rule reports the clocks for which domain could not be inferred from SDC constraints.

For such clocks, a domain is assigned based on the mode specified by the `sdc_domain_mode` parameter.

To debug the violation of this rule, refer to the reports in the *spyglass_reports/sdc2sgdc* directory.

Message Details

`Missing domain for <num> clock(s)`

Where *<num>* is the number of clocks for which domain is missing.

Severity

Error | Fatal

SPEF Checking Rules

Overview

This section describes the SPEF checking rules.

SPEFSTX_14

The -spef_topname value cannot be coupled with -instname or -modname value.

Language

Verilog, VHDL

Rule Description

The SPEFSTX_14 rule reports a violation if the value specified to the `-spef_topname` argument of the `spef_data` constraint is same as the value of the `-instname` or `-modname` arguments.

Violation Details

```
The -spef_topname value cannot be coupled with -instname or
-modname value
```

Severity

Error

SPEFSTX_15

Net specified in the R_NET section is not found in the design.

Language

Verilog, VHDL

Rule Description

The SPEFSTX_15 rule reports a violation when a net specified in the R_NET section is not found in the design.

In such cases, no object model is populated for this R_NET.

Violation Details

```
Net '<net-name>' specified in the R_NET section is not found in
the design
```

Severity

Error

SPEFWRN_4

Net node for a D_NET is part of the *CONN section but is not a part of fanin/fanout of the net

Description

The SPEFWRN_4 rule reports a violation when a net node for a D_NET is part of the *CONN section but is not a part of fanin/fanout of the net.

Language

Verilog, VHDL

Messages and Suggested Fix

[WARNING] For D_NET <net_name>, net node '<node_name>' though part of the *CONN section, is not a part of the fanin/fanout of the net.

Severity

Warning

SPEFWRN_5

Net node for a D_NET is part of fanin/fanout of the net but is not part of the *CONN section

Verilog, VHDL

Description

The *SPEFWRN_5* rule reports a violation when a net node for a D_NET is part of fanin/fanout of the net but is not part of the *CONN section.

Language

Verilog, VHDL

Messages and Suggested Fix

[WARNING] For D_NET <net-name>, net node '<node-name>' though part of the fanin/fanout of the net, is not a part of the *CONN section.

Severity

Warning

SpyGlass Abstraction Flow Messages

Overview

The rules of this category report violations in the SoC flow.

SGDC_abstract_port01

Existence check for '-module' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port01* rule reports a violation if the module specified by the `-module` argument of the `abstract_port` constraint does not exist in the current design.

To fix this violation, update the `-module` argument of the `abstract_port` constraint by specifying the name of the module that exists in the current design.

Message Details

The following message appears if the module *<module-name>* is not instantiated in the design *<design-name>*:

```
Constraint 'abstract_port': '<module-Name>' [SubModule] is
never instantiated within environment '<name>'
```

Example

Consider the following `abstract_port` constraint:

```
abstract_port -module blk -ports P1 -clock CLK1
```

In this case, the *SGDC_abstract_port01* rule reports a violation if the `blk` module does not exist in the design.

Severity

Fatal

SGDC_abstract_port02

Existence check for '-port' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port02* rule reports a violation if the port name specified in the `-ports` argument of the `abstract_port` constraint does not match with any of the ports of the module specified by the `-module` argument of this constraint.

To fix this violation, update the `-port` argument of the `abstract_port` constraint to specify a port that exists on the module specified by the `-module` argument.

Message Details

The following message appears when the port *<port-name>* is not the port of the module *<module-name>*:

```
Constraint 'abstract_port': '<port-name>' [TopPort] not found on/within module '<module-name>'
```

Example

Consider the following `abstract_port` constraint:

```
abstract_port -module blk -ports P1 -clock CLK1
```

In this case, the *SGDC_abstract_port02* rule reports a violation if the P1 port does not exist in the port list of the `blk` module.

Severity

Fatal

SGDC_abstract_port03

Existence check for non-hierarchical name in '-clock' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port03* rule reports a violation if the clock name specified in `-clock` argument of the `abstract_port` constraint is not a port or a net in the current design.

Message Details

Message 1

The following message appears when the clock *<clock-name>* specified by the `-clock` argument of the `abstract_port` constraint is not found in the module *<module-name>*:

[FATAL] Constraint 'abstract_port': '<clock-name>' specified with '-clock' field not found on/within module '<module-name>'.

To fix this violation, specify a correct hierarchical clock name in the `-clock` argument so that it matches with a clock in the module specified by the `-module` argument of the `abstract_port` constraint.

Message 2

The following message appears when a non-hierarchical clock name *<clock-name>* specified by the `-clock` argument of the `abstract_port` is not found in the module *<module-name>*:

[INFO] Constraint 'abstract_port': '<clock-name>' specified with '-clock' field not found on/within module '<module-name>'.Considering it as virtual clock

To fix the violation, identify the violating constraint and check if the specified clock name is a virtual clock.

If it is a virtual clock, no fix is required. If it is not a virtual clock, specify a correct clock name in the *-clock* argument of the `abstract_port` constraint.

Example

Consider that the `blk` module in a design does not contain the `CLK1` clock. In this case, the *SGDC_abstract_port03* rule reports a violation if you specify the following `abstract_port` constraint:

```
abstract_port -module blk -ports P1 -clock CLK1
```

Severity

Fatal | Info

SGDC_abstract_port04

Check for vector name in '-clock' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port04* rule reports a violation if a clock specified in the `-clock` argument of the `abstract_port` constraint is a part-select or full vector of a block port.

Messages Details

The following message appears if the clock *<clock-name>* specified by the `-clock` argument of the `abstract_port` constraint is a vector net for the module *<module-name>*:

```
Constraint 'abstract_port': '<clock-name>' specified with '-clock' field for module '<module-name>' must not be a vector net
```

To fix this violation, modify the `-clock` argument of the `abstract_port` constraint to specify a clock that is not a vector net.

Example

Consider that the `CLK1` clock is a vector net in the `blk` module. In this case, the *SGDC_abstract_port04* rule reports a violation if you specify the following constraint:

```
abstract_port -module blk -ports P1 -clock CLK1
```

Severity

Fatal

SGDC_abstract_port05

Check whether wildcard name in -clock field matches with multiple port

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port05* rule reports a violation if the wildcard name specified in the `-clock` argument of the `abstract_port` constraint matches with multiple block ports.

Message Details

The following message appears when a wildcard clock name *<clock-name>* specified by the `-clock` argument of the `abstract_port` constraint matches with multiple ports of the module *<module-name>*:

```
Constraint 'abstract_port': '<clock-name>' specified with '-clock' field matches more than one port of module '<module-name>'
```

To fix this violation, modify the wildcard clock name in the `-clock` argument of the `abstract_port` constraint so that it uniquely matches with one of the ports of the module specified by the `-module` argument.

Example

Consider that the `blk` module has four ports: `c1`, `c2`, `c3`, and `p1`. Now consider that you specify the following constraint:

```
abstract_port -module blk -ports addr -clock "c*"
```

In this case, the *SGDC_abstract_port05* rule reports a violation as the wildcard expression specified in the `-clock` argument matches with the `c1`, `c2`, and `c3` ports of the `blk` module.

Severity

Fatal

SGDC_abstract_port06

Existence check for '-combo' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port06* reports a violation if the value specified by the `-combo` argument of the `abstract_port` constraint is other than `yes`, `no`, or `unknown`.

Message Details

The following message appears if the value specified in the `-combo` argument of the `abstract_port` constraint has a value other than `yes`, `no`, or `unknown`:

```
Constraint 'abstract_port': '-combo' field can only be 'yes' or 'no' or 'unknown'
```

To fix this violation, specify the value `yes`, `no`, or `unknown` in the `-combo` argument of the `abstract_port` constraint.

Example

Consider the following constraint:

```
abstract_port -module block1 -ports in1 -clock clk1 -combo null
```

In this case, the *SGDC_abstract_port06* rule reports a violation because the `-combo` argument is assigned a value other than `yes`, `no`, or `unknown`.

Severity

Fatal

SGDC_abstract_port07

Existence check for non-hierarchical name in '-from' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port07* rule reports a violation if clock names specified in the `-from` argument of the `abstract_port` constraint does not exist as a port or a net in the current design.

Message Details

Message 1

The following message appears when the hierarchical clock name <*clock-name*> specified in the `-from` argument of the `abstract_port` constraint is not found in the module <*module-name*>:

```
[FATAL] Constraint 'abstract_port': '<from-clock-name>'
specified with '-from' field not found on/within module
'<module-name>'
```

To fix this violation, specify a correct hierarchical clock name in the `-from` argument of the `abstract_port` constraint so that it matches with a port or a net in the current design.

Message 2

The following message appears when the non-hierarchical clock <*clock-name*> specified in the `-from` argument of the `abstract_port` constraint is not found in the module <*module-name*> and is considered as a virtual clock:

```
[WARNING] Constraint 'abstract_port': '<from-clock-name>'
specified with '-from' field not found on/within module
'<module-name>'.Considering it as virtual clock
```

To fix this violation, specify a correct non-hierarchical clock name in the -

`from` argument of the `abstract_port` constraint so that the specified clock it is found in the module.

Example

Consider the design block `block1` that contains the port `out1`. This port is driven from the hierarchical source clock `b1.t1.ck`. However, this hierarchical clock source is not present in the design.

Now consider that you specify the following constraint:

```
abstract_port -module block1 -ports out1 -clock clk1 -from
"b1.t1.ck" -to clk1 -sync inactive
```

In this case, the *SGDC_abstract_port07* rule reports a fatal violation if the hierarchical clock is not found in the module.

Severity

Fatal | Info

SGDC_abstract_port08

Check for vector name in '-from' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port08* rule reports a violation if the port specified in the `-port` argument of the `abstract_port` constraint is a part-select or full vector of a block port.

Message Details

The following message appears if the port *<port-name>* specified in the `-port` argument of the `abstract_port` constraint is a part-select or full vector of a block port:

```
Constraint 'abstract_port': '<port-name>' specified with '-from' field for '<module-name>' module must not be a vector net
```

Severity

Fatal

SGDC_abstract_port10

Checks whether wildcard name in -from argument matches with multiple ports

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port10* rule reports a violation if the wildcard name specified in the `-from` argument of the `abstract_port` constraint matches with multiple block ports.

Message Details

The following message appears when a wildcard clock name *<clock-name>* specified by the `-from` argument of the `abstract_port` constraint matches with multiple ports of the module *<module-name>*:

```
Constraint 'abstract_port': '<clock-name>' specified with '-from' field matches more than one port of module '<module-name>'
```

To fix this violation, modify the wildcard clock name in the `-from` argument of the `abstract_port` constraint so that it uniquely matches with one of the ports of the module specified by the `-module` argument.

Example

Consider the design blocks `block1` and `block2` with the clock ports `CK1` and `CK2`, respectively. Also consider another block, `block3`, with the clock ports `CK1` and `CK2`.

Now consider that you specify the following constraint:

```
abstract_port -module "block*" -ports in1 -clock testclock -from "CK*" -to testclock -sync inactive
```

In this case, the *SGDC_abstract_port10* rule reports a violation for the `block3`.

Severity

Fatal

SGDC_abstract_port11

Reports incorrectly specified clock name in the -to argument of the abstract_port constraint

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port11* rule reports a violation if the clock specified in the `-to` argument of the `abstract_port` constraint does not exist as a port or net in the current design.

Message Details

Message 1

The following message appears when the hierarchical clock name <*clock-name*> specified by the `-to` argument of the `abstract_port` constraint is not present in the module <*module-name*>:

[FATAL] Constraint 'abstract_port': '<clock-name>' specified with '-to' field not found on/within module '<module-name>'

To fix this violation, specify a correct hierarchical clock name in the `-to` argument of the `abstract_port` constraint so that this clock matches with a port or net in the module specified by the `-module` argument of this constraint.

Message 2

The following message appears when the non-hierarchical clock name <*clock-name*> specified by the `-to` argument of the `abstract_port` constraint is not present in the module <*module-name*>:

[WARNING] Constraint 'abstract_port': '<clock-name>' specified with '-to' field not found on/within module '<module-name>'. Considering it as virtual clock

To fix this violation, specify a correct non-hierarchical clock name in the `-to` argument of the `abstract_port` constraint so that this clock

matches with a port or net in the module specified by the `-module` argument of this constraint.

Example

Consider the design block block1 containing the port `out1` that is driven from the hierarchical clock `b1.t1.ck`. This hierarchical clock source is non-existent in the design.

Now consider that you specify the following constraint:

```
abstract_port -module block1 -ports out1 -clock clk1 -to
"b1.t1.ck" -from clk1 -sync inactive
```

In this case, the *SGDC_abstract_port11* rule reports a violation because the hierarchical clock is not found in the module as it does not exist in the design.

Severity

Fatal | Warning

SGDC_abstract_port12

Check for vector name in '-to' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port12* rule reports a violation if the port specified in the `-to` argument of the `abstract_port` constraint is a part-select or full vector of a block port.

Message Details

The following message appears if the port *<port-name>* specified in the `-to` argument of the `abstract_port` constraint is a part-select or full vector of a block port:

```
Constraint 'abstract_port': '<port-name>' specified with '-to' field for '<module-name>' module must not be a vector net
```

Severity

Fatal

SGDC_abstract_port13

Check whether wildcard name in the-to argument matches with multiple ports

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port13* rule reports a violation if the wildcard name specified in the `-to` argument of the `abstract_port` constraint matches with multiple block ports.

Message Details

The following message appears when a wildcard clock name *<clock-name>* specified by the `-to` argument of the `abstract_port` constraint matches with multiple ports of the module *<module-name>*:

```
Constraint 'abstract_port': '<clock-name>' specified with '-to' field matches more than one port of module '<module-name>'
```

To fix this violation, modify the wildcard clock name in the `-to` argument of the `abstract_port` constraint so that it uniquely matches with one of the ports of the module specified by the `-module` argument.

Example

Consider the design blocks `block1` and `block2` with the clock ports `CK1` and `CK2`, respectively. In addition, there exists another block `block3` with the clock ports `CK1` and `CK2`.

Now consider that you specify the following constraint:

```
abstract_port -module "block*" -ports in1 -clock tclk -from testclock -to "CK*" -sync inactive
```

In this case, the *SGDC_abstract_port13* rule reports a violation for the `block3` module because the wildcard expression, `CK*`, specified in the `-to` argument matches with multiple ports, `CK1` and `CK2`.

Severity

Fatal

SGDC_abstract_port14

Existence check for '-delay' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port14* rule reports a violation if the value specified with the `-delay` argument of `abstract_port` constraint is not an integer or an integer less than two.

Message Details

The following message appears if the value specified with the `-delay` argument of `abstract_port` constraint is not an integer or an integer less than two:

```
Constraint 'abstract_port': Number of required flip-flops specified with '-delay' can only be greater than 1
```

Severity

Fatal

SGDC_abstract_port15

Existence check for '-seq' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port15* rule reports a violation if the value specified in the `-seq` argument of the `abstract_port` constraint is other than `yes` or `no`.

Message Details

The following message appears if the `-seq` argument of the `abstract_port` constraint is set to a value other than `yes` or `no`:

```
Constraint 'abstract_port': '-seq' field can only be 'yes' or 'no'
```

To fix this violation, set the value of the `-seq` argument of the `abstract_port` constraint to either `yes` or `no`.

Example

Consider the following constraint:

```
abstract_port -module block1 -ports in1 -clock clk1 -seq 1
```

In this case, the *SGDC_abstract_port15* rule reports a violation because the `-seq` argument is set to a value other than `yes` or `no`.

Severity

Fatal

SGDC_abstract_port16

Existence check for '-related_ports' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port16* rule reports a violation if the input port specified by the `-related_ports` argument of the `abstract_port` constraint does not exist in module specified by the `-module` argument of this constraint.

Message Details

The following message appears if the input port *<port-name>* specified by the `-related_ports` argument of the `abstract_port` constraint is not present in the module *<module-name>*:

```
Constraint 'abstract_port': '<port-name>'[TopPort] not found on/within module '<module-name>'
```

To fix this violation, specify an existing input port in the `-related_ports` argument of the `abstract_port` constraint for the specified module.

Example

Consider the design block `block1` that has the output port `out` with a sequential path from the input port `in_3`.

Now consider that you specify `in3` instead of `in_3` in the `-related_ports` argument, as shown below:

```
abstract_port -module block1 -ports out clock clk1 -related_ports in3
```

In this case, the *SGDC_abstract_port16* rule reports a violation because the `in3` port is not found in the module.

Severity

Fatal

SGDC_abstract_port18

Value check for '-sync' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The SGDC_abstract_port18 rule reports a violation if the value specified in the `-sync` argument of the `abstract_port` constraint is other than `active` or `inactive`.

Message Details

The following message appears if the value specified in the `-sync` argument of the `abstract_port` constraint is other than `active` or `inactive`:

```
Constraint 'abstract_port': '-sync' field can only be 'active' or 'inactive'
```

To fix this violation, specify `active` or `inactive` in the `-sync` argument of the `abstract_port` constraint.

Example

Consider the following constraint:

```
abstract_port -module block1 -ports qual -clock clk2 -from clk1 -to clk2 -sync 1
```

In this example, the *SGDC_abstract_port18* rule reports a violation because the value of the `-sync` argument is other than `active` or `inactive`.

Severity

Fatal

SGDC_abstract_port21

Port specified with 'abstract_port' constraint does not exist in the design

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port21* rule reports a violation if the port specified in `abstract_port` constraint does not exist in the design.

This violation appears in case of parameterized instantiation of a block.

Message Details

The following message appears if the port *<port-name>* specified in the argument *<argument-name>* of the `abstract_port` constraint does not exist in the parameterized abstracted block *<block-name>*:

```
Value '<port-name>' specified with field '<argument-name>' of abstract_port is not found in a personality of parameterized abstracted block '<block-name>'. Please refer SoC flow documentation
```

Severity

Fatal

SGDC_abstract_port22

Existence check for '-combo_ifn' field of constraint 'abstract_port'

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_port22* rule reports a violation if the port specified in the `-combo_ifn` argument of the `abstract_port` constraint does not exist as a port of the module specified in the `-module` argument.

Message Details

The following message appears if the port specified in the `-combo_ifn` argument of the `abstract_port` constraint does not exist as a port of the module specified in the `-module` argument:

```
Constraint 'abstract_port': <message>
```

Severity

Fatal

SGDC_Abstract01

Rule informs that block abstracted view has been used

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract01* rule reports an information message indicating that the generated abstract view has been used in the SoC flow.

Message Details

This rule reports the following message:

`Abstracted view has been read for block '<block-name>'`

Severity

Info

SGDC_Abstract02

Rule informs that block abstracted view has not been provided for all the products

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract02* rule reports a violation indicating that the abstract view is not provided for all the product rules being run in the SoC flow.

Message Details

Message 1

This rule reports the following message if the abstract view of the block *<block-name>* is not provided or the abstract_port constraint is not specified for the abstract view:

```
For product '<product-name>', abstract view for block
'<block-name>' has not been provided or does not contain any
abstract_port constraint or requires re-generation
```

Message 2

This rule reports the following message if the abstract view of the parameterized block *<block-name>* is not provided or the abstract_port constraint is not specified for the abstract view:

```
For product '<product-name>', abstract view for block
'<block-name>' has either not been provided or does not contain
any abstract_port constraint. Please generate proper abstract
view for this block using 'set_option param {<parameter-name>}'
specification
```

In this case, the Verilog or VHDL parameters should be specified for the block while generating the abstract view.

Severity

Warning

SGDC_Abstract03

Rule informs that interface has not been provided for an abstracted block

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract03* rule reports a violation if the block interface information is not provided for the abstract view of a block in the SoC flow.

It is mandatory to specify the block interface information in the SoC flow.

Message Details

This rule reports the following message if the block interface information is missing for the abstract view of the block *<block-name>*:

```
RTL interface required for abstraction flow has not been
provided for block '<block-name>'
```

Severity

Fatal

SGDC_Abstract04

Rule informs that in SoC flow, license is available only for CDC product, hence other product rules will get disabled.

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract04* rule reports a violation if in the SoC flow, you try to run the rules of non CDC products in the absence of the `soc_abstraction` license.

In the absence of this license, the rules of all the non CDC products get disabled in the SoC flow.

Message Details

This rule reports the following message if you try to run the rules of non CDC products in the absence of the `soc_abstraction` license:

```
Failed to check out license feature '<license-name>'. SpyGlass would disable rules of '<product-names>' products.
```

Severity

Warning

SGDC_Abstract05

Rule informs that interface has not been provided for an abstracted block

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract05* rule reports a violation if you provide an abstract view for the top-level module in the SoC flow.

You must specify the abstract view and block interface information for the module that is not the top-level module.

Message Details

This rule reports the following message if you provide the abstract view for a top-level module:

```
Abstract view provided for top module '<top-mod-name>'
```

Severity

Fatal

SGDC_Abstract07

Rule informs that in SoC flow, only single product policy is allowed to perform SoC analysis.

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract07* rule reports a violation if you run rules of multiple products to perform SoC analysis.

In the SoC flow, you must run rules of a single product corresponding to which the abstract view is to be used.

Message Details

This rule reports the following message if you run rules of multiple products in the SoC flow:

```
Single Product policy should be specified for performing SoC run
```

Severity

Fatal

SGDC_Abstract08

Rule reports fatal violations encountered in SoC flow.

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract08* rule reports a violation if the generated abstract view has issues related to it ports or pins. These issues are described in *Message Details*.

Message Details

This rule reports the following messages based on the port or pin related issues in the generated `abstract_port` constraint:

- `abstract_port`: Invalid port(s) specification(s) '*<port_name>*' provided for module *<module_name>*.
- Constraint '`abstract_port`': port '*<port_name>*' provided with `-ports` field not found in module '*<module_name>*'.
- Constraint '`abstract_port`': invalid value provided with `-related_ports`.
- Constraint '`abstract_port`': '`-inst_master`' field can only have one of following values:

  RTL_FD, RTL_FDPC, RTL_FDPCE, RTL_FDCP, RTL_FDCPE, RTL_FDC, RTL_FDP, RTL_FDE, RTL_FDPE, RTL_FDCE, RTL_LD, RTL_LDPC.
- Constraint '`abstract_port`': invalid input pin value '*<pin_name>*' provided for `-inst_pin field`.
- Constraint '`abstract_port`': invalid output pin value '*<pin_name>*' provided for `-inst_pin` field.

Severity

Fatal

SGDC_Abstract09

Rule informs that in SoC flow, user should manually package other files referred in original block-level constraint file(s).

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract09* rule reports a violation if you do not specify additional files that are referred in original block-level constraint files for abstract view generation in the SoC flow.

Specify these additional files by using any of the following constraints:

- `include <sgdc_file>`
- `sgdc -import <block_name> <block_constraint_file>`
- `sdc_data -type <sdc_file>`
- `power_data -format <cpf|upf> -file <cpf/upf file>`
- `activity_data -format <vcd/fsdb/saif> -file <file>`

Message Details

This rule reports the following messages if you do not specify additional files that are referred in original block-level constraint files for abstract view generation:

```
Block '<block-name>' SGDC file(s) refers additional files which needs to be manually packaged along with the block-level SGDC file(s) in directory '<directory>'
```

Severity

Warning

SGDC_Abstract10

Rule informs that in SoC flow for a parameterized block default current design specification is being picked which can cause improper results. User should generate abstract views for the block using reported param values.

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract10* rule reports a violation if you do not generate the abstract view of a parameterized block by using the `set_option param <parameter-name>` project file command in the SoC flow.

By default, SpyGlass generates the abstract view of a parameterized block by using the default design specifications. This can generate improper results in the SoC flow. Therefore, you must generate abstract views for such parameterized blocks by using the reported parameter values.

Message Details

This rule reports the following message if you do not generate the abstract view of a parameterized block by using the `set_option param <parameter-name>` project file command:

```
SoC results might not be proper as default abstract view has been used for parameterized block '<block-name>'. Please generate proper abstract view for this block using 'set_option param {<parameter-name>}' specification
```

Severity

Warning

SGDC_Abstract11

Rule informs that SoC Validation may not be valid, since the block was not clean during product's Abstract Model Generation.

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract11* rule reports a violation if the block using which the abstract view is to be generated is not SpyGlass clean, that is, SpyGlass errors and warnings for such block are not resolved.

Message Details

This rule reports the following message for a block that is not SpyGlass clean:

```
SoC Validation may not be valid, since the block was not clean
during <product-name> Abstract Model Generation. Block
'<block-name>' had <num>.
```

Where *<num>* refers to the number of errors and warnings generated for the block using which its abstract view is to be generated in the SoC flow.

Severity

Warning

SGDC_Abstract16

Block instance is black boxed

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract16* rule reports an informational message to indicate that a block instance is black boxed as it is specified in the `-bbox_instance_list` argument of the `sgdc -import` constraint.

Message Details

This rule reports the following message:

```
Block instance '<inst-name>' (block '<block-name>') bboxe'd as specified in -bbox_instance_list
```

Severity

Info

SGDC_Abstract17

Same instances are specified in the -instance_list and -bbox_instance_list arguments of the sgdc -import constraint.

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract17* rule reports a violation when the same block instances are specified in the `-instance_list` and `-bbox_instance_list` arguments of the `sgdc -import` constraint.

Message Details

This rule reports the following message:

```
Block instance '<inst-name>' (block '<block-name>') specified in both -bbox_instance_list and -instance_list, and would be bboxe'd
```

Severity

Info

SGDC_Abstract18

Block instances are synthesized as no block SGDC file is specified for such instances

Language

Verilog, VHDL

Rule Description

The *SGDC_Abstract18* rule reports a violation if the given instances of a block are completely synthesized (while a few others are abstracted) as no block SGDC file is given for such instances

Message Details

This rule reports the following message:

```
Instances '<instances>' of block '<block-name>' have been fully synthesized as no abstracted sgdc has been given for these instances
```

Severity

Warning

SGDC_abstract_file01

Rule informs that in SoC flow, abstracted models provided for a product might be incorrect.

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_file01* rule reports a violation if an out-of-date abstract view of a product is being used in the SoC flow.

In such cases, regenerate the abstract view with the latest product version.

Message Details

This rule reports the following message if the generated abstract view of a product is incorrect:

```
SoC results for product <product-name> might be incorrect as
abstracted model for block '<block-name>' is not up-to-date.
Please re-generate their abstract model using latest
<product-name> policy
```

Severity

Warning

SGDC_abstract_file02

This rule reports that the provided abstracted models are generated with newer policy, and are not-compatible with current policy version.

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_file02* rule reports a violation if the abstract view is used in an old product version.

Consider an example in which you generate an abstract view of the SpyGlass CDC product in the SpyGlass 5.1 version. However, you are using this abstract view in the SpyGlass 4.7.1 version. In such cases, this rule reports a violation.

To fix this violation, perform any of the following actions:

- Regenerate the abstract view in the SpyGlass version being used.
- Use the abstract view in the latest SpyGlass version.

Message Details

This rule reports the following message if the abstract view is used in an old product version:

```
Abstracted models provided for product <product-name> are generated with newer <product-name> policy, and are not compatible with the current <product-name> product used. Please re-generate these abstract models with current <product-name> policy
```

Severity

Error

SGDC_abstract_file03

This rule reports that block SGDC file used during abstract model generation does not exist or not readable.

Language

Verilog, VHDL

Rule Description

The *SGDC_abstract_file03* rule reports a violation if the block SGDC file used during abstract view generation does not exist or is not readable.

Message Details

This rule reports the following message if the block SGDC file used during abstract view generation does not exist or is not readable:

```
Block SGDC file '<block-SGDC-file>' specified in abstract model
SGDC file '<abstract-SGDC-file>' does not exist or not readable
```

Severity

Fatal

DumpBlockInterface

Reports the interface information of the abstracted block in the generated abstract model

Language

Verilog, VHDL

Rule Description

The *DumpBlockInterface* rule reports the interface information of the abstracted block in the generated abstract model.

This rule runs if the `include_block_interface` and `block_abstract` options are specified.

Severity

Info

List of Topics