SpyGlass® Lint Turbo Structural User Guide

Version N-2017.12-SP2, June 2018

SYNOPSYS®

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

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

SpyGlass Lint Turbo Structural

The SpyGlass Lint Turbo Structural solution enables you to consolidate and better manage the violations reported by the SpyGlass Lint rules.

Prerequisite:

Make sure you have the `turbo_struct` license, to run this solution.

This user guide covers the following topics:

Using the Turbo Flow

To use the **turbo** mode, set the following parameter:

```
set_option turbo yes
```

When the **turbo** option is set, the violations are classified into Primary and Secondary violations. If you click on a Primary violation, a spreadsheet is opened. This spreadsheet contains secondary violations for each primary violation. You can apply the desired waivers for the violations present in the spreadsheet. You can also use the old waivers. Waivers created in turbo mode are waived in non-turbo mode.

Turbo Initiatives

The following table provide details about various initiatives/enhancements introduced in the rules that lead to consolidation and better management of the violations by classifying them into primary and secondary violations.

Initiative	Performs
TURBO_BUS_MERGE	Enables bus merging.
TURBO_CONSOLIDATE_DESIGN_SCOPE	Consolidates and reports violations, in a spreadsheet, based on block (always/ process)/task/ function/ assignment/ signed-variable/expr/ package/macro/ typdef/`define/per case statement/port/signal/ variable declaration.
TURBO_CONSOLIDATE_IO_PAIR	Consolidates and reports violations, in a spreadsheet, for connections between the same set of input and output ports of a module.
TURBO_CONSOLIDATE_MODULE	Consolidates and reports violations, in a spreadsheet, based on module name.
TURBO_CONSOLIDATE_MODULE_CONFIGURATION	Consolidates and reports violations, in a spreadsheet, from different configurations of a module.
TURBO_CONSOLIDATE_PER_SOURCE	Consolidates and reports violations, in a spreadsheet, based on the source (clock, reset, and signal name, etc.).
TURBO_CONSOLIDATE_PORT	Consolidates and reports violations, in a spreadsheet, based on the module instance offending port.
TURBO_DISABLE_ON_RTL	Disables a rule on RTL stage of design.

Initiative	Performs
TURBO_GROUP_MESSAGE	This is a global initiative that modifies messages and indicates the count of violating signals. All violations are reported in a spreadsheet.
TURBO_IGNORE_INTERNAL_OBJECTS	Disables reporting violations if any cells/nets/instances (combo/ multiplexes/tristate) are generated internally.
TURBO_IGNORE_PADDING	Disables reporting violations for extensions of bits or constant integers (like 32'h45).
TURBO_IGNORE_PARTIAL_BUS	Disables reporting violations if any bit of vector/multi-bit is set/read/ used.
TURBO_IGNORE_REDUNDANT_RULE	Enables the smart rule execution. See the *Smart Rule Execution* section for more details.
TURBO_IGNORE_STATIC_CONSTANTS	Disables reporting violations on expressions which contain a static/ constant operand, for example, based numbers, parameters, unsized based numbers, and initialization.
TURBO_REMOVE_DUP_MSG	Removes the duplicate messages.
TURBO_SET_RECOMMENDED_PARAMETER	Sets a parameter to the best fit value. See the *Turbo Parameter Settings* section for details.

Rule-wise Initiatives

The following tables present the list of rules in each product where message-consolidation initiatives are available:

- *erc*
- *latch*
- *lint*
- *miscellaneous*
- *morelint*
- *openmore*
- *simulation*
- *starc*
- *starc2002*
- *starc2005*
- *timing*

erc

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
checkMultipleDrivers	Verilog + VHDL	Only tristate pins may be WORed (multiple drivers only allowed, if all driving pins are of type tristate)	Not applicable	TURBO_IGNORE_REDUNDANT_RULE
checkPinConnectedToSupply	Verilog + VHDL	IO-ports or Output ports of cells/ modules may not be connected to supply signals	Pin of <count> cell <cell-name> (instance <instance-name>) of module <module-name> connected to supply signals. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
DisabledAnd	Verilog + VHDL	And/Nand gate is disabled	Input pin of <count> (comb-and/comb-nand) gates (instance <instance-name>) of module <module-name> tiedlow.	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_PER_SOURCE (applicable to spreadsheet only)

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
DisabledOr	Verilog + VHDL	Or/Nor gate is disabled	Input pin of <count> (comb-or/comb-nor) gates (instance <instance-name>) of module <module-name> tiedlow.	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_PER_SOURCE (applicable to spreadsheet only)
FlopClockConstant	Verilog + VHDL	Flip-flop clock pin driven by a constant value	Clock pin of '10' flop (instance '<inst-name'> of module '<module-name>' tie to constant	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_PER_SOURCE (applicable to spreadsheet only)
FlopDataConstant	Verilog + VHDL	Flip-flop data pin driven by a constant value	Data pin of <count> Flops (instance <instance-name>) of module <module-name> tie to constant.	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_PER_SOURCE (applicable to spreadsheet only)

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
FlopEConst	Verilog + VHDL	Flip-flop enable pin is permanently disabled or enabled	Enable pin of '<count>' flop (instance '<inst-name'> of module '<module-name>' tie to constant	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_PER_SOURCE (applicable to spreadsheet only)
FlopSRConst	Verilog + VHDL	Flip-flop set or reset pin is permanently enabled	Set or reset pin of <count> Flops (instance <instance-name>) of module <module-name> tie to constant	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_PER_SOURCE (applicable to spreadsheet only)
LatchDataConstant	Verilog + VHDL	Latch data pin driven by a constant value	Data pin of <count> Latches (instance <instance-name>) of module <module-name> tie to constant	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_PER_SOURCE (applicable to spreadsheet only)

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
LatchEnableConstant	Verilog + VHDL	Latch enable pin driven by a constant value	Enable pin of <count> Latches (instance <instance-name>) of module <module-name> tie to constant	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_PER_SOURCE (applicable to spreadsheet only)
MuxSelConst	Verilog + VHDL	Mux select is constant.	Select pin of <count> Mux (instance <instance-name>) of module <module-name> tie to constant	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_PER_SOURCE (applicable to spreadsheet only)
NoContAssign	Verilog	Continuous assignment statement present in technology-mapped netlist	Not applicable	TURBO_DISABLE_ON_RTL
TristateConst	Verilog + VHDL	Tristate gate enable is constant	Enable pin of <count> Tristate (instance <instance-name>) of module <module-name> tie to constant	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

latch

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W336L	Verilog	Blocking assignment to latch output should be avoided	Signal '<signal-name>' is assigned using blocking assignment. For '<count>' similar case(s), please refer to the spreadsheet for details	TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
LatchGatedClock	Verilog + VHDL	Do not use gated/ internally generated clock to drive latches	<count> latch(es) (output variable '<variable-name>') with gated or internally generated clock. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_BUS_MERGE

lint

3

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W111	Verilog	Not all elements of an array are read	Not all element of array '<array-name>' are read. For <count> similar occurrence(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_REMOVE_DUP_MSG
W111	VHDL	Not all elements of an array are read	Not all element of array '<array-name>' are read. For <count> similar occurrence(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_REMOVE_DUP_MSG

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W116	Verilog	Unequal length operands in bitwise logical/ arithmetic/ ternary operator	Not applicable	TURBO_REMOVE_DUP_MSG
W116	VHDL	Unequal length operands in bit wise logical/ arithmetic/ relational operator	Not applicable	TURBO_REMOVE_DUP_MSG
W120	Verilog	A signal/ variable has been declared but is not used	Variable '<variable-name>' declared but not used. For <count> similar occurrence(s), please refer to the spreadsheet for details.	TURBO_IGNORE_PARTIAL_BUS TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_REMOVE_DUP_MSG
W120	VHDL	A signal/ variable has been declared but is not used	Variable '<variable-name>' declared but not used. For <count> similar occurrence(s), please refer to the spreadsheet for details.	TURBO_IGNORE_PARTIAL_BUS TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_REMOVE_DUP_MSG

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W121	Verilog	A variable names collides with and may shadow another variable	Name '<variable-name>' is not unique at <count> places and may shadow another variable. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
W123	Verilog	A variable has been read but is not set	Not applicable	TURBO_REMOVE_DUP_MSG
W123	VHDL	A signal or variable has been read but is not set	Not applicable	TURBO_REMOVE_DUP_MSG
W154	Verilog	Do not declare nets implicitly	<count> more implicit declaration(s) of the net detected in the design. Please refer to the spreadsheet for details	TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
W159	Verilog	Condition contains a constant expression	Constant expression '<expression>' in condition. For <count> similar occurrence(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
W164a	Verilog	LHS width is less than RHS width of assignment (Truncation)	'<count>' occurrences of width mismatch on assignment '<assignment-node>'	TURBO_GROUP_MESSAGE
W164a	VHDL	LHS width is less than RHS width of assignment (Truncation)	Not applicable	TURBO_REMOVE_DUP_MSG

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W164b	Verilog	LHS width is greater than RHS width of assignment (Extension)	'<count>' occurrences of width mismatch on assignment '<assignment-node>'	TURBO_GROUP_MESSAGE
W164b	VHDL	LHS width is greater than RHS width of assignment (Extension)	Not applicable	TURBO_REMOVE_DUP_MSG
W164c	Verilog	LHS width is greater than RHS width of assignment (Extension)	LHS width '<widthl>' is greater than RHS width '<widthl>' [Hierarchy: '<hier-path>'], <count> occurrences of width mismatch on same assignment. Please refer to the spreadsheet for details.	TURBO_GROUP_MESSAGE
W171	Verilog	Case label is not constant	<count> non constant Case-labels in a case statement. Please refer to the spreadsheet for details.	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
W175	Verilog	A parameter/ generic has been defined but is not used	Module '<module-name>' has '<count>' unused Parameter. Please refer to the spreadsheet for details.	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W175	VHDL	A parameter/ generic has been defined but is not used	Entity '<entity-name>' has '<count>' unused generic. Please refer to the spreadsheet for details.	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
W190	Verilog	Task or procedure declared but not used	Not applicable	TURBO_CONSOLIDATE_DESIGN_SCOPE
W190	VHDL	Task or procedure declared but not used	Not applicable	TURBO_CONSOLIDATE_DESIGN_SCOPE
W240	Verilog	An input has been declared but is not read	Input '<input-port-name>' declared but not read. For <count> similar occurrence(s), please refer to the spreadsheet for details.	TURBO_IGNORE_PARTIAL_BUS TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_REMOVE_DUP_MSG
W240	VHDL	An input has been declared but is not read	Input '<input-port-name>' declared but not read. For <count> similar occurrence(s), please refer to the spreadsheet for details	TURBO_IGNORE_PARTIAL_BUS TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_REMOVE_DUP_MSG

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W263	Verilog	A case expression width does not match case select expression width	'<count>' case label width does not match case selector width of module '<module-name>' at line '<line-number>'	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_REMOVE_DUP_MSG
W280	Verilog	A delay has been specified in a non-blocking assignment	<block-type> has <count> non-blocking assignment(s) where intra-assignment delay is used. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE
W287b	Verilog	Output port of an instance is not connected	Instance output port '<output-port-name>' is not connected. For similar '<count>' violations of module '<module-name>', please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION
W287b	VHDL	Output port of an instance is not connected	Instance output '<instance-name>' not used. [Elaborated Module Name: <module-name>], for the similar '<count>' violation(s) of the elaborated module, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PORT TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_BUS_MERGE
W362	Verilog	Unequal length in arithmetic comparison operator	Not applicable	TURBO_REMOVE_DUP_MSG

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W401	Verilog + VHDL	Clock signal is not an input to the design unit	Clock '<clock-name>' is not an input to design unit '<design-unit-name>'. For similar '<count>' violation(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PER_SOURCE TURBO_GROUP_MESSAGE
W402b	Verilog	Asynchronous set/reset signal is not an input to the module	'signal-name' to flop '<flip-flop-name>' is gated or internally generated. For similar '<count>' violation(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PER_SOURCE TURBO_GROUP_MESSAGE
W415a	Verilog	Signal may be multiply assigned (beside initialization) in the same scope.	Signal <signal-name> is being assigned <count> times inside module '<module-name>'. Please refer to the spreadsheet for details	TURBO_SET_RECOMMENDED_PARAMETER TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_PER_SOURCE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
W443	Verilog	'X' value used	'<count>' occurrence of 'X' state used in '<block-name>' block of module '<module-name>' at line '<line-number>'	TURBO_IGNORE_STATIC_CONSTANTS TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
W443	VHDL	'X' value used	'<count>' occurrence of 'X' state used in '<block-name>' block of module '<module-name>' at line '<line-number>'	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W456a	Verilog	A signal is included in the sensitivity list of a combinational always block but none of its bits is read in that block	'<count>' signals/ variables are not required in sensitivity list of module '<module-name'> at line <'line-number'>	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_REMOVE_DUP_MSG
W456a	VHDL	A signal is included in the sensitivity list of a combinational process block but none of its bits is read in that block	Not applicable	TURBO_REMOVE_DUP_MSG
W464	Verilog	Unrecognized synthesis directive used in the design	<count> more unrecognized synthesis directive(s) used in the design. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
W464	VHDL	Unrecognized synthesis directive used in the design	<count> more unrecognized synthesis directive(s) used in the design. Please refer to the spreadsheet for details	TURBO_SET_RECOMMENDED_PARAMETER TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
W481a	Verilog	Possibly unsynthesizable loop: step variable differs from variable used in condition	Not applicable	TURBO_REMOVE_DUP_MSG

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W484	Verilog	Possible loss of carry or borrow due to addition/ subtraction	Possible assignment overflow: lhs width <widthl> (Expr: '<lexpr>') should be greater than rhs width <widthr> (Expr: '<rexpr>') to accommodate carry/ borrow bit, [Hierarchy: '<hierpath>'], <count> occurrences of assignment overflow on same assignment. Please refer to the spreadsheet for details	TURBO_GROUP_MESSAGE
W528	Verilog	A signal or variable is set but never read	Not applicable	TURBO_IGNORE_PARTIAL_BUS TURBO_REMOVE_DUP_MSG
W528	VHDL	A signal or variable is set but never read	Not applicable	TURBO_IGNORE_PARTIAL_BUS TURBO_REMOVE_DUP_MSG
W553	Verilog	Different bits of a bus are driven in different combinational blocks	net/bus '<signal-name>' is driven '<count>' times inside more than one combinational block. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W563	Verilog	Reduction of a single-bit expression is redundant	Unary reduction operator used <count> times on single bit usage of variable '<variable-name>'. Please refer to the spreadsheet for details	TURBO_GROUP_MESSAGE TURBO_REMOVE_DUP_MSG
W71	Verilog	A case statement (or selected signal assignment) does not have a default or OTHERS clause	Not applicable	TURBO_REMOVE_DUP_MSG
W287a	Verilog	Some inputs to instance are not driven or unconnected	Input '<input-name>' of instance '<instance-name>' is unconnected or Undriven.[Hierarchy: '<hier-path>']. For similar <count> violation(s) of module <module-name>, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
W287a	VHDL	Some inputs to instance are not driven or unconnected	Input Signal '<input-name>' of instance '<instance-name>' not driven. [Elaborated Module Name: <hier-name>], for the similar '<count>' violation(s) of the corresponding master, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PORT TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_BUS_MERGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
W191	Verilog	Function declared but not used	NA	TURBO_REMOVE_DUP_MSG
W191	VHDL	Function declared but not used	NA	TURBO_REMOVE_DUP_MSG
W490	Verilog	A control expression/ sub-expression is a constant	NA	TURBO_IGNORE_STATIC_CONSTANTS
W552	Verilog	Different bits of a bus are driven in different sequential blocks	Bus '<signal-name>' is driven '<count>' times inside more than one sequential block. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
W494a	VHDL	Input port is not used	<count> input port(s) of same entity not used. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE

miscellaneous

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
ConstSig	Verilog+ VHDL	Signal has a constant value or can only switch to a constant value	Signal '<signal-name>' has a constant value or can only switch to a constant value. For '<count>' similar occurrence(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
DeadCode	Verilog+ VHDL	Code does not contribute to functionality of the design	1. The net '<net-name>' does not drive anything, for the similar <count> occurrence(s), please refer to the spreadsheet for details 2. The net '<net-name>' is not driven by anything, for the similar <count> occurrence(s), please refer to the spreadsheet for details 3. Change on net '<net-name>' has no effect on any of the outputs, for the similar <count> occurrence(s), please refer to the spreadsheet for details 4. None of the inputs have any effect on net '<net-name>', for the similar <count> occurrence(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_BUS_MERGE
Mux01	Verilog+ VHDL	Connected muxes should not have a common select.	Some inputs of mux '<mux1-name>' do not propagate to connected mux '<mux2-name>', due to common select. For '<count>' similar occurrence(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PER_SOURCE TURBO_IGNORE_INTERNAL_OBJECTS TURBO_REMOVE_DUP_MSG

morelint

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
AsgnToOneBit-ML	Verilog	Assigning a 0 or 1(32-bits) to a 1 bit is not allowed	1-bit is assigned a 0 or 1 (32 bits) in module '<module-name>' at '<count>' place(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE
BitOrder-ML	Verilog	Bit order specification should follow recommended convention	Bit order specification(s) at <count> more places do not follow the recommended convention. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
ChkUndefMacro-ML	Verilog	Macro is not defined before undefining it	<count> occurrence(s) of improper usage of 'undef' for macro detected in the file '<file-name>'. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE
ConstDrivenNet-ML	Verilog	All internal nets of module should not be assigned a constant value	Please refer to the 'ConstDrivenNet-ML_Ve_001.csv' for rest <count> violations of Verilog cases	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
ConstDrivenNet-ML	VHDL	All internal nets of module should not be assigned a constant value	Please refer to the 'ConstDrivenNet-ML_Vh_001.csv' for rest <count> violations of Verilog cases	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
DisallowCaseZ-ML	Verilog	Design should not use casez constructs	Not applicable	TURBO_CONSOLIDATE_MODULE_CONFIGURATION
HangingInstOutput-ML	Verilog + VHDL	Net connected to output port of instance is unconnected	Net '<net-name>' connected to output port of instance is unconnected. For similar <count> occurrence(s) for same module '<module-name>', please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_MODULE
NoArithOp-ML	Verilog	Arithmetic Operators should be avoided.	Expression '<expr-name>' containing '<operator>' operator. For similar <count> occurrence(s) for the same '<scope-name>' scope, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
NoArithOp-ML	VHDL	Arithmetic Operators should be avoided.	Expression '<expr-name>' containing '<operator>' operator. For similar <count> occurrence(s) for the same '<scope-name>' scope, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
NoAssignX-ML	Verilog	RHS of the assignment contains 'X	'RHS of the assignment contains 'X' <reason> at '<count>' place(s) in always block. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE
NoBusPartClock-ML	Verilog + VHDL	Bus bits or slices should not be used as clocks	'<count>' times Bus bits or slices used as clocks. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PER_SOURCE TURBO_BUS_MERGE
NoExprInPort-ML	VHDL	Port connections in instances should not contain expressions	Instantiation of module '<modulename>' has '<count>' similar occurrence(s).Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
NoFeedThrus-ML	Verilog + VHDL	Block should not contain feed-throughs	There is feed-through from input '<input-name>' to output '<output-name>', for <count> occurrence(s) of the similar feed-through paths, please refer to the spreadsheet for details	TURBO_BUS_MERGE TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_CONSOLIDATE_IO_PAIR
NoParamMultConcat-ML	Verilog	Do not use parameter as left operand in multiple concatenation expression	Parameter '<parameter-name>' used '<count>' times in the multiple concatenation expression. Please refer to the spreadsheet for the details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
NoSigCaseX-ML	Verilog	Design should not use signals in casex and casez constructs	'<count>' signals used in '<casex | casesz> (<select-line-signal)' item at line '<line-number>'	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
NoXInCase-ML	VHDL	Case expression and case choices should not have 'X'	'<count>' case expressions of case construct at line '<line-number>' of arch '<architecture-name>' contains X	TURBO_CONSOLIDATE_MODULE_CONFIGURATION

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
ParamOverrideMismatch-ML	Verilog	Mismatch in the number of parameter over-rides and number of parameters in the instantiated module	Module '<module-name>' has '<count>' instantiation for mismatch in number of parameter over-rides and number of parameters in the instantiated module. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
PartConnPort-ML	Verilog	Port is unconnected or partially connected	Port '<port-name>' is unconnected or partially connected in instance '<instance-name>'[Hierarchy: '<hier-path>']. Module '<module-name>' has '<count>' similar occurrence(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
ReEntrantOutput-ML	Verilog + VHDL	The re-entrant outputs should be avoided	Output port '<output-port-name>' is driving the input port '<input-port-name>' of the same instance. For '<count>' similar occurrence(s), please refer to the spreadsheet for details	TURBO_BUS_MERGE TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
ResetFlop-ML	Verilog + VHDL	All the flip-flops should have either asynchronous set/reset or synchronouss et/reset	Flip-flop '<flip-flop-name>' has no set or reset.[Hierarchy: '<hier-path>'], for the similar <count> occurrence(s) of flip-flops, please refer to the spreadsheet for details	TURBO_REMOVE_DUP_MSG TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE TURBO_BUS_MERGE
SignedUnsignedExpr-ML	Verilog	Do not mix signed & unsigned variables/ constants in expressions, assignment statements or in comparisons.	Unsigned expression used with Signed expression <count> times within same statement. Please refer to the spreadsheet for details	TURBO_IGNORE_STATIC_CONSTANTS TURBO_GROUP_MESSAGE TURBO_SET_RECOMMENDED_PARAMETER
UndrivenInTerm-ML	Verilog + VHDL	Undriven but loaded input terminal of an instance detected	'<count>' input terminal(s) of module '<module-name>' are undriven	TURBO_CONSOLIDATE_PORT TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
UndrivenNUnloaded-ML	Verilog + VHDL	Undriven and Unloaded nets/ terminals detected in the design	Detected undriven and unloaded(unconnected) <terminal-name>. For similar <count> occurrence(s) of same module '<module-name>', please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PORT TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
UndrivenOutPort-ML	Verilog + VHDL	Undriven but loaded output port of a module detected	'<count>' output port of module '<module-name>' are undriven	TURBO_CONSOLIDATE_PORT TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
UndrivenOutTermNLoaded-ML	Verilog + VHDL	Undriven output pins connected to instance input	'<count>' input terminals of module '<module-name>' are connected to undriven output terminal	TURBO_CONSOLIDATE_PORT TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
UnloadedInPort-ML	Verilog + VHDL	Unloaded but driven input port of a module detected	Detected '<count>' unloaded(unconnected) (input port type <port-type>) of module '<module-name>'	TURBO_CONSOLIDATE_PORT TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_MODULE_CONFIGURATION

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
UnloadedNet-ML	Verilog + VHDL	Unloaded but driven net detected in the design	Detected '<count>' unloaded(unconnected) (net type <net-type>) of module '<module-name>'	TURBO_BUS_MERGE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
UnloadedOutTerm-ML	Verilog + VHDL	Unloaded but driven output terminal of an instance detected	Detected '<count>' unloaded(unconnected) (output terminal type <terminal-name><connected to a floating net> (<net-name>)) of module '<module-name>'	TURBO_CONSOLIDATE_PORT TURBO_GROUP_MESSAGE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
NoWidthInBasedNum-ML	Verilog	Width should be specified for all based numbers	Width specification missing for <count> based number(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE TURBO_IGNORE_PADDING
ParamWidthMismatch-ML	Verilog	Parameter width does not match with the value assigned	or <count> occurrence(s) of parameter width mismatch in module <module-name>, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
NestedCaseStmt-ML	Verilog	Nesting depth of case construct exceeds specified depth	Nested case construct found. Depth exceeding specified limit $casedepth at '<count>' place(s) in source '<source-file-name>'. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
NestedCaseStmt-ML	VHDL	Nesting depth of case construct exceeds specified depth	Nested case construct found. Depth exceeding specified limit $casedepth at '<count>' place(s) in source '<source-file-name>'. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
UnrecSynthDir-ML	Verilog	Synthesis directive is not recognized	<count> more unrecognized synthesis directive(s) for pragmas used in the design. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
UnrecSynthDir-ML	VHDL	Synthesis directive is not recognized	<count> more unrecognized synthesis directive(s) for pragmas used in the design. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
RedundantLogicalOp-ML	Verilog	Logical operation result is same as one of the operands or is a constant	Redundant logical operation is performed at <count> more places. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
RedundantLogicalOp-ML	VHDL	Logical operation result is same as one of the operands or is a constant	Redundant logical operation is being performed on a constant at <count> more places. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
SynchReset-ML	Verilog	Do not use synchronous reset in the design	Synchronous reset used at '<count>' place(s) inside module '<module-name>'. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
SynchReset-ML	Verilog + VHDL	Do not use synchronous reset in the design	Synchronous reset used at '<count>' place(s) inside module '<module-name>'. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
HangingInstInput-ML	Verilog + VHDL	Net connected to input port of instance is undriven	'<count>' input port of master '<master-mod-name>' are '<undriven>'	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
UndrivenNet-ML	Verilog + VHDL	Undriven but loaded net is detected in the design	Detected undriven net <net-name>. For similar <count> occurrence(s) of same module '<module-name>', please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE

openmore

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
ArrayIndex	Verilog	Bus signals are declared with low-order bit first	<count> arrays not declared in recommended format of [MSB:LSB]. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
ArrayIndex	VHDL	Bus signals are declared with low-order bit first	Not applicable	TURBO_REMOVE_DUP_MSG
InferLatch	Verilog + VHDL	Latch inferred	'<count>' latche(s) inferred for signal '<signal-name> in module '<module-name>'	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_BUS_MERGE TURBO_SET_RECOMMENDED_PARAMETER
IntClock	Verilog + VHDL	Internally generated clock detected	Internally generated Clock '<clock-name>'(flop: <flip-flop-name>) detected, for the similar <count> occurrence(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_BUS_MERGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
IntReset	Verilog + VHDL	Internally generated reset detected	<count> occurrence(s) of Internally generated asynchronous set/ reset '<signal-name>'. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_BUS_MERGE
ConsCase	Verilog	Name not used consistently with same case	Module <module-name> has <count> names which are not used with consistent case. Please refer to the spreadsheet for details.	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
ConsCase	VHDL	Name not used consistently with same case	Name '<identifier-name>' not used with consistent case. For <count> similar occurrence(s) , please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
ActLowName	Verilog	Active-low signal name does not follow the naming convention	Active Low signal '<signal-name>' does not follow naming convention. For <count> similar occurrence(s) of module <module-name>, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE
ActLowName	VHDL	Active-low signal name does not follow the naming convention	Active Low signal '<signal-name>' does not follow naming convention. For <count> similar occurrence(s) of module <module-name>, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE

NOTE: *In the turbo mode, irrespective of the parameter setting, the InferLatch rule will not report violation for those latches whose input 'D' is tied to constant.*

simulation

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
sim_race01	Verilog	Assignment and use of signal in same simulation cycle (Read-Write Race)	'<count>' occurrences of Read-Write Race for signal '<signal-name>' w.r.t. always block at line '<line-number>'. Please refer to the spreadsheet for details	TURBO_IGNORE_PARTIAL_BUS TURBO_CONSOLIDATE_DESIGN_SCOPE

starc

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
STARC-1.4.3.4	Verilog + VHDL	Flip-flop clock signals must not be used as non-clock signals	Clock signal '<signal-name>' used as a non-clock (Used with name '<non-clock-name>'). For similar <count> cases, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PER_SOURCE
STARC-1.6.6.3	Verilog	Do not instantiate library cells in the design	Library cell '<cell-name>' instantiated. For similar '<count>' violations for module '<module-name>', please refer to the spreadsheet for details	TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
STARC-1.6.6.3	VHDL	Do not directly instantiate cells in the design	Library cell '<cell-name>' instantiated. For similar '<count>' violations for module '<module-name>, please refer to the spreadsheet for details	TURBO_GROUP_MESSAGE
STARC-2.1.6.1	Verilog	Array specification should follow recommended convention.	<count> arrays not declared in recommended format of [MSB:LSB]. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE TURBO_REMOVE_DUP_MSG
STARC-2.1.6.1	VHDL	Specification of one-dimensional array should be downto	<count> arrays not declared in recommended format of [MSB:LSB]. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
STARC-2.10.3.5	Verilog	Base should be specified for constant value used.	Missing base type specification in constant value at <count> place(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_CONSOLIDATE_MODULE_CONFIGURATION
STARC-2.10.6.1	Verilog	Possible loss of carry or borrow in addition/ subtraction	<count> occurrences of assignment overflow on same assignment. Please refer to the spreadsheet for details	TURBO_GROUP_MESSAGE TURBO_IGNORE_REDUNDANT_RULE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
STARC-2.2.2.2	Verilog	A variable specified in the sensitivity list must be read in the contained block	<variable-name> signal/variable(s) not read in <block-type> is/are not required in sensitivity list. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_IGNORE_STATIC_CONSTANTS TURBO_REMOVE_DUP_MSG
STARC-2.2.2.2	VHDL	A signal should not be included in sensitivity list of a process if it is not read in that process	Not applicable	TURBO_REMOVE_DUP_MSG
STARC-3.2.2.7	Verilog	Constants should be defined using parameters only.	Use parameter instead of macro at <count> place(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
STARC-3.3.2.2a	Verilog + VHDL	Internal clocks must be controllable from external pins	Internal clock '<clock-name>' (flop: '<flip-flop-name>') not controllable from external pins, for the similar <count> occurrence(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_BUS_MERGE
STARC-1.3.1.3	Verilog + VHDL	Asynchronous reset/preset signals must not be used as non-reset/ preset or synchronous reset/preset signals	Asynchronous <type> signal '<signal-name>' (<flop | latch>: '<flip-flop-name | latch-name>') used as non-<type>/ synchronous-<type> at instance '<instance-name>' (File Name: '<file-name>' ,Line no.: '<line-number>'). For similar '<count>' case(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PER_SOURCE

starc2002

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
STARC02-1.4.3.4	Verilog + VHDL	Connected net name of a module instance must be same as or based on the master module output port name.	Clock signal '<signal-name>1' used as a non-clock (Used with name '<non-clock-name>'). For similar <count> cases, please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PER_SOURCE
STARC02-2.1.6.1	Verilog	Array specification should follow recommended convention.	<count> arrays not declared in recommended format of [MSB:LSB]. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE TURBO_REMOVE_DUP_MSG
STARC02-2.1.6.1	VHDL	Specification of one-dimensional array should be downto	<count> arrays not declared in recommended format of [MSB:LSB]. Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
STARC02-2.10.3.5	Verilog	A function description must assign return values to all possible states of the function.	Missing base type specification in constant value at <count> place(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE
STARC02-2.10.3.7	Verilog	Match the bit width with the base number part	Mismatch in actual width and specified width for <count> based number(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_IGNORE_PADDING

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
STARC02-2.10.6.1	Verilog	Possible loss of carry or borrow in addition/ subtraction	Possible assignment overflow: lhs width <widthl> (Expr: '<lexpr>') should be greater than rhs width <widthr> (Expr: '<rexpr>') to accommodate carry/ borrow bit, [Hierarchy: '<hier-path>'], <count> occurrences of assignment overflow on same assignment. Please refer to the spreadsheet for details.	TURBO_GROUP_MESSAGE TURBO_IGNORE_REDUNDANT_RULE
STARC02-3.2.2.7	Verilog	Constants should be defined using parameters only.	Use parameter instead of macro at <count> place(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
STARC02-1.3.1.3	Verilog + VHDL	Do not use case variants of names in the same scope.	Asynchronous <type> signal '<signal-name>' (<flop | latch>: '<flip-flop-name | latch-name>') used as non-<type>/synchronous-<type> at instance '<instance-name>' (File Name: '<file-name>' ,Line no.: '<line-number>'). For similar '<count>' case(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PER_SOURCE

starc2005

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
STARC05-1.4.3.4	Verilog + VHDL	Flip-flop clock signals must not be used as non-clock signals	'<count>' Non-clock signal '<non-clock-signal-name>' infer from same RTL used as clock signal	TURBO_CONSOLIDATE_PER_SOURCE
STARC05-2.1.3.1	Verilog	Bit-width of function arguments must match bit-width of the corresponding function inputs	<count> occurrences of width mismatch on same function call argument '<argument-name>'. Please refer to the spreadsheet for details	TURBO_GROUP_MESSAGE TURBO TURBO_IGNORE_PADDING
STARC05-2.10.3.5	Verilog	Base should be specified for constant value used	Missing base type specification in constant value at <count> place(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE
STARC05-2.10.3.7	Verilog	Match the bit width with the base number part	Mismatch in actual width and specified width for <count> based number(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_IGNORE_PADDING

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
STARC05-2.10.6.1	Verilog	Possible loss of carry or borrow in addition/ subtraction	Possible assignment overflow: lhs width <widthl> (Expr: '<lexpr>') should be greater than rhs width <widthr> (Expr: '<rexpr>') to accommodate carry/ borrow bit, [Hierarchy: '<hier-path>'], <count> occurrences of assignment overflow on same assignment. Please refer to the spreadsheet for details	TURBO_GROUP_MESSAGE TURBO_IGNORE_REDUNDANT_RULE
STARC05-3.2.2.7	Verilog	Constants should be defined using parameters only	Use parameter instead of macro at <count> place(s). Please refer to the spreadsheet for details	TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_GROUP_MESSAGE
STARC05-3.3.1.1	Verilog + VHDL	Internal clocks must be controllable from external pins	Internal clock '<clock-name>' (flop: '<flip-flop-name>') not controllable from external pins, for the similar <count> occurrence(s), please refer to the	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_CONSOLIDATE_DESIGN_SCOPE TURBO_BUS_MERGE

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
STARC05-3.3.1.4b	Verilog + VHDL	A flip-flop should have an asynchronous set or an asynchronous reset	Flip-flop '<flip-flop-name>' has neither asynchronous set nor asynchronous reset. [Hierarchy: '<hier-path>'], for the similar '<count>' occurrence(s),	TURBO_CONSOLIDATE_MODULE_CONFIGURATION TURBO_GROUP_MESSAGE TURBO_BUS_MERGE
STARC05-1.3.1.3	Verilog + VHDL	Asynchronous reset/preset signals must not be used as non-reset/ preset or synchronous reset/preset signals	Asynchronous <type> signal '<signal-name>' (<flop | latch>: '<flip-flop-name | latch-name>') used as non-<type>/ synchronous-<type> at instance '<instance-name>' (File Name: '<file-name>' ,Line no.: '<line-number>'). For similar '<count>' case(s), please refer to the spreadsheet for details	TURBO_CONSOLIDATE_PER_SOURCE

timing

Rule	Lang.	Rule Title	Parent Message	Turbo Initiatives
LogicDepth	Verilog + VHDL	Logic depth exceeds specified number of levels (using parameter delaymax/ delaymax_memtoflop / delaymax_floptomem / delaymax_memtomem/delaymax_err/ delaymax_inputtoflop / delaymax_floptoflop/ delaymax_inputtooutput/ delaymax_floptooutput)	Not applicable	TURBO_SET_RECOMMENDED_PARAMETER
LogNMux	Verilog + VHDL	LogN mux with large number of inputs detected - potential performance problem	Not applicable	TURBO_SET_RECOMMENDED_PARAMETER

Turbo Parameter Settings

The default value of the following rule parameters is changed when the **turbo** mode is enabled:

Parameter	Rules
checkfullrecord	W456a, W528, W123, W120, W240
checkfullbus	W240, W528, W120, UnUsedFunctionInput-ML, UndrivenInTerm-ML
min_based_width is set to 1	NoWidthInBasedNum-ML
delaymax is set to 500	LogicDepth
logmux_max is set to 5	LogNMux
check_initialization_assignment, ignore_multi_assign_in_forloop, ignore_bitwiseor_assignment	W415a
ignoreforindex	SignedUnsignedExpr-ML
checkInHierarchy	HangingInstInput-ML, UndrivenOutPort-ML, UndrivenInTerm-ML, UndrivenNet-ML, UnloadedInPort-ML, UnloadedOutTerm-ML, UnloadedNet-ML, UndrivenNUnloaded-ML, UndrivenOutTermNLoaded-ML
checkRTLCInst	UndrivenInTerm-ML
checkalldimension	ArrayIndex
ignoreModuleInstance	W123
ignore_chain_cell	DisabledAnd, DisabledOr
nocheckoverflow	STARC05-2.1.3.1,W116,W164a,W164b,W263,W110,W486,W362
strict	W342, W343

However, to retain the default value of the above parameters in the turbo mode, specify the following command:

```
set_option disable_turbo_param yes
```

By default, the strict parameter is always enabled for the *InferLatch* rule. In order to disable it, use the `disable_turbo_param` option.

NOTE: *In the **turbo** mode, the default value of the parameters will be synchronized with GuideWare 2015.12 setting.*

base_turbo_limit_viol

This parameter specifies the limit of violation messages to be reported in the message tree, while remaining messages are reported in the spreadsheet.

By default, this parameter is set to `2`.

Set the `base_turbo_limit_viol` parameter to a positive integer value to limit the number of violation messages to be reported in message tree. You can also specify different values for different rules using the comma-separated (no spaces) `<rulename>=<value>` pairs. See the examples below for more clarity.

NOTE: *There should be no spaces between the rule name and its limit value before and after the **=** symbol, that is, <rule-name>=<value>.*

NOTE: *If you specify a value that is less than 1, the rule considers the default value only, which is 2.*

Examples:

- `set_parameter base_turbo_limit_viol 10,SynchReset-ML=15,ConsCase=4`

  In this case, for the rules other than *SynchReset-ML* and *ConsCase* value of this parameter will be set to `10`, while for the *SynchReset-ML* and *ConsCase* rules, it will be set to `15` and `4` respectively.

- `set_parameter base_turbo_limit_viol SynchReset-ML=15,ConsCase=4`

  In this case, for the rules other than *SynchReset-ML* and *ConsCase*, the value of this parameter will be set to `2` (default value).

- If the `base_turbo_limit_viol` rules is not set, then the default value, which is `2`, is considered for all the rules.

NOTE: *The parameter is valid only when the* `'set_option turbo yes'` *is specified.*

The `base_turbo_limit_viol` parameter is supported by the following rules:

Rule name	Product	Language
ArrayIndex	openmore	Verilog
ConstDrivenNet-ML	morelint	Verilog
ConstDrivenNet-ML	morelint	VHDL
ConsCase	openmore	Verilog
NestedCaseStmt-ML	morelint	Verilog
ParamWidthMismatch-ML	morelint	Verilog
SynchReset-ML	morelint	Verilog
ActLowName	openmore	Verilog
ConsCase	openmore	VHDL
NestedCaseStmt-ML	morelint	VHDL
W494a	lint	VHDL
SynchReset-ML	morelint	VHDL
UnrecSynthDir-ML	morelint	VHDL

Smart Rule Execution

SpyGlass Lint Turbo Structural enables the smart execution of rules. When a superset rule is run, in that case subset rule will not run. So, you can use a superset rule to perform the tasks of specified subset rules.

The smart rule execution works in the **Turbo** mode only.

For the smart rule execution to perform, the following should be set:

- `set_option turbo yes`
- `set_option smart_rule_execution yes`

The following table lists the rules that are part of smart rule execution:

Superset Rule	Subset Rule(s)
W484	STARC-2.10.6.1, STARC05-2.10.6.1, STARC02-2.10.6.1
STARC05-2.10.6.1	STARC-2.10.6.1, STARC02-2.10.6.1
STARC02-2.10.6.1	STARC-2.10.6.1
W415	checkMultipleDrivers

NOTE: *Smart rule execution is also applicable for the Starc, Starc2002 and Starc2005 rules. The Starc2005 rules are superset of Starc2002 and Starc rules. The Starc2002 rules are superset of Starc rules.*

NOTE: *If the smart_rule_execution option is set to yes and the turbo_struct license is not found, then the smart rule execution feature is disabled.*

Waiving Primary Messages in the Turbo Mode

In the **turbo** mode, you can waive primary messages along with the respective secondary messages.

When you click on the waive option, a confirmation box is displayed with the following options, as shown in the figure:

- **Yes**: Primary messages are waived along with all the secondary messages.

- **No**: Primary messages are waived. However, if you select multiple messages (primary along with secondary messages of the same or different rules), only secondary messages are waived.
- **Cancel**: No violations are waived.

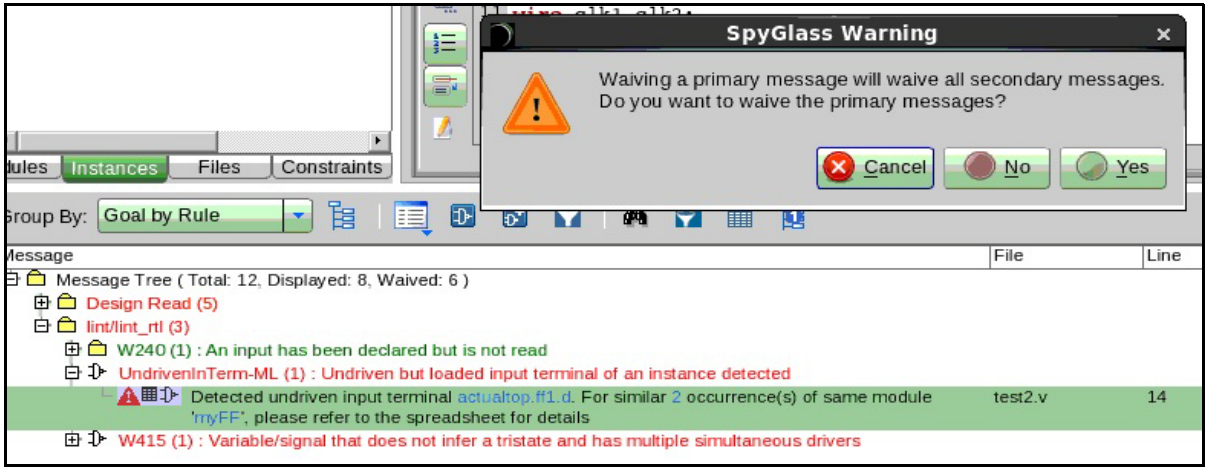


FIGURE 1. Primary message waiver confirmation

Defining the Preferred Waiver Behavior

You can define the preferred waiver behavior by using the **Waiving primary messages** set of options in the **Tools > Preferences > Waiver** menu of **SpyGlass Console**.

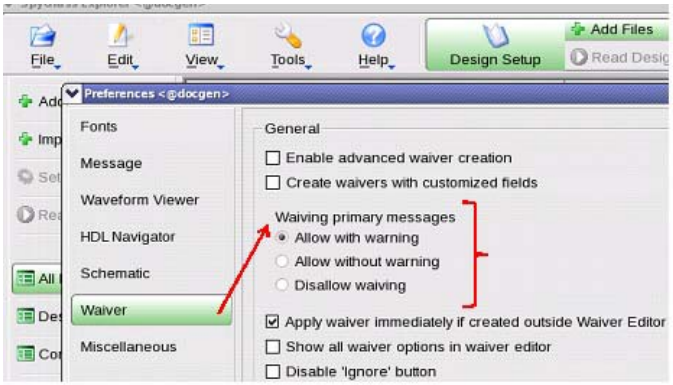


FIGURE 2. Defining the primary messages waiver behavior

The following options are available:

- **Allow with warning:** This option is selected by default. In case of any primary message in the selected message list, the warning/confirmation box is displayed.
- **Allow without warning:** Primary messages are waived without any warning and the warning/confirmation box is not displayed.
- **Disallow waiving**: Primary messages are not waived and the warning/confirmation box is also not displayed. This changes the right-click menu behavior as follows:
  - Right-click when both primary and non-primary violations selected:
    - Options for waiving messages are displayed in the right-click menu. However only non-primary messages are waived on selecting it.
  - Right-click when only the primary messages are selected:
    - Options for waiving messages are not displayed in the right-click menu.

The moresimple_turbo Report

In the **turbo** mode, the `moresimple_turbo.rpt` report is also generated by default.
This report contains all the violation messages from the message tree and spreadsheet.

The following is a sample of this report:

```
#########################################################################
#
# This file has been generated by SpyGlass:
#     Report Name      : moresimple_turbo
#     Report Created by: nadunig
#     Report Created on: Tue Mar 15 18:24:08 2016
#     Working Directory: /remote/atr_sl_user/nadunig/Test/560/98424/
case3
#     SpyGlass Version : 5.6.1-FCS-C2
#     Policy Name      : SpyGlass(5.6.1)
#                        erc(5.6.1)
#                        lint(5.6.1)
#                        morelint(5.6.1)
#
#     Total Number of Generated Primary Messages   :         10
#     Total Number of Generated Secondary Messages :          2
#     Number of Waived Primary Messages            :          0
#     Number of Waived Secondary Messages          :          0
#     Number of Reported Primary Messages          :         10
#     Number of Reported Secondary Messages        :          2
#     Number of Overlimit Messages                 :          0
#
```

```
#
################################################################################

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
MORESIMPLE_TURBO REPORT:

############### BuiltIn -> RuleGroup=Design Read ###############
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ID      ParentID   Rule                 Alias                Severity
File
Line    Wt
Message
================================================================================
[3]     N.A      DetectTopDesignUnits   DetectTopDesignUnits   Info
checkPinConnectedToSupply.v
10      2
Module top is a top level design unit
[0]     N.A      AutoGenerateSglib      AutoGenerateSglib      Info
Work/Linux4/test/spyglass_cache/autogenerated_sglib/lc/
spyglass_lc_aggregate_reports/moresimple.rpt    0       2
Sglib 'Work/Linux4/test/spyglass_cache/autogenerated_sglib/
aggregate.sglib' has been auto-generated successfully
[2]     N.A       ElabSummary           ElabSummary           Info
Work/Linux4/test/test_goal/spyglass_reports/SpyGlass/elab_summary.rpt
0       2
Please refer file 'Work/Linux4/test/test_goal/spyglass_reports/SpyGlass/
elab_summary.rpt' for elab summary report
[1]     N.A       TurboModeStatus       TurboModeStatus       Info
N.A.
0       10
Turbo-Mode is enabled in the current run as turbo_struct license feature
```

```
successfully checked out
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
############### Non-BuiltIn -> Goal=test_goal ###############
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ID    ParentID  Rule              Alias   Severity   File
Line    Wt    Message
==============================================================================

[4]      N.A        RegInputOutput-ML                    Warning
checkPinConnectedToSupply.v    5       10    Port 'a' is not registered
[Hierarchy: 'top']

[5]      N.A        RegInputOutput-ML                    Warning
checkPinConnectedToSupply.v    5       10    Port 'b' is not registered
[Hierarchy: 'top']

[6]      N.A        RegInputOutput-ML                    Warning
checkPinConnectedToSupply.v    6       10    Port 'c' is not registered
[Hierarchy: 'top']

[7]      N.A        RegInputOutput-ML                    Warning
checkPinConnectedToSupply.v    13      10    Port 'd' is not registered
[Hierarchy: 'top']

[8]      N.A        RegInputOutput-ML                    Warning
checkPinConnectedToSupply.v    14      10    Port 'e' is not registered
[Hierarchy: 'top']

[B]      N.A        checkPinConnectedToSupply            Warning
checkPinConnectedToSupply.v    5       2     Pin of 2 cell LAN2 (instance
top.U1.I1) of module mod connected to supply signals. Please refer to the
spreadsheet for details

[9]      [B]        checkPinConnectedToSupply            Warning
checkPinConnectedToSupply.v    5       2     pin Z for instance top.U2.I1
(Cell LAN2) connected to supply signals

[A]      [B]        checkPinConnectedToSupply            Warning
checkPinConnectedToSupply.v    5       2     pin Z for instance top.U1.I1
(Cell LAN2) connected to supply signals
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Known Problems

The following are the current known issues in the Turbo mode:

- Performance (run-time) issue in GUI while applying waivers in spreadsheet.
- Assume if there are two parents messages, P1 and P2, and each parent message is having two child messages, ‘C1’ and ‘C2’ corresponding to the parent message ‘P1’ and child messages ‘C3’ and ‘C4’ corresponding to the parent message ‘P2’ (child message is a static message, that is, no variable argument in it). In this case if a waiver is applied to the child message ‘C1’ of parent message ‘P1’ then waiver will be applied to all the child messages, but parent message ‘P1’ will be waived and there will be no impact on the parent message ‘P2,’ whereas the child message of parent message ‘P2’ also gets waived.
- When you open spreadsheet tab to apply waiver, after applying waivers if we press cancel button on waiver window, the waivers are still applied.
- For a message having same argument but different line/file, if you apply waiver, it is set on all messages. But if you unwaive one message then it will be applicable to only that one. So there is inconsistency.
- For rules which are spyCreateruleByCopy, the CSV file directory is created inside the parent rules->policy directory structure of spyglass_reports/<policy_name>.
- Parent violations have been made transient, and waivers are not saved for parent, for subsequent runs. This may lead to difference with respect to parent message if you load the GUI (after running the design in the batch run).
- Peak memory and the runtime may increase slightly. As multiple callbacks DS is getting invoked.
- If a parent message is not added into the vdb by the sgHandleMessage call (violStatus from sgHandleMessage is: SG_VIOL_STATUS_ERROR), then all of its child messages will be lost. Currently no non-error child message we are made as parent violations.
- As parent messages can be added as a child messages, assume a parent message has a flip-flop I1.q1[0] and it has four child messages, that is, I1.q1[0], I1.q1[1], I1.q1[2] and I1.q1[3]. If the child q1[0] is internally waived by the tool, then parent is reported as I1.q1[0] with three child messages (except the child I1.q1[0]). Ideally I1.q1[1] or I1.q1[2] or

I1.q1[3] should be reported as a parent argument, which mean the parent arguments are not changed, so the count in the parent message is being adjusted.

- Child count in the parent message can be one due to various reasons.
  - Few child messages are internally waived [other than user waivers]
  - Child messages are getting bus-merged
  - If child messages are part of `enable_dup_msg_removal`, then the parent with single child will be dumped
- For some violations of the *LatchGatedClock* rule, gated/internally generated clock source terminal and latch instance will only be highlighted instead of the complete path.
- If the message label changes, in that case there may be a case of break in waiver with respect to the old release.
- If parameterized module is instantiated in multiple places and all are having the same message then in few rules user may see one duplicate violation out of the spreadsheet in message tree. This will happen to a rule having a the `-enable_dup_msg_removal` support
- If one parent message is waived (through user waiver or msg-tree), then infrastructure should waive all its child messages.
- Internal waivers may get applied on parent message, so all its child messages will be lost.
- Waive –du field is not applied on spreadsheet violation, while its applicable for normal message tree violations.
- Overload severity through external command will not be applicable for new parent messages (as their label will be different).
- In the Turbo mode, bus-merging may break in a few cases. This is due to `enable_smdb_reports`, which populates the reports in the Turbo mode.

SpyGlass Functional Lint

Recent advancements in the industry have put emphasis to couple static lint checker with formal verification to augment current verification flows.

The existing SpyGlass Lint rules adopt structural analysis techniques. Some of these rules report many false violations because of the lack of design intent information.

SpyGlass supports Functional Lint rules that use formal verification strategies resulting in less noise and accurate results.

This section describes the following topics:

- *Prerequisites for Using Turbo Functional Rules*
- *Turbo Capabilities*
- *Functional Lint to Lint Rule Mapping*
- *Generating Waivers for Structural Rules*
- *The SpyGlass Functional Lint Rules*

For information on lint concepts and debug, refer to the *SpyGlass Lint and SpyGlass Auto Verify Rules Reference Guides.*

Prerequisites for Using Turbo Functional Rules

Specify the following:

- The `lint_func` licenses
- Clocks and resets in an SGDC file (recommended approach)

  If you do not specify clocks and resets in an SGDC file, set the `use_inferred_clocks` and `use_inferred_resets` parameter to `yes` to enable SpyGlass to automatically infer clocks and resets.
- Single top-level design
- If the `turbo_struct` license is successfully checked out, then the noise reduction for the structural rule will be ported to the formal rule.
- By default, waveform viewer is disabled. You can enable it using the *Edit -> Preferences* option in SpyGlass Explorer.

Turbo Capabilities

The SpyGlass Functional Lint product supports noise reduction capability, similar to the SpyGlass Turbo Lint (structural) product.

Functional Lint to Lint Rule Mapping

The following table lists the rules in the *SpyGlass Functional Lint* product and the corresponding rule in *SpyGlass Lint* product:

TABLE 1 Rule Mapping

Functional Lint Rule	Lint Rule
Av_width_mismatch_assign	W164a
Av_width_mismatch_case	W263
Av_width_mismatch_port	W110
Av_width_mismatch_function	STARC-2.1.3.1
Av_signed_unsigned_mismatch	SignedUnsignedExpr-ML
Av_width_mismatch_expr	STAR-2.10.3.2a
Av_width_mismatch_expr02	W116
Av_width_mismatch_expr03	W362
Av_case_default_redundant	W551
Av_case_default_missing	W71
Av_dontcare_mismatch	W467

Generating Waivers for Structural Rules

SpyGlass Functional Lint allows you to filter false violations generated by the SpyGlass Lint rules. SpyGlass Functional Lint generates an SGDC file, which when used with the subsequent SpyGlass Lint run waives the false violations filtered by Formal-Lint. This SGDC file is generated at the following location:

```
<run_directory>/spyglass_reports/auto-verify/
structural_rule_waiver.sgdc
```

Example

Consider the following example:

```
module top(input clk, rst);
parameter p = 1'b1;
wire [1:0] a;
wire [1:0] b;
reg [1:0] k;

function bit myfunction1;
  input a, b ;
  case ({a,b})
         1'b1 :   myfunction1 = a;
         default: myfunction1 = b;
  endcase
endfunction

function void myvoidfunction;
input a, b ;
output c;
c  = a + myfunction1({a,a},{b,b}) + p +
myfunction1({a,a},{b,b});
endfunction

function bit myfunction;
input a, b ;
case ({a,b})
       1'b1 :   begin
```

```
                    myfunction = a+b +p;
                    myvoidfunction({a,a},{b,b},k[1]);
                  end
        default:  myfunction = a+b +p;
endcase
endfunction
wire d = myfunction({a,a},{b,b});
endmodule
```

For the above example, the *Av_width_mismatch_function* rule reports the following violation messages:

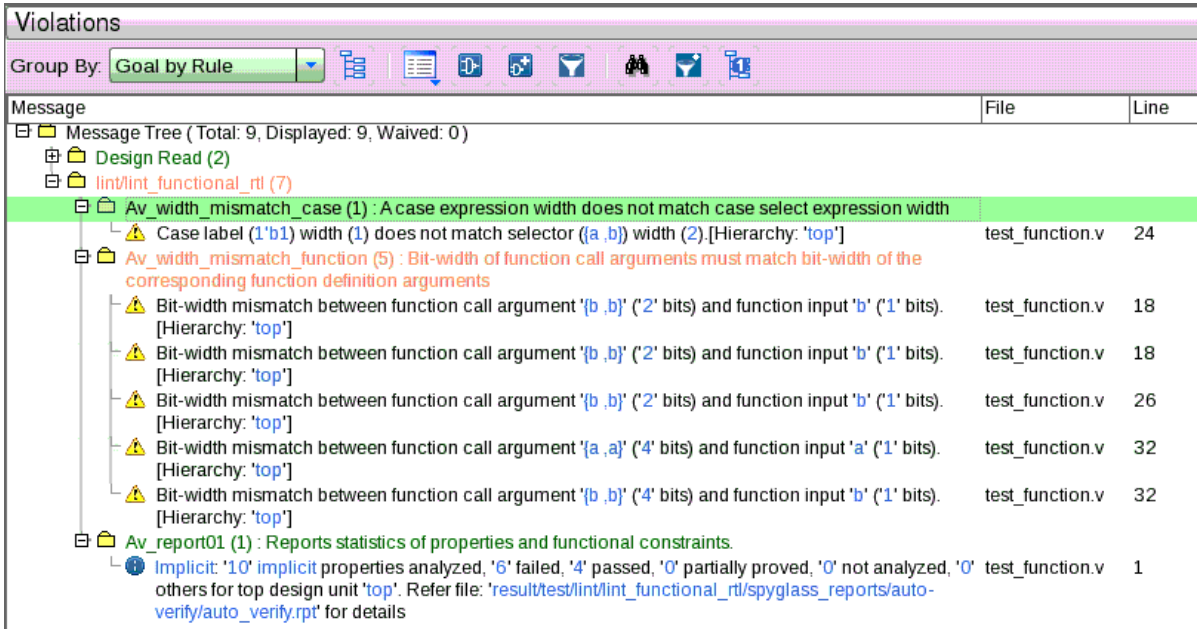


FIGURE 1. Av_width_mismatch_function Messages

For the *Av_width_mismatch_function* rule, the following *structural_rule_waiver.sgdc* file is used, which will filter the false violations reported by t he SpyGlass Lint rule:

```
waive -rule W263 -msg q%Case label (1'b1) width (1) does not
match selector ({a ,b}) width (2).[Hierarchy: 'all-
hierarchy']%

waive -rule STARC-2.1.3.1 -msg q%Bit-width mismatch between
function call argument '{a ,a}' ('2' bits) and function input
'a' ('1' bits).[Hierarchy: 'all-hierarchy']%

waive -rule STARC02-2.1.3.1 -msg q%Bit-width mismatch between
function call argument '{a ,a}' ('2' bits) and function input
'a' ('1' bits).[Hierarchy: 'all-hierarchy']%

waive -rule STARC05-2.1.3.1 -msg q%Bit-width mismatch between
function call argument '{a ,a}' ('2' bits) and function input
'a' ('1' bits).[Hierarchy: 'all-hierarchy']%

waive -rule STARC-2.1.3.1 -msg q%Bit-width mismatch between
function call argument '{a ,a}' ('2' bits) and function input
'a' ('1' bits).[Hierarchy: 'all-hierarchy']%

waive -rule STARC02-2.1.3.1 -msg q%Bit-width mismatch between
function call argument '{a ,a}' ('2' bits) and function input
'a' ('1' bits).[Hierarchy: 'all-hierarchy']%

waive -rule STARC05-2.1.3.1 -msg q%Bit-width mismatch between
function call argument '{a ,a}' ('2' bits) and function input
'a' ('1' bits).[Hierarchy: 'all-hierarchy']%

waive -rule STARC-2.1.3.1 -msg q%Bit-width mismatch between
function call argument '{a ,a}' ('2' bits) and function input
'a' ('1' bits).[Hierarchy: 'all-hierarchy']%

waive -rule STARC02-2.1.3.1 -msg q%Bit-width mismatch between
function call argument '{a ,a}' ('2' bits) and function input
'a' ('1' bits).[Hierarchy: 'all-hierarchy']%
```

```
waive -rule STARC05-2.1.3.1 -msg q%Bit-width mismatch between
function call argument '{a ,a}' ('2' bits) and function input
'a' ('1' bits).[Hierarchy: 'all-hierarchy']%
```

As shown in *Figure 1*, SpyGlass functional lint rule, *Av_width_mismatch_function,* filters three false violations reported by the SpyGlass Lint rules, STARC-2.1.3.1, STARC02-2.1.3.1, and STARC05-2.1.3.1.

The SpyGlass Functional Lint Rules

The SpyGlass Functional Lint has the following rules:

- *Av_width_mismatch_assign*
- *Av_width_mismatch_case*
- *Av_width_mismatch_port*
- *Av_width_mismatch_function*
- *Av_signed_unsigned_mismatch*
- *Av_width_mismatch_expr*
- *Av_width_mismatch_expr02*
- *Av_width_mismatch_expr03*
- *Av_case_default_redundant*
- *Av_case_default_missing*
- *Av_dontcare_mismatch*

Av_width_mismatch_assign

LHS width is less than RHS width of assignment (Truncation)

When to Use

Use this rule to identify assignments in which the LHS width is less than the RHS width.

Description

The *Av_width_mismatch_assign* rule reports assignments in which the LHS width is less than the (implied) width of the RHS expression.

NOTE: *The Av_width_mismatch_assign rule supports the TURBO_GROUP_MESSAGE noise reduction initiative. For details of this initiative, refer to the SpyGlass Turbo Lint Application Note.*

Parameter(s)

- `nocheckoverflow`: Default value is `no`. Set this parameter to `yes` to calculate the width of expressions as per LRM. Other possible value is the list of rules for which you want to set this parameter to `y es`.
- `check_unsign_overflow`: Default value is `no`. Set this parameter to `yes` to not suppress overflow in unsigned expressions when the `sign` extension is used, as shown in the following example:

```
wire [5:0] a,b;
wire [6:0] c;
wire [8:0] d;
assign c = {a[5],a} + {b[5],b};
```

- `strict`: Default value is `no`. In the default behavior, only the assignments in which the RHS expression contains wires or reg are checked. Set this parameter to `yes` to perform strict checking for all the assignments.
- `check_static_value`: Default value is `no`. Set this parameter to `yes` to report violations for static expressions and the non static expressions containing static expressions.

- check_counter_assignment: Default value is no. Set this parameter to yes to report violations for counter type of assignments, as shown in the following example:

  ```
  assign w1[3:0] = w1[3:0] + 1;
  ```

NOTE: *Ensure that the value of the check_static_value and check_counter_assignment parameters is set to yes.*

- check_counter_assignment_turbo: Default value is no. Set this parameter to yes or comma-separated list of rules to consider an assignment statement as counter type assignment in the following conditions:
  - Any constant (including based numbers or parameters) is added to or subtracted from a variable/signal on the RHS of the assignment.
  - The variable on RHS matches the variable on the LHS expression, that is, the same variable is used on both sides, LHS and RHS, of assignment with same bit-width.
- scope: Default value is block. Set this parameter to chip to consider the complete fan-in cone of an assertion.
- disable_rtl_deadcode: Default value is no. Set this parameter to yes to not make assertions for the disabled code (evaluated statically) using loops and conditional statements, such as if condition and ternary operator.
- use_lrm_width: Default value is no. Set this parameter to yes to consider the LRM width of integer constants, which is 32 bits.
- check_concat_max_width: Default value is no. In this case, no violation is reported when the width of the LHS expression is present between the width of the RHS expression without considering zero concatenated bits and the width of the RHS after adding zero concatenated bits.

  If you set this parameter to yes, the RHS width is considered as the width after adding zero concatenated bits. That is, the violation is reported if the LHS width does not match the RHS width after adding zero concatenated bits. This parameter is applicable for Verilog only.
- handle_shift_op: Default value is no. In this case, no violation is reported if the shifted or non-shifted width of a shift expression (at the port connection of a module instance) matches the width of the

corresponding module port definition. But the rule does not calculate the shifted width, if the RHS of the shift expression is non-static. Set this parameter to `shift_left`, `shift_right`, `shift_both`, `no_shift`, or comma separated list of rule names, to compare shifted or non shifted widths for left and right shift expressions.

- `av_seqdepth`: Specifies a sequential depth so that the input cone of properties can be abstracted by cutting the logic behind the specified depth in that cone (as specified by `av_seqdepth`) or till the module boundary (that is, `scope=block`), whichever is reached earlier.

Constraint(s)

- `clock`: Use this constraint to specify clock signals in a design.
- `reset`: Use this constraint to specify reset signals in a design.
- `set_case_analysis`: Use this constraint to specify case analysis conditions.
- `ip_block`: Use this constraint to specify module names so that no formal checking is done for the constructs inside those modules.

  The syntax of using the `ip_block` constraint is as follows:

  ```
  ip_block -name <module-name>
  ```

Messages and Suggested Fix

The following message appears to specify the width mismatch between LHS and RHS expressions:

[WARNING] LHS: '<lhs-expr>' width <lhs-width> is less than RHS: '<rhs-expr>' width <rhs-width> in assignment [Hierarchy: '<hier-path>']

This violation can be formally failed, partially-proved, not analyzed, or other by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues

This violation appears when the LHS width is less than the RHS width.

Consequences of Not Fixing

If you do not fix this violation, SpyGlass truncates some bits in the assignment.

How to Debug and Fix

Refer to the *auto_verify.rpt* report for details on this violation.

To debug this violation, trace the waveform viewer and check the values of the signals present in the RHS of the assignment, which are causing the width of RHS to be more than the width of LHS.

Example Code and/or Schematic

Consider the following example:

```
`define S0 2'b00
`define S1 2'b01
`define S2 2'b10
`define S3 2'b11

module top( input clk, rst,
            input [4:0]a1,
            input [4:0]a2,
            input [2:0]a3,
            output reg [3:0] out1,
            output reg [3:0] out2,
            output reg [2:0] out3
            );

    reg [1:0] state1;
    reg [1:0] state2;
    reg [1:0] state3;
    reg [1:0] state4;

    always @(posedge clk or posedge rst) begin
            if(rst) begin
                state1 <= `S0;
                out3 <= 3'b000;
            end
            else begin
```

```
                priority case(state1)
                        `S0: state1 <= `S1;
                        `S1: state1 <= `S2;
                        `S2: state1 <= `S0;
                default: begin
                    state3 <= `S3;
                out3 = 3'b11?;
            end
                endcase
                if(state1 >= a3 ) begin
                        out3 <= a1[1:0] + a3; //filtered violations
                if(a1 > out1)
                    out2 <= a2; //violations not filtered
                end
            end
    end
endmodule
```

For the above example, the *Av_width_mismatch_assign* rule reports following violation messages:

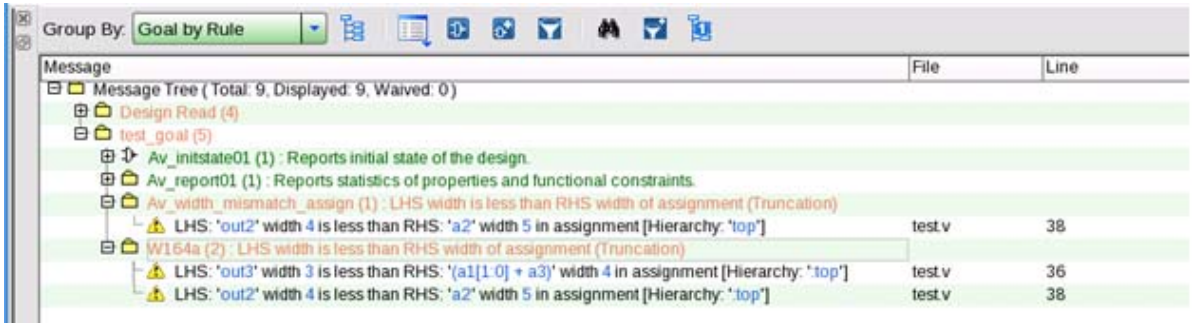


FIGURE 2. Av_width_mismatch_assign Messages

In the above example, violations filtered by SpyGlass Functional Lint are highlighted in green. However, violations not filtered by SpyGlass Functional Lint are highlighted in red.

As shown in the above figure, SpyGlass functional lint rule, *Av_width_mismatch_assign,* filters one violation reported by the SpyGlass

Lint rule, W164a.

Default Severity Label

Warning

Rule Group

Implicit-Properties

Reports and Related Files

No report or related file

Av_width_mismatch_case

A case expression width does not match case select expression width

When to Use

Use this rule to identify assignments in which a case expression width does not match with the select expression width.

Description

The *Av_width_mismatch_case* rule reports a violation when the `case` expression width does not match the `select` expression width.

NOTE: *The Av_width_mismatch_case rule supports the TURBO_CONSOLIDATE_DESIGN_SCOPE and TURBO_REMOVE_DUP_MSG noise reduction initiatives. For details of these initiatives, refer to the SpyGlass Turbo Lint Application Note.*

Parameter(s)

- `nocheckoverflow`: Default value is `no`. Set this parameter to `yes` to calculate the width of expressions as per LRM. Other possible value is the list of rules for which you want to set this parameter to `yes`.
- `scope`: Default value is `block`. Set this parameter to `chip` to consider the complete fan-in cone of an assertion.
- `use_lrm_width`: Default value is `no`. Set this parameter to `yes` to consider the LRM width of integer constants, which is 32 bits.
- `av_seqdepth`: Specifies a sequential depth so that the input cone of properties can be abstracted by cutting the logic behind the specified depth in that cone (as specified by `av_seqdepth`) or till the module boundary (that is, `scope=block`), whichever is reached earlier.

Constraint(s)

- `clock`: Use this constraint to specify clock signals in a design.
- `reset`: Use this constraint to specify reset signals in a design.
- `set_case_analysis`: Use this constraint to specify case analysis conditions.

- `ip_block`: Use this constraint to specify module names so that no formal checking is done for the constructs inside those modules.

  The syntax of using the `ip_block` constraint is as follows:

  ```
  ip_block -name <module-name>
  ```

Messages and Suggested Fix

The following message appears to specify the mismatch between the width of a `case` expression and a `select` expression:

[WARNING] Case label (<case-label>) width (<case-width>) does not match selector (<select-label>) width (<select-width>).[Hierarchy: '<hier-path>].

This violation can be formally failed, partially-proved, or not analyzed by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues.

This violation appears when the `case` expression width does not match the `select` expression width.

Consequences of Not Fixing

If you do not fix this violation, it may lead to unwanted target in `case` expressions during selection.

How to Debug and Fix

To debug this violation, trace the waveform viewer and check the values of the signals present in the `case` label or `case select` expression (whose width is more than the other).

To fix this violation, perform the following steps:

1. Double-click on the violation message.

This step highlights the line of `case` label in the *HDL Viewer* pane.

2. Scroll up in the *HDL Viewer* pane to view the `case` selector and the width mismatch.
3. Modify the case expression to match with the `select` expression in all bits.

   This makes the behavior of the RTL code easier to understand and does not rely on undefined behavior in handling unmatched bits.

Example Code and/or Schematic

Consider the following example:

```
module top(input actclk);
    wire clk;
    wire d1; reg q; reg [2:0]e;

    always @ (actclk or clk or d1)
      case (actclk)
        2'b00 : e <= d1;
     2'b01 : e <= d1 + 1;
        2'b11 : e <= d1 + 2;
      endcase

endmodule

module supertop(wire actclk, d, input [2:0]in_1, in_2, output
reg q);
    top inst1(actclk);
endmodule
```

For the above example, the *Av_width_mismatch_case* rule reports the following violation messages:

FIGURE 3. Av_width_mismatch_case Messages

In the above example, violations filtered by SpyGlass Functional Lint are highlighted in green. However, violations not filtered by SpyGlass Functional Lint are highlighted in red.

As shown in the above figure, SpyGlass functional lint rule, *Av_width_mismatch_case,* filters one violation reported by the SpyGlass Lint rule, *W253*.

Default Severity Label

Warning

Rule Group

Implicit-Properties

Reports and Related Files

No report or related file

Av_width_mismatch_port

An instance port connection has different width compared to the port definition

When to Use

Use this rule to identify the width mismatch between the bus width and instance port width while connecting a bus to an instance port.

Description

The *Av_width_mismatch_port* rule reports a violation if you try connecting a bus to an instance port but there is a mismatch between the bus width and the instance port width.

Parameter(s)

- `nocheckoverflow`: Default value is `no`. Set this parameter to `yes` to calculate the width of expressions as per LRM. Other possible value is the list of rules for which you want to set this parameter to `yes`.
- `report_blackbox_inst`: Default value is `no`. Set this parameter to `yes` to report violations for port width mismatch for black box instances.
- `scope`: Default value is `block`. Set this parameter to `chip` to consider the complete fan-in cone of an assertion.
- `use_lrm_width`: Default value is `no`. Set this parameter to `yes` to consider the LRM width of integer constants, which is 32 bits.
- `handle_shift_op`: Default value is `no`. In this case, no violation is reported if the shifted or non-shifted width of a shift expression (at the port connection of a module instance) matches the width of the corresponding module port definition. But the rule does not calculate the shifted width, if the RHS of the shift expression is non-static. Set this parameter to `shift_left`, `shift_right`, `shift_both`, `no_shift`, or comma separated list of rule names, to compare shifted or non shifted widths for left and right shift expressions.
- `report_blackbox_inst`: Default value is no. Set this parameter to yes to report violations for port width mismatch for black box instances.

- av_seqdepth: Specifies a sequential depth so that the input cone of properties can be abstracted by cutting the logic behind the specified depth in that cone (as specified by av_seqdepth) or till the module boundary (that is, scope=block), whichever is reached earlier.

Constraint(s)

- clock: Use this constraint to specify clock signals in a design.
- reset: Use this constraint to specify reset signals in a design.
- set_case_analysis: Use this constraint to specify case analysis conditions.
- ip_block: Use this constraint to specify module names so that no formal checking is done for the constructs inside those modules.

  The syntax of using the ip_block constraint is as follows:

  ```
  ip_block -name <module-name>
  ```

Messages and Suggested Fix

The following message appears if there is a mismatch between the bus width and instance port width:

[WARNING] Incompatible width for port '<port-name>'(width <port-width> in module '<module-name>') on instance '<instance-name>'(terminal width <instance-term-width>) [Hierarchy: '<hier-path>'].

This violation can be formally failed, partially-proved, or not analyzed by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues.

This violation appears when there is a width mismatch between the bus width and instance port width during connection of a bus to an instance port.

Consequences of Not Fixing

The excess bits on the bus are ignored if the bus is too wide to be connected to an input or the bus is too small to be connected to an output

How to Debug and Fix

To debug this violation, trace the waveform viewer to check the values of the signals present in the port connection.

To fix this violation, perform the following steps:

1. Double-click on the violation message.

   The *HDL Viewer* window highlights the module instance where a port connection has more width as compared to the port definition.
2. Specify correct number of bits across the hierarchy boundary, and then explicitly ignore the bits that are not required.

Example Code and/or Schematic

Consider the following message:

```
module top(input clk);
    wire [3:0] sel;
    wire signed   [4:0] a = -8;
    wire unsigned [4:0]a1;
    wire [4:0] b;
    wire [5:0] c;
    parameter p1 = 1;
    parameter p2 = 1;
    parameter p  = 1;
    slave inst(.a(a) , .aa(a1[p1:0]) , .b(b[p2:0]) ,
   .c(c[p:0]));
endmodule
module slave (input [3:0]a , input [3:0]aa , input [3:0]b ,
input [3:0]c);
endmodule
module supertop(input clk);
    top #(3,4,3) inst1(clk);
    top #(4,3,3) inst2(clk);
endmodule
```

For the above example, the *Av_width_mismatch_port* rule reports four violations, *Violation 1*, *Violation 2*, and *Two Violations Not Analyzed By the Formal Engine*.

However, no *Av_width_mismatch_port* rule violation is reported for the port connection `c[3:0]` through the `supertop.inst1` or `supertop.inst2` hierarchy.

Violation 1

```
Incompatible width for port 'aa'(width 4 in module 'slave') on
instance 'inst'(terminal width 5) [Hierarchy:
'supertop.inst2'].
```

In this case, the values of the signal present in the port connection may attain `10` (hex value) corresponding to the `a1[4:0]` signal for the `supertop.inst2` hierarchy whose width is more than the width of the `aa` port.

The following figure shows the waveform viewer of this violation:

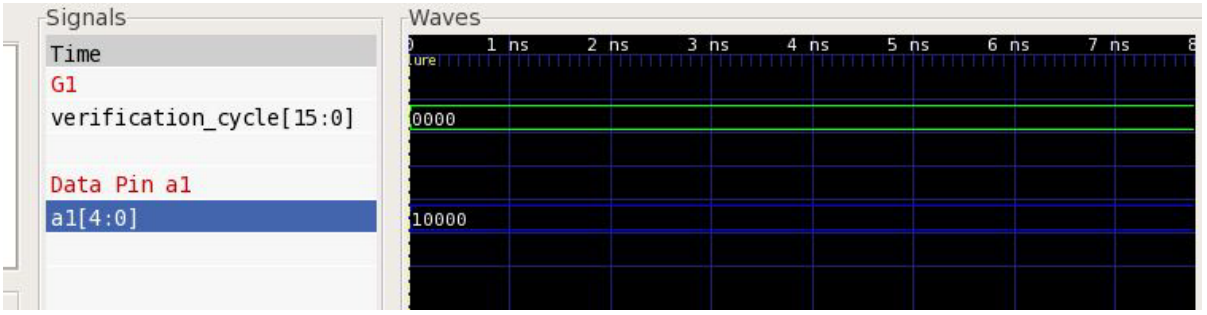


FIGURE 4. Waveform Viewer of the Av_width_mismatch_port Rule Violation - Example 1

Violation 2

```
Incompatible width for port 'b'(width 4 in module 'slave') on
instance 'inst'(terminal width 5) [Hierarchy:
'supertop.inst1'].
```

In this case, the values of the signals present in the port connection can be

16 corresponding to the `b[4:0]` signal for the `supertop.inst1` hierarchy whose width is more than the width of the `b` port in the `slave` module.

The following figure shows the waveform viewer of this violation:

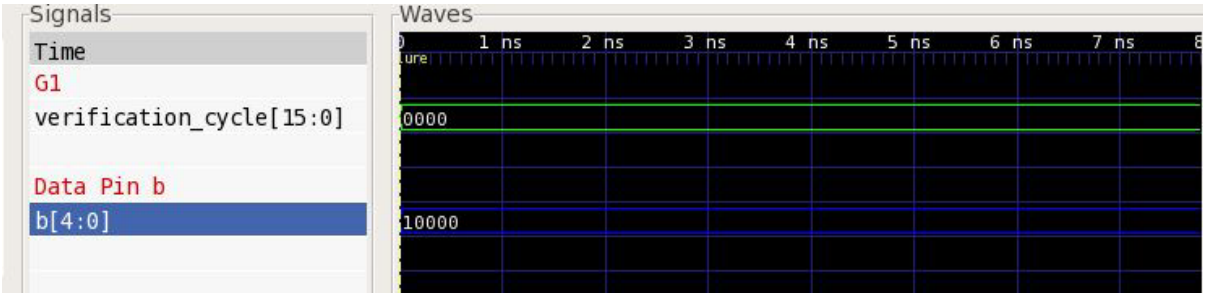


FIGURE 5. Waveform Viewer of the Av_width_mismatch_port Rule Violation - Example 2

Two Violations Not Analyzed By the Formal Engine

The following two violations are *not analyzed* by the formal engine:

```
Incompatible width for port 'a'(width 4 in module 'slave') on
instance 'inst'(terminal width 5) [Hierarchy:
':supertop:inst1@top']
```

```
Incompatible width for port 'a'(width 4 in module 'slave') on
instance 'inst'(terminal width 5) [Hierarchy:
':supertop:inst2@top']
```

The above violations appear because the width mismatch between the signed and unsigned expressions is not passed to the formal engine. Therefore, they are *not analyzed* by the formal engine.

Default Severity Label

Warning

Rule Group

Implicit-Properties

Reports and Related Files

No report or related file

Av_width_mismatch_function

Bit-width of function call arguments must match bit-width of the corresponding function definition arguments

When to Use

Use this rule to detect bit-width mismatches between the function input and the corresponding function call argument.

Description

The *Av_width_mismatch_function* rule reports bit-width mismatch between the function input and the corresponding function call argument.

NOTE: *The Av_width_mismatch_function rule supports the TURBO_GROUP_MESSAGE TURBO and TURBO_IGNORE_PADDING noise reduction initiatives. For details of these initiatives, refer to the SpyGlass Turbo Lint Application Note.*

Parameter(s)

- `nocheckoverflow`: Default value is `no`. Set this parameter to `yes` to calculate the width of expressions as per LRM. Other possible value is the list of rules for which you want to set this parameter to `yes`.
- `scope`: Default value is `block`. Set this parameter to `chip` to consider the complete fan-in cone of an assertion.
- `check_pad_concat`: The default value is `no` so that this rule ignores leading zeros in concat/multi-concat operator while width calculation. Set the value of the parameter to `yes` or <rule-name> to consider leading zeros in concatenation/multi-concatenation operator while width calculation.
- `av_seqdepth`: Specifies a sequential depth so that the input cone of properties can be abstracted by cutting the logic behind the specified depth in that cone (as specified by `av_seqdepth`) or till the module boundary (that is, `scope=block`), whichever is reached earlier.

Constraint(s)

- `clock`: Use this constraint to specify clock signals in a design.
- `reset`: Use this constraint to specify reset signals in a design.

- `set_case_analysis`: Use this constraint to specify case analysis conditions.
- `ip_block`: Use this constraint to specify module names so that no formal checking is done for the constructs inside those modules.

  The syntax of using the `ip_block` constraint is as follows:

  ```
  ip_block -name <module-name>
  ```

Messages and Suggested Fix

The following message appears if there is a bit-width mismatch between the function input and the corresponding function call argument:

[WARNING] Bit-width mismatch between function call argument '<func-call-arg>' ('<func-bits>' bits) and function input '<func-input>' ('<input-bits>' bits).[Hierarchy: '<hier>']

This violation can be formally failed, partially-proved, or not analyzed by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues

This violation appears if the bit-width of the function call arguments do not match with the bit-width of the corresponding function definition arguments.

Consequences of Not Fixing

If the function input is narrower than the function call argument, the value of the function call argument is left truncated and the function is evaluated on the truncated value.

How to Debug and Fix

Modify the code such that the bit-width of the function call arguments match with the bit-width of the corresponding function definition arguments.

Example Code and/or Schematic

Consider the following example:

```
module top(input clk, rst);
parameter p = 1'b1;
wire [1:0] a;
wire [1:0] b;
reg [1:0] k;

function bit myfunction1;
  input a, b ;
  case ({a,b})
         1'b1 :   myfunction1 = a;
         default: myfunction1 = b;
  endcase
endfunction

function void myvoidfunction;
input a, b ;
output c;
c  = a + myfunction1({a,a},{b,b}) + p +
myfunction1({a,a},{b,b});
endfunction

function bit myfunction;
input a, b ;
case ({a,b})
       1'b1 :   begin
                  myfunction = a+b +p;
                  myvoidfunction({a,a},{b,b},k[1]);
                end
       default:  myfunction = a+b +p;
endcase
endfunction
wire d = myfunction({a,a},{b,b});
endmodule
```

For the above example, the *Av_width_mismatch_function* rule reports the

following violation messages:

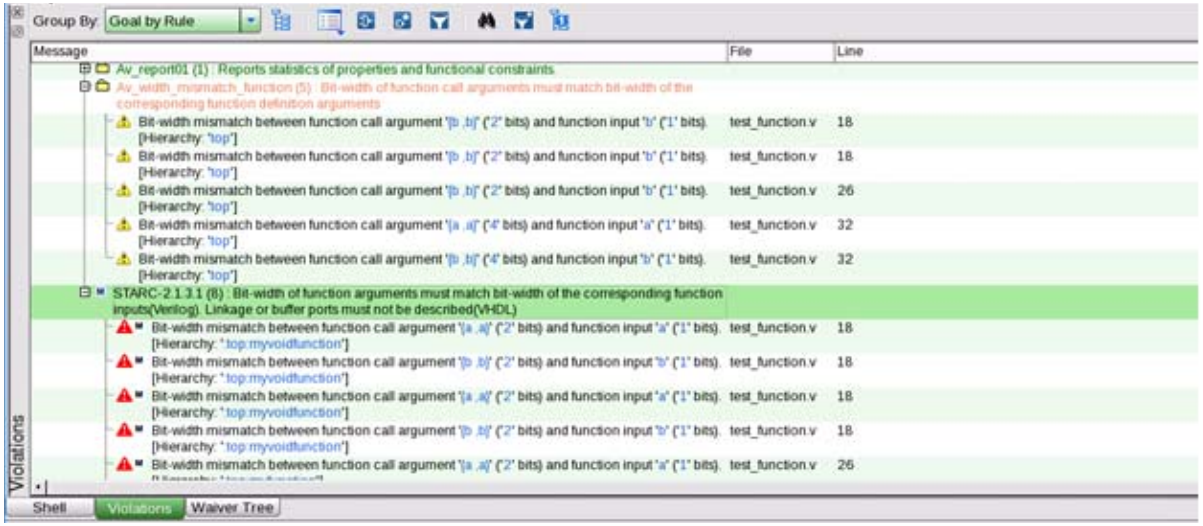


FIGURE 6. Av_width_mismatch_function Messages

As shown in the above figure, SpyGlass functional lint rule, *Av_width_mismatch_function,* filters three false violations reported by the SpyGlass Lint rule, STARC-2.1.3.1.

Default Severity Label

Warning

Rule Group

Implicit-Properties

Reports and Related Files

No report or related file

Av_signed_unsigned_mismatch

Mixed signed and unsigned types

When to Use

Use this rule to detect the cases in which signed and unsigned types are mixed.

Description

The *Av_signed_unsigned_mismatch* rule reports a violation if signals of the signed and unsigned types are mixed to form a single expression or there is a sign conversion in an assignment or a comparison statement.

NOTE: *The Av_signed_unsigned_mismatch rule supports the TURBO_IGNORE_STATIC_CONSTSANTS, TURBO_GROUP_MESSAGE, and TURBO_SET_RECOMMENDED_PARAMETER noise reduction initiatives. For details of these initiatives, refer to the SpyGlass Turbo Lint Application Note.*

The rule details are covered under the following topics:

- *Cases when the Type Conversions are Checked by this Rule*
- *Rule Assumptions to Avoid Spurious Violations*
- *Operators Ignored from Rule Checking*
- *Considering Integer Constants as Signed*
- *Ignoring Violations for the for-loop and while-loop Indexes*

Cases when the Type Conversions are Checked by this Rule

This rule checks for the type conversions, such as signed to unsigned or unsigned to signed, in the following cases:

- Sign conversion in an assignment (only if strict is set)
- Sign conversion during comparison in constructs, such as `case` and `if`.
- Sign conversion during binary operations in an expression/sub-expression

Rule Assumptions to Avoid Spurious Violations

This rule considers an unsized integer constant, such as 4 as both signed and unsigned.

Operators Ignored from Rule Checking

This rule does not report for the following operators:

- Right shift operators both logical and arithmetic i.e. ">>" & ">>>"
- Left shift operators both logical and arithmetic i.e. "<<" & "<<<"
- Power operator "**"

Considering Integer Constants as Signed

Set the dccompat parameter to yes to consider unsized integer constants as signed.

Ignoring Violations for the for-loop and while-loop Indexes

Set the ignoreforindex parameter to yes to ignore violations for the for-loop and while-loop indexes.

When a bit select or a part select is used as a loop index, violations for the entire array are ignored. The following code snippet shows such array:

```
integer j[3];
 for(j[0] = 1; j[0] < 3; j[0]++)
  j[1] = j[1] + 3'b111;
```

In the above example, no Av_signed_unsigned_mismatch violation appears for all the array bits in j, because j is marked as for/while loop index.

Parameter(s)

- nocheckoverflow: Default value is no. Set this parameter to yes to calculate the width of expressions as per LRM. Other possible value is the list of rules for which you want to set this parameter to yes.
- scope: Default value is block. Set this parameter to chip to consider the complete fan-in cone of an assertion.
- dccompat: Default value is no. Set this parameter to yes to follow Design Compiler (DC) conventions for rule checking.

- `ignoreforindex`: Default value is `no`. Set this parameter to `yes` to ignore violations for the for-loop and while-loop indexes. For details, see *Ignoring Violations for the for-loop and while-loop Indexes*.
- `strict`: Default value is `no`. Set this parameter to `yes` to check all the assignments in addition to wire and reg in RHS.
- `av_seqdepth`: Specifies a sequential depth so that the input cone of properties can be abstracted by cutting the logic behind the specified depth in that cone (as specified by `av_seqdepth`) or till the module boundary (that is, `scope=block`), whichever is reached earlier.

Constraint(s)

- `clock`: Use this constraint to specify clock signals in a design.
- `reset`: Use this constraint to specify reset signals in a design.
- `set_case_analysis`: Use this constraint to specify case analysis conditions.
- `ip_block`: Use this constraint to specify module names so that no formal checking is done for the constructs inside those modules.

  The syntax of using the `ip_block` constraint is as follows:

  ```
  ip_block -name <module-name>
  ```

Messages and Suggested Fix

This rule reports the following message:

[WARNING] Unsigned expression '<expr1>' used with signed expression '<expr2>'<stmt>

Where, <*stmt*> can be in assignment or in comparison depending upon whether the type (signed or unsigned) conversion has occurred in an assignment statement or in a comparison statement.

This violation can be formally failed, partially-proved, or not analyzed by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues

This violation appears when an unsigned expression is used with a signed expression.

How to Debug and Fix

Either use signed or unsigned types in an expression. Do not mix the signals of the signed and unsigned types to form a single expression.

Example Code and/or Schematic

Consider the following example:

```
module top( input clk, rst,
            input a1,
            output reg b1
            );

    //Av_signed_unsigned_mismatch
    test_sub test_sub_instance(a1,b1);

endmodule

module test_sub(d, q);
  input d;
  output q;
  reg q;
  reg r, clk;
  integer i, j;
  wire wireVar;
  wire a = i + wireVar + j;
  always @(posedge clk)
  begin
    if(r)
      q = d<<j;
  end
endmodule
```

For the above example, the *Av_signed_unsigned_mismatch* rule reports the following violation messages:

FIGURE 7. Av_signed_unsigned_mismatch Messages

As shown in the above figure, SpyGlass functional lint rule, *Av_signed_unsigned_mismatch*, filters one false violation reported by the SpyGlass Lint rule, SignedUnsignedExpr-ML.

Default Severity Label

Warning

Rule Group

Implicit-Properties

Reports and Related Files

No report or related file

Av_width_mismatch_expr

Bit-width of operands of a logical operator do not match

When to Use

Use this rule to detect bit-width mismatch between the operands of logical operators.

Description

The *Av_width_mismatch_expr* rule reports bit-width mismatches between the operand expressions of logical operations.

Width Calculation Approach

Set the nocheckoverflow parameter to yes to calculate width according to LRM.

The width-calculation approach is as follows:

Operator	LRM Width	Normal Width
+,-	Max (lhswidth,rhswidth)	Max (lhswidth,rhswidth) + 1
*	Max (lhswidth,rhswidth)	lhswidth+rhswidth
/	Max (lhswidth,rhswidth)	lhswidth
%	Max (lhswidth,rhswidth)	rhswidth

Example:

In the following expression, this rule reports a violation when the nocheckoverflow parameter is set to no:

```
(in1[3:2] + in1[1]) && in2[3:2]
```

To suppress violation for the above example, set the nocheckoverflow parameter to yes.

Rule Exceptions

- If an integer constant is used in one of the operands, this rule does not report a violation if the width of that constant is less than or equal to the width of the other operand.

- This rule does not report violations for the expressions where the width of the constant expression, constant integer, or base number is less than the width of the other operand.

  Example:

```
assign out = base [4:0] && (1023>>6) ; //No Violation
assign out1 = base1 [4:0] || 10 ; //No Violation
assign out2 = base2 [4:0] || 1023; //Violation
```

Parameter(s)

- nocheckoverflow: Default value is no. Set this parameter to yes to calculate the width of expressions as per LRM. Other possible value is the list of rules for which you want to set this parameter to yes.
- scope: Default value is block. Set this parameter to chip to consider the complete fan-in cone of an assertion.
- disable_rtl_deadcode: Default value is no. Set this parameter to yes to not make assertions for the disabled code (evaluated statically) using loops and conditional statements, such as if condition and ternary operator.
- use_lrm_width: Default value is no. Set this parameter to yes to consider the LRM width of integer constants, which is 32 bits.
- av_seqdepth: Specifies a sequential depth so that the input cone of properties can be abstracted by cutting the logic behind the specified depth in that cone (as specified by av_seqdepth) or till the module boundary (that is, scope=block), whichever is reached earlier.

Constraint(s)

- clock: Use this constraint to specify clock signals in a design.
- reset: Use this constraint to specify reset signals in a design.
- set_case_analysis: Use this constraint to specify case analysis conditions.
- ip_block: Use this constraint to specify module names so that no formal checking is done for the constructs inside those modules.

  The syntax of using the ip_block constraint is as follows:

```
ip_block -name <module-name>
```

Messages and Suggested Fix

This rule reports the following message:

[ERROR] Operand bit-width mismatch for operator '<op-name>': '<op1-exp>'('<op1-width>' bits) and '<op2-exp>'('<op2-width>' bits). [Hierarchy: <hier-path>]

This violation can be formally failed, partially-proved, or not analyzed by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues

This violation appears if there are bit-width mismatches between the operand expressions of logical operations.

Consequences of Not Fixing

Bit-width mismatch between operands of a logical operation may lead to incorrect results.

How to Debug and Fix

Review the code and correct the reported mismatch.

Example Code and/or Schematic

Consider the following example:

```
module top();
wire [2:0]c;
wire [2:0]d;
wire [3:0]e;

function bit myfunction1;
  input [2:0]a; input [3:0] b ;
  case ({a,b})
         1'b1 :  myfunction1 = a || b;
```

```
            default: myfunction1 = b || {a,a};
  endcase
endfunction
assign c = { myfunction1(d,e) , myfunction1(d,e) ,
myfunction1(d,e)};
endmodule
```

For the above example, the *Av_width_mismatch_expr* rule reports the following violation messages:

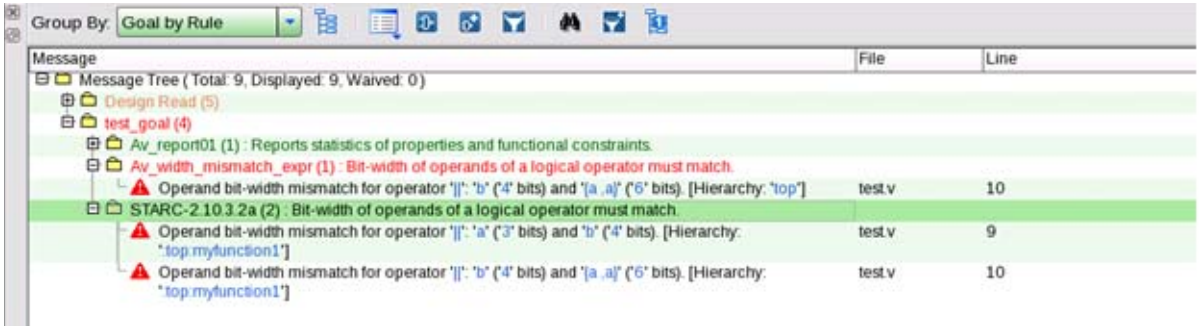


FIGURE 8. Av_width_mismatch_expr Messages

In the above example, violations filtered by SpyGlass Functional Lint are highlighted in green. However, violations not filtered by SpyGlass Functional Lint are highlighted in red.

As shown in the above figure, SpyGlass functional lint rule, *Av_width_mismatch_expr*, filters one false violation reported by the SpyGlass Lint rule, STARC-2.10.3.2a.

Default Severity Label

Error

Rule Group

Implicit-Properties

Reports and Related Files

No report or related file

Av_width_mismatch_expr02

Identifies the unequal length operands in the bit-wise logical, arithmetic, and ternary operators

When to Use

Use this rule to identify bit-width mismatch between operands of the bit-wise logical, arithmetic, and ternary operators.

Description

The *Av_width_mismatch_expr02* rule flags bit-width mismatch between operands of bit-wise logical, arithmetic, or ternary operators.

Following is the list of operators covered under the *Av_width_mismatch_expr02* rule:

Arithmetic Operators	
Subtraction (-)	Addition (+)
Multiplication (*)	Division (/)
Modulus (%)	
Bit-wise Operators	
bit-wise xor (^)	bit-wise negation (~)
bit-wise and (&)	bit-wis e or (|)
Ternary Operator (? :)	

NOTE: *If the count of operators is more than 500 in an expression, then the expression is ignored for rule checking.*

Width Calculation

If you set the value of the `nocheckoverflow` parameter to `yes` or *Av_width_mismatch_expr02*, the *Av_width_mismatch_expr02* rule checks the bit-width as per the LRM, as shown in the following table:

Operator	LRM width	Normal width
+, -	Max (LHS width, RHS width)	Max (LHS width, RHS width)
*	Max (LHS width, RHS width)	LHS width + RHS width

Operator	LRM width	Normal width
/	Max (LHS width, RHS width)	LHS width
%	Max (LHS width, RHS width)	RHS width

For constants, the natural width is considered.

NOTE: *The Av_width_mismatch_expr02 rule does not check for expressions in signal or variable indexes, such as bit-select expression.*

NOTE: *This rule does not report a violation for integer variable expressions.*

Rule Exceptions

The *Av_width_mismatch_expr02* rule does not report violations for expressions where the width of a constant expression, a constant integer, or a base number is less than the width of the other operand.

Language

Verilog

Default Weight

10

Parameter(s)

- `check_static_value`: Default value is no. Set the value of the parameter to `yes` or `<rule_list>` to report violation for cases with width mismatch, involving static expressions and non-static expressions having a static part. Other possible values are `only_const` and `only_expr`. For an expression having one operand constant and other non constant, a violation is reported only when the width of the constant operand is greater.
- `disable_rtl_deadcode`: The default value is no. Set the value of the parameter to yes to disable violations for disabled code in loops and conditional (if condition, ternary operator) statements.
- `reportconstassign`: The default value is no. Set the value of the parameter to yes to enable the *Av_width_mismatch_expr02* rule to check for constants whose width is less than the operand.
- `strict`: Default value is `no`. Therefore, the rule behavior is as follows:
  - ❒ Ignores addition (+) and multiplication (*) operations

- ❒ Reports violation on subtraction (-), division (/), or modulus (%) operations only if the width of the right operand is greater than the width of the left operand.

Set this parameter to `yes` to check for the addition and multiplication operations and to report subtraction, division, or modulus operations when there is a width mismatch between operands (both `A > B` and `B > A` for operations `A-B`, `A/B`, and `A%B`). You can also set this parameter to check for ternary operators

- `use_lrm_width`: Default value is `no`. Set this parameter to `yes` to consider the LRM width of integer constants, which is 32 bits. The rule does not check the bit-width as per LRM by default.
- `nocheckoverflow`: Default value is `no`. Set the value of the parameter to yes or rule name to calculate the width as per the LRM. See *Width Calculation* to know more about calculating width.
- `use_carry_bit`: Default value is `no` and the width is taken as maximum of the two operands for a binary expression having plus and minus operators. Set this parameter to `yes` or `<rule-name>` to get width after considering the carry bit of addition. No violation is reported, even using this parameter, for sub-expressions of a binary expression if all terms have the same width and all operators are either plus or minus. Also, refer to *Examples Code and/or Schematic*.

Constraint(s)

None

Messages and Suggested Fix

The following message appears at the location of an operation of operator *<opr-name>* where there is a bit-width mismatch between left expression *<exprl>* of bit-width *<bit-widthl>* and right expression *<exprr>* of bit-width *<bit-widthr>*:

[WARNING] For operator (<opr-name>), left expression: "<exprl>" width <bit-widthl> should match right expression: "<exprr>" width <bit-widthr>. [Hierarchy: '<hier-path>']

Where, *<hier-path>* is the complete hierarchical path of the containing scope.

This violation can be formally failed, partially-proved, not analyzed, or other by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues

A violation is reported when there is a bit-width mismatch between the left expression of bit-width and right-expression of bit-width.

Consequences of Not Fixing

While working with expressions of different bit-widths may be the intended behavior, it is also a potentially error-prone design practice. For example, the addition of two words of unequal widths may indicate that you forgot to update the width of one of the buses.

How to Debug and Fix

Double-click the violation message. The HDL window highlights the line, where, width mismatch in operators is detected.

To resolve the violation, check each case for potential messages, especially in the bit-wise operators. Review the RTL code mentioned in message, this code may cause some unintended behavior. Make all arguments in such comparisons of equal width such as by explicitly extending narrower operators in a concatenation, to see cases where upper bits are zeroed.

Examples Code and/or Schematic

Consider the following example:

```
`define S0 2'b00
`define S1 2'b01
`define S2 2'b10
`define S3 2'b11

module top( input clk, rst,
            input [4:0]a1,
            input [4:0]a2,
            input [2:0]dib,
```

```
        output reg [2:0] out1,
        output reg [2:0] out2,
        output reg [2:0] out3
        );

reg [1:0] state1;
reg [1:0] state2;
reg [1:0] state3;
reg [1:0] state4;

always @(posedge clk or posedge rst) begin
        if(rst) begin
            state1 <= `S0;
            out1 <= 3'b000;
        end
        else begin
            priority case(state1)
                    `S0: state1 <= `S1;
                    `S1: state1 <= `S2;
                    `S2: state1 <= `S0;
            endcase
            if(state1 >= dib ) begin
                    out1 <= a1[1:0] + dib;
            end
        end
end

always @(posedge clk or posedge rst) begin
        if(rst) begin
            state2 <= `S0;
        end
        else if (a2[2]) begin
            case (state2)
                    `S0: state2 <= `S1;
                    `S1: state2 <= `S2;
                    `S2: state2 <= `S3;
            endcase
        end else if (state2 >= a2[2:0] )begin
```

```
                    out2 <= a1[1:0] - dib;
                end
            end
        endmodule
```

Now, consider the value of the `strict` parameter is set to yes.

The *Av_width_mismatch_expr02* rule generates the following violation messages:

FIGURE 9. The Av_width_mismatch_expr02 Messages

In the above example, violations filtered by SpyGlass Functional Lint are highlighted in green. However, violations not filtered by SpyGlass Functional Lint are highlighted in red.

As shown in the above figure, SpyGlass functional lint rule, *Av_width_mismatch_expr02*, filters one false violation reported by the SpyGlass Lint rule, W116.

Default Severity Label

Warning

Rule Group

Implicit-properties

Reports and Related Files

No related reports or files.

Av_width_mismatch_expr03

Reports an arithmetic comparison operator with unequal length

When to Use

Use this rule to identify arithmetic comparison operator with unequal length.

Description

The *Av_width_mismatch_expr03* rule reports arithmetic comparison operations with operands of unequal widths. Following is the list of operators covered under this rule:

Relational Operators	
Greater than (>)	Greater than or equal to (>=)
Less than (<)	Less than or equal to (<=)
Equality Operators	
logical Equality(==)	logical inequality(!=)
case equality(===)	case inequality(!==)

NOTE: *If the count of operators is more than 500 in an expression, then the expression is ignored for rule checking.*

If the `nocheckoverflow` parameter is set to `yes` or *Av_width_mismatch_expr03*, the width of the expression is calculated as per the LRM. However, for constants, normal width is considered.

Operator	LRM Width	Normal Width
+, -	Max (lhswidth, rhswidth)	Max (lhswidth, rhswidth) + 1
*	Max (, rhswidth)	lhswidth + rhswidth
/	Max (lhswidth, rhswidth)	lhswidth
%	Max (lhswidth, rhswidth)	rhswidth

Handling of Unary Negation:

In case of unary negation, the width is calculated as follows:

- **Variable and based numbers:** If an operand is a variable or a based number, the width is incremented by one. For example, width of `-a[3:0]` is 5, and width of `-3'b101` is 4, etc. This is applicable only when the `nocheckoverflow` parameter is set to `no`.
- **Constant and unsized based number:** In case of constants, the natural width is calculated first and is then incremented by one. For example, width of `-9` is 5. An unsized based number is treated similarly. For example, width of `-'b101` is 4. However, if you set the `use_lrm_width` parameter to `yes`, the width is considered as 32 bit.

Rule Exceptions

The *Av_width_mismatch_expr03* rule does not report violation for the following cases:

- counter cases, for example, data == data+1.
- If any one operand is of integer data type. See *Example Code and/or Schematic* for details.

Language

Verilog

Parameters

- `check_sign_extend`: Default value is no. Set the value of the parameter to yes to check for width mismatch due to sign extension in signed comparisons.
- `disable_rtl_deadcode`: The default value is `no`. Set the value of the parameter to yes to disable violations for disabled code in loops and conditional (if condition, ternary operator) statements.
- `use_lrm_width`: Default value is `no`. This indicates the *Av_width_mismatch_expr03* rule considers the natural width of integer constants. Set this parameter to `yes` to consider the LRM width, which is 32 bits.
- `nocheckoverflow`: Default value is `no`. This indicates the *Av_width_mismatch_expr03* rule does not check the bit-width as per LRM. Set this parameter to yes or rule name to check the bit-width as per LRM.

- `check_static_value` and `strict`: By default, the *Av_width_mismatch_expr03* rule does not report violation when the right or left expression is a constant, including parameter, sized or unsized based number, and unsized integer. Setting the `check_static_value` parameter to `yes` changes this behavior, and setting the `strict` parameter in addition further alters the behavior. The following table summarizes these variations in behavior:

Type of left or right expression	**check_static_value set to** `no`	***check_static_value* set to** `yes`	
		***strict* set to** `no`	***strict* set to** `yes`
Parameter	Does not report	Reports if the width of constant is larger	Reports any width mismatch
Sized based number (8'h15)	Does not report	Reports if the width of constant is larger	Reports any width mismatch
Unsized based number ('h15)	Does not report	Reports if the width of constant is larger	Reports if the width of constant is larger
Unsized integer (18)	Does not report	Does not report	Reports if the width of constant is larger

  See *Example Code and/or Schematic* for details. Also, when set to `yes`, the `check_static_value` parameter checks for width mismatch involving static expressions and non-static expressions that contain a static part. Refer to the `check_static_value` section for more details.

NOTE: *When the strict parameter is set, the Av_width_mismatch_expr03 rule does not report violation for width mismatch in the for loop condition.*

Constraints

None

Messages and Suggested Fix

The following message appears at the location where a width *<widthl>* of a left expression *<exprl>* does not match a width *<widthr>* of a right expression *<exprr>* in an operation of an arithmetic operator *<op-name>*.

[WARNING] For operator (<op-name>), left expression: "<exprl>" width <widthl> should match right expression: "<exprr>" width <widthr> [Hierarchy: '<hier-path>']

Where, *<hier-path>* is the complete hierarchical path.

This violation can be formally failed, partially-proved, not analyzed, or other by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues

A violation is reported when an arithmetic operation has operands of unequal length.

Consequences of Not Fixing

For some range of values of the wider operand in arithmetic comparison operations, the comparison operation evaluates to a constant, independent of the value of the narrower operand. This may result in an unexpected behavior.

How to Debug and Fix

Double-click the violation message. The HDL Viewer window highlights the line where the width mismatch is found for comparison operator.

This is not a major issue. However, you can avoid possible problems by explicitly comparing sub-expressions of equal width. This also enhances the code readability.

Example Code and/or Schematic

Consider the following example:

```
`define S0 2'b00
`define S1 2'b01
`define S2 2'b10
`define S3 2'b11
module top( input clk, rst,
     input [4:0]a1,
     input [4:0]a2,
     input [2:0]dib,
     output reg [2:0] out1,
     output reg [2:0] out2,
     output reg [2:0] out3
        );
      reg [1:0] state1;
     reg [1:0] state2;
     reg [1:0] state3;
     reg [1:0] state4;
   always @(posedge clk or posedge rst) begin
       if(rst) begin
            state1 <= `S0;
              out1 <= 3'b000;
            end
            else begin
                priority case(state1)
                        `S0: state1 <= `S1;
                        `S1: state1 <= `S2;
                        `S2: state1 <= `S0;
                endcase
                if(state1 >= dib ) begin
                        out1 <= a1[1:0] + dib;
                end
            end
    end
    always @(posedge clk or posedge rst) begin
            if(rst) begin
                state2 <= `S0;
            end
            else if (a2[2]) begin
                case (state2)
```

```
                    `S0: state2 <= `S1;
                    `S1: state2 <= `S2;
                    `S2: state2 <= `S3;
                endcase
            end else if (state2 >= a2[2:0] )begin
                    out2 <= a1[1:0] - dib;
            end
        endendmodule
```

For the above example, the *Av_width_mismatch_expr03* rule reports the following violation messages:

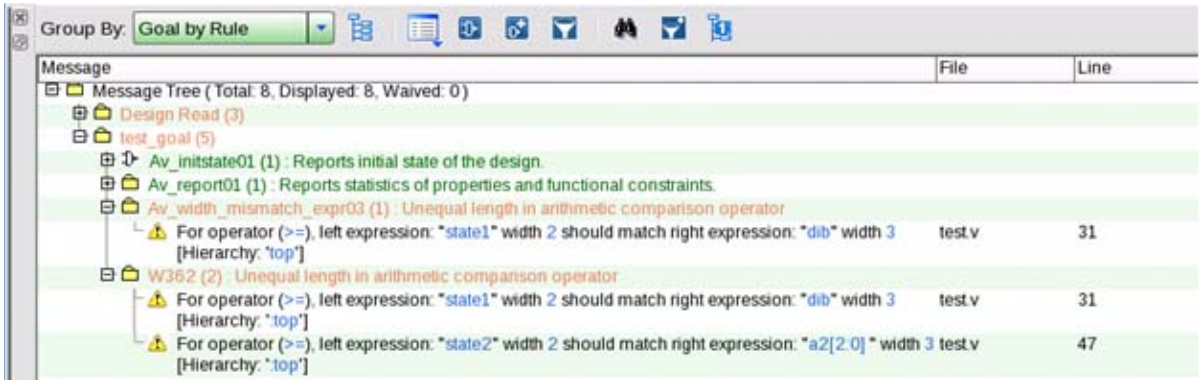


FIGURE 10. Av_width_mismatch_expr03 Messages

In the above example, violations filtered by SpyGlass Functional Lint are highlighted in green. However, violations not filtered by SpyGlass Functional Lint are highlighted in red.

As shown in the above figure, SpyGlass functional lint rule, *Av_width_mismatch_expr03*, filters one false violation reported by the SpyGlass Lint rule, W362.

Default Severity Label

Warning

Rule Group

Implicit-properties

Reports and Related Files

None

Av_case_default_redundant

Ensure that a case statement marked full_case or a priority/unique case statement does not have a default clause.

When to Use

Use this rule to identify the case statements that are marked either `full_case` or `priority/unique_case` and have a `default` clause.

Description

The *Av_case_default_redundant* rule reports violation for the following constructs with a default clause:

- case constructs with full_case pragma
- priority/unique case constructs

The *Av_case_default_redundant* rule checks for priority cases inside always blocks, initial blocks, tasks and functions in all scopes like generate block, packages, global scope, and interfaces.

The rule reports violation when a priority modifier is used with `case`, `casex`, or `casez` statements with `default` clause as one of its case selection item.

Language

Verilog

Default Weight

5

Parameter(s)

`check_case_type`: Default value is all. Therefore, the *Av_case_default_redundant* rule checks for priority, unique, and full_case cases. Set the value of the parameter to either priority or unique or full_case to check for priority, unique and full_case cases, respectively. You can also specify multiple values for the parameter so as to check for the specified cases.

Constraint(s)

None

Messages and Suggested Fix

The following message appears at the first line of a case construct with the default clause and full_case pragma or priority/unique case construct with a default clause:

[WARNING] Case statement marked <type> has a default clause

Where, <*type*> can be `full_case` or `priority_case` or `unique_case`.

This violation can be formally failed, partially-proved, not analyzed, or other by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues

The *Av_case_default_redundant* rule may report a violation because of the following reasons:

- Full_case pragma or the default clause is redundant.
- In case of priority/unique case the default clause is redundant.

Consequences of Not Fixing

This rule points to the need of intent review. If the designer meant to specify full_case then there should not be a reason for a default clause. A review at the RTL coding stage can help you uncover subtle design issues.

How to Debug and Fix

Use either the full_case pragma directive or the default clause in a case construct. For priority/unique case constructs, default clause is redundant.

To fix the violation, remove either the full_case pragma or the default clause.

Example Code and/or Schematic

Consider the following example:

```
module top( input clk, rst,
            input [4:0]a1,
            input [4:0]a6p,
```

```
            input [4:0]a7p,
            output reg [3:0]b6p,
            output reg [3:0]b7p);
     always @ (a1 or a6p or a7p)
       case (a1) //synopsys full_case
         2'b00 : b6p <= a6p;
      2'b01 : b7p <= a7p;
      2'b11 : b7p <= a6p + a7p;
      2'b10 : b7p <= a7p - a6p;
      default: b7p <= 2'b11;
    endcase

     always @ (a1 or a6p or a7p)
       casez (a1[1]) //synopsys full_case
         1'b0 : b6p <= a6p;
          1'b1 : b7p <= a6p + a7p;
          default: b7p <= 2'b11;
        endcase
endmodule
```

In the above example, the *Av_case_default_redundant* rule reports following violation messages:

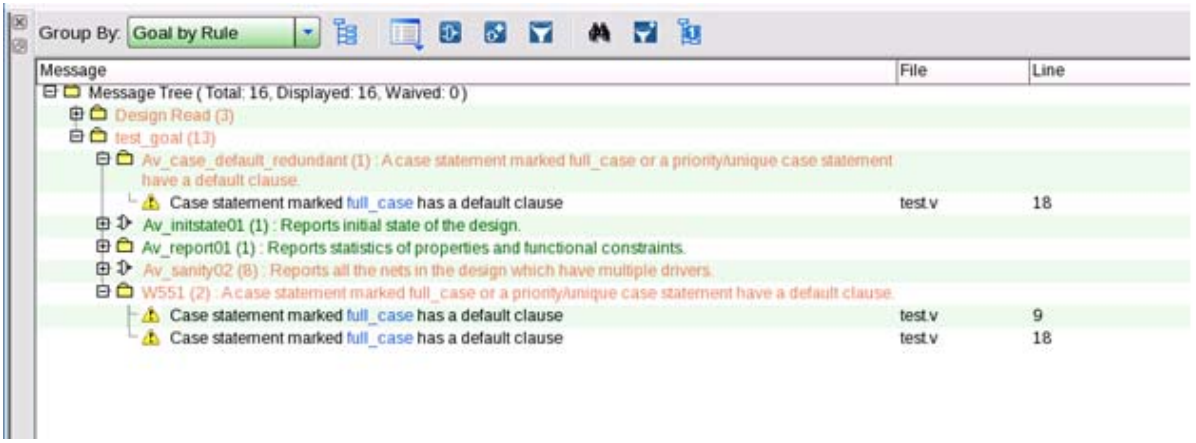


FIGURE 11. Av_case_default_redundant Messages

In the above example, violations filtered by SpyGlass Functional Lint are

highlighted in green. However, violations not filtered by SpyGlass Functional Lint are highlighted in red.

As shown in the above figure, SpyGlass functional lint rule, *Av_case_default_redundant*, filters one false violation reported by the SpyGlass Lint rule, W551.

Default Severity Label

Warning

Rule Group

Implicit-properties

Reports and Related Files

No related reports or files.

Av_case_default_missing

Ensure that a case statement or a selected signal assignment has a default clause

When to Use

Use this rule to identify case constructs without the default clause.

Description

The *Av_case_default_missing* rule reports violation for case constructs without the default clause and case constructs and selected signal statements.

A default clause should always be specified in a case construct to handle unexpected situations even if the construct covers all potential situations.

NOTE: *The Av_case_default_missing rule supports generate-if and generate-for block.*

Rule Exceptions

The *Av_case_default_missing* rule does not report for missing default clause in the following cases:

- If the target signals in the case construct are assigned using a blocking or non-blocking assignment statement before the case statement
- Case constructs that are inside always construct that infer a flip-flop (Set `check_sequential` rule parameter to report such cases).
- Case constructs with associated full_case pragma and unique / priority cases.
- Fully-specified case constructs.
- Case statements with static case select (including generate case).

Language

Verilog

Default Weight

5

Parameter(s)

- `strict`: The default value is no. Set the value of the parameter to yes to report fully-specified case constructs without the default clause.

- `check_sequential`: The default value is no. Set the value of the parameter to yes to enable rule checking in sequential block.

Constraint(s)

None

Messages and Suggested Fix

The following message appears at the location where a case construct is defined without a default clause:

[WARNING] Case statement does not have a default clause and is not preceded by assignment of target signal <block-type> [Hierarchy: '<hier-path>']

Where, *<block-type>* can be either a sequential or a combinational block and the *<hier-path>* is the complete hierarchical name of the containing scope excluding subprograms.

This violation can be formally failed, partially-proved, not analyzed, or other by the formal engine. The not-analyzed violations appear when the width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues

A violation is reported when a `case` construct is defined without a `default` clause.

Consequences of Not Fixing

If all possible cases of the case construct selector are covered, this is not directly an error. However, if the case construct selector has an undefined value (`X` or `Z`) and there is no default clause, then the design simulation may produce unexpected results.

If the width of the case construct selector changes as the design evolves, then what had once been a fully covered case construct may become only partially covered and can lead to inferred latches. Hence, it is recommended to always describe a default clause even if all possible cases of

the case construct selector are described.

How to Debug and Fix

Double-click the violation message. The HDL Viewer window is displayed. The HDL Viewer window highlights the line where the case statement with missing default is declared.

To fix the violation, add a default clause to specify default behavior. If you are specifying simulation X behavior, bracket this behavior in `translate_off` and `translate_on` pragmas.

Example Code and/or Schematic

Consider the following example:

```
module top( input clk, rst,
            input [4:0]a1,
            input [4:0]a6p,
            input [4:0]a7p,
            output reg [3:0]b6p,
            output reg [3:0]b7p);

    always @ (a1 or a6p or a7p)
      case (a1)
        2'b00 : b6p <= a6p;
        2'b01 : b7p <= a7p;
        2'b11 : b7p <= a6p + a7p;
       endcase

    always @ (a1 or a6p or a7p)
      case (a1[0])
     1'b0 : b6p <= a6p;
     1'b1 : b7p <= a6p + a7p;
    endcase
endmodule
```

Now, consider that the value of the `strict` parameter is set to yes.

For this example, the *Av_case_default_missing* generates the following violation messages:

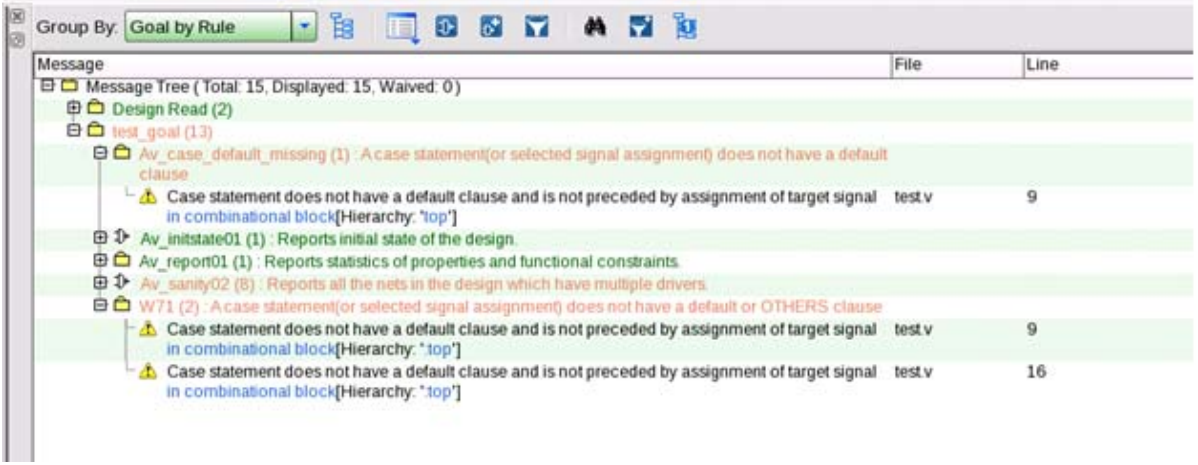


FIGURE 12. Av_case_default_missing Messages

In the above example, violations filtered by SpyGlass Functional Lint are highlighted in green. However, violations not filtered by SpyGlass Functional Lint are highlighted in red.

As shown in the above figure, SpyGlass functional lint rule, *Av_case_default_missing*, filters one false violation reported by the SpyGlass Lint rule, W71.

Default Severity Label

Warning

Rule Group

Implicit-properties

Reports and Related Files

No related reports or files.

Av_dontcare_mismatch

Use of don't-care except in case labels may lead to simulation / synthesis mismatch

When to Use

Use this rule to identify the usage of don't care character in the design.

Description

The *Av_dontcare_mismatch* rule reports violation for based numbers that contain the don't care character.

Rule Exceptions

The *Av_dontcare_mismatch* rule does not flag a violation for a parameter, generic, or constants that are assigned a don't-care value and are used only as case-label.

Also, no rule checking is done for unused macro definitions.

Language

Verilog, VHDL

Default Weight

5

Parameter(s)

None

Constraint(s)

None

Messages and Suggested Fix

The following message appears at the location where a don't care character (?) is encountered in a based number <*num*>:

[WARNING] Based number <num> contains a don't care (?) - might lead to simulation/synthesis mismatch

This violation can be formally failed, partially-proved, not analyzed, or other by the formal engine. The not-analyzed violations appear when the

width mismatch between the signed and unsigned expressions is not passed to the formal engine.

For more information on the property status reported during the functional analysis, see *SpyGlass Auto Verify Rules Reference Guide*.

Potential Issues

Violation may arise when a based number contains a `don't care` value.

Consequences of Not Fixing

There is no physical counterpart for the don't-care value. In simulation, these values are typically mapped to 'Z' which causes a tristate to be inferred. However, this behavior should be avoided as it may result in inferring spurious logic.

How to Debug and Fix

Double-click the violation message. The HDL Viewer window highlights the line where the `don't` care value, `?`, is used in the design other than in the case label.

To fix the violation, test both 0 and 1 values in a comparison. Also, in an assignment, choose either 0 or 1.

Example Code and/or Schematic

Consider the following example:

```
`define S0 2'b00
`define S1 2'b01
`define S2 2'b10
`define S3 2'b11

module top( input clk, rst,
            output reg [3:0] out0,
            output reg [2:0] out1
            );

    reg [1:0] state1;
    reg [1:0] state2;
    reg [1:0] state3;
```

```
    reg [1:0] state4;

    always @ (out0)
    begin
        out0 <= 4'b110?;
    end

    always @(posedge clk or posedge rst) begin
            if(rst) begin
                state3 <= `S0;
            end
            else begin
                priority case (state3)
                        `S0: state3 <= `S1;
                        `S1: state3 <= `S2;
                        `S2: state3 <= `S0;
                       default: begin
                    state3 <= `S3;
                    out1 = 3'b11?;
                end
                endcase
            end
    end
endmodule
```

In the above example, the *Av_dontcare_mismatch* rule reports following violation messages:

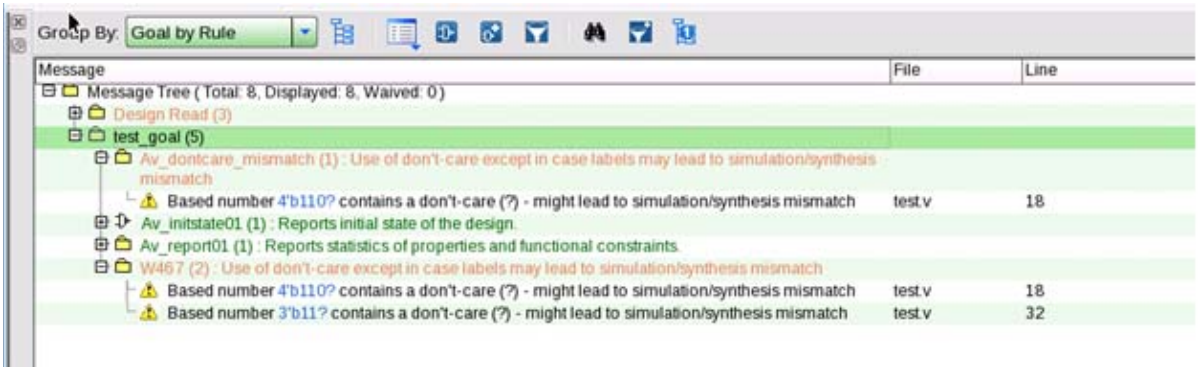


FIGURE 13. Av_dontcare_mismatch Messages

In the above example, violations filtered by SpyGlass Functional Lint are highlighted in green. However, violations not filtered by SpyGlass Functional Lint are highlighted in red.

As shown in the above figure, SpyGlass functional lint rule, *Av_dontcare_mismatch*, filters one false violation reported by the SpyGlass Lint rule, W467.

Default Severity Label

Warning

Rule Group

Expression Rules

Reports and Related Files

No related reports or files.

SpyGlass Lint Abstraction Flow

Overview

Based on the goal run while generating an abstract view of a block and validating assumptions on an abstract view, SpyGlass generates and validates product-specific details.

This section describes the different type of information generated and validated during the SoC methodology flow.

- *Generating an Abstract View in SpyGlass Lint*
- *Validating Assumptions on Abstract View in SpyGlass Lint*
- *Using the Abstract View in SpyGlass Lint*
- *Using the Automatic SoC Flow in SpyGlass Lint*

In the SpyGlass 5.6 release, the *Invoking the Automatic SoC Flow* has been enhanced.

Generating an Abstract View in SpyGlass Lint

To generate an abstract view while using SpyGlass Lint solution, run the `lint_abstract` goal.

The following figure shows the process of generating an abstract view in the SpyGlass Lint solution:

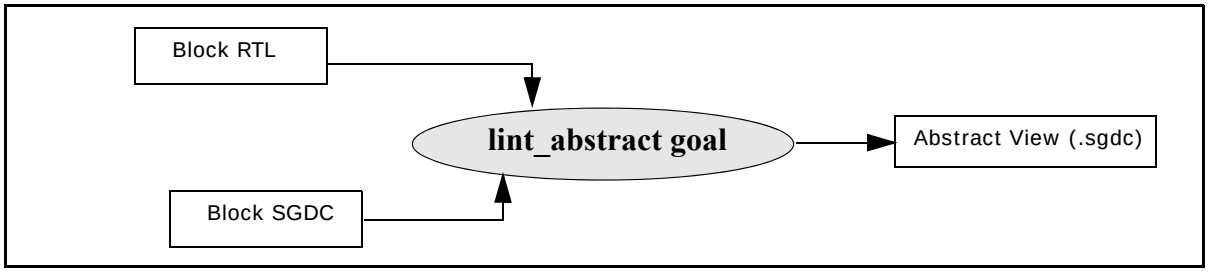


FIGURE 1.

To generate Abstract View for clock-reset integrity checks, `Advanced_CDC` license is required.

NOTE: *For generating the abstract view, you can capture the block interface information by using the include_block_interface project file option. Alternatively, you can manually specify the block interface information in the form of SGDC constraints for the abstract model.*

Example - Generating an Abstract View in SpyGlass Lint

This section describes an example of generating an abstract view of the `blockA` block in the SpyGlass Lint solution.

It covers the following topics:

- *Reading the blockA.v and blockA.sgdc Files in SpyGlass*
- *Generating an Abstract View of blockA*
- *Understanding the Generated Abstract View of blockA*

Reading the blockA.v and blockA.sgdc Files in SpyGlass

Consider the following files (RTL and SGDC for `blockA`) specified for generating an abstract view:

// Block RTL (blockA.v)

```
module blockA (output op1, op2, input in1,
               in2, in3, rst, clk, tm);
    reg ff1, ff2, ff3;
    wire d1, clkn;
    assign clkn = !clk;
    assign d1   = in1 & in2;
    always @(posedge clk or posedge rst) begin
        if (rst) begin
            ff1 <= 1'b0;
            ff2 <= 1'b0;
        end
        else begin
            ff1 <= d1;
            ff2 <= ff1;
        end
    end
     always @(posedge clkn or posedge rst) begin
        if (rst) begin
            ff3 <= 1'b0;
        end
        else begin
            ff3 <= !(tm & in2);
        end
    end
    assign op1 = ff2 & ff3;
    assign op2 = ff3 & in3;
endmodule
```

// Block SGDC (blockA.sgdc)

```
current_design blockA
set_case_analysis -name tm -value 1
```

Following is the schematic of the above design:

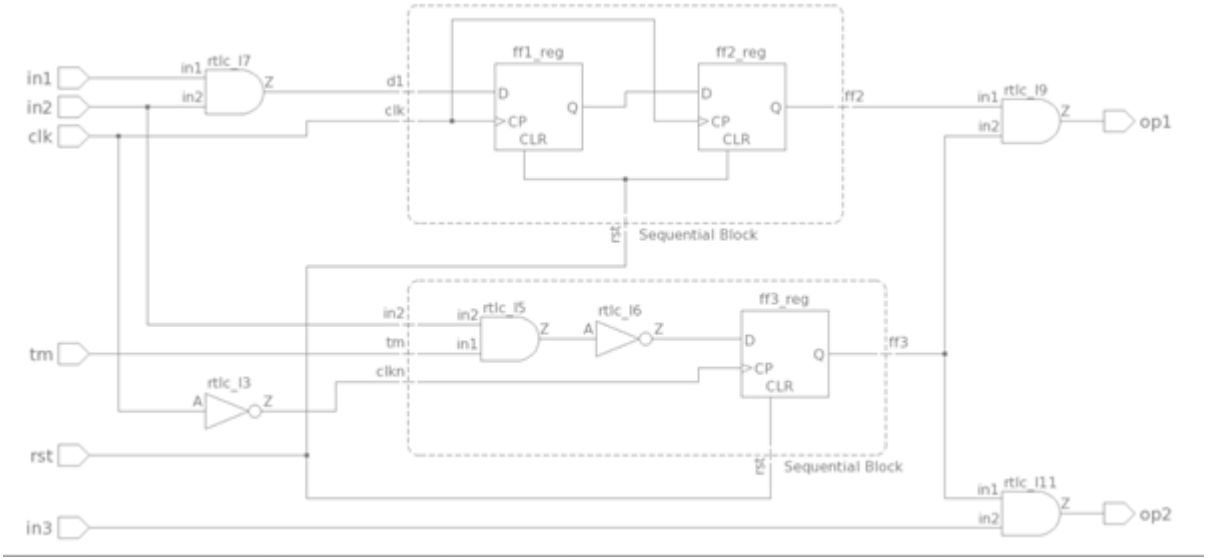


FIGURE 2.

To read in the above design and its SGDC file, use the following project file:

```
# File: blockA.prj

##Data Import Section
read_file -type verilog blockA.v
read_file -type sgdc blockA.sgdc

##Common Options Section
set_option language_mode mixed
set_option projectwdir .
set_option current_methodology $SPYGLASS_HOME/GuideWare/
2015.12/soc/rtl_handoff/
set_option top blockA
```

Generating an Abstract View of blockA

Specify the following command to generate the abstract view for the design read in the *Reading the blockA.v and blockA.sgdc Files in SpyGlass* step:

```
spyglass -project blockA.prj -batch -goal lint/lint_abstract
```

NOTE: *For generating the abstract view, you can capture the block interface information by using the include_block_interface project file option. Alternatively, you can manually specify the block interface information in the form of SGDC constraints for the abstract model.*

Understanding the Generated Abstract View of blockA

After *Reading the blockA.v and blockA.sgdc Files in SpyGlass* and *Generating an Abstract View of blockA*, the following SGDC file is generated representing the abstract view:

```
current_design "blockA" -def_param

abstract_port -ports "op1" -connected_inst "\blockA.ff2_reg
" -inst_master "RTL_FDC" -inst_pin "Q" -path_logic combo
-path_polarity buf -mode set_case_analysis -scope base
-data "REGISTERED_PORT_WITH_COMBO_CLOUD"

abstract_port -ports "op2" -connected_inst "\blockA.ff3_reg
" -inst_master "RTL_FDC" -inst_pin "Q" -path_logic combo
-path_polarity buf -mode set_case_analysis -scope base
-data "REGISTERED_PORT_WITH_COMBO_CLOUD"

abstract_port -ports "in1" -connected_inst "\blockA.ff1_reg
" -inst_master "RTL_FDC" -inst_pin "D" -path_logic combo
-path_polarity buf -mode set_case_analysis -scope base
-data "REGISTERED_PORT_WITH_COMBO_CLOUD"

abstract_port -ports "in2" -connected_inst "\blockA.ff3_reg
" -inst_master "RTL_FDC" -inst_pin "D" -path_logic combo
-path_polarity inv -mode set_case_analysis -scope base
-data "REGISTERED_PORT_WITH_COMBO_CLOUD"

abstract_port -ports "in2" -connected_inst "\blockA.ff1_reg
" -inst_master "RTL_FDC" -inst_pin "D" -path_logic combo
-path_polarity buf -mode set_case_analysis -scope base
-data "REGISTERED_PORT_WITH_COMBO_CLOUD"
```

```
abstract_port -ports "in3" -related_ports "op2" -path_logic
combo -path_polarity buf -mode set_case_analysis -scope base
-data "PORT_WITH_RELATED_PORT"

abstract_port -ports "rst" -connected_inst "\blockA.ff2_reg
" -inst_master "RTL_FDC" -inst_pin "CLR" -path_logic buf
-path_polarity buf -mode set_case_analysis -scope base
-data "REGISTERED_RST_PORT"

abstract_port -ports "clk" -connected_inst "\blockA.ff3_reg
" -inst_master "RTL_FDC" -inst_pin "CP" -path_logic inv
-path_polarity inv -mode set_case_analysis -scope base
-data "REGISTERED_CLK_PORT"

abstract_port -ports "clk" -connected_inst "\blockA.ff2_reg
" -inst_master "RTL_FDC" -inst_pin "CP" -path_logic buf
-path_polarity buf -mode set_case_analysis -scope base
-data "REGISTERED_CLK_PORT"

abstract_port -ports "tm" -scope base -data
"REGISTERED_PORT_WITH_COMBO_CLOUD"
```

In the above abstract view, the `lint_abstract` goal generates the following information:

- The `abstract_port` constraint for only one unique path of each port.

  For example, if an input pin of a block is connected to the data-pin of 10 flip-flops without any combinational or inverter logic, only one entry is generated in the abstract view (SGDC file).

  However, if there are paths that have some combinational logic or an inverter, a separate entry is generated for each for buffer, inverter, and combinational path. For example, the `in2` port reaches the `ff1_reg` and `ff3_reg` flip-flops with different polarity. Therefore, there will be two entries in the generated SGDC file.

- The `abstract_port` constraints for the `op1` and `op2` output ports with the `-path_logic combo` argument and the `-path_polarity buf` argument.

  Here, `combo` represents a combinatorial path from the source object to

the destination out-pin. Similarly, `buf` represents a buffer type as there is no path inversion between the source object and the destination out pin `Q` of the `ff2_reg` and `ff3_reg` flip-flop.

- `RTL_FDC` is the name of master module of the `ff2_reg` and `ff3_reg` flip-flops and the `-inst_pin` argument represents the pin of the master module `RTL_FDC`, which is connected to the block ports, such as `op1` and `op2`.
- Similarly, the `abstract_port` constraint is also generated for the `blockA` input port, which hits flip-flops. For example, the `in1` and `in2` ports are connected to the `ff1_reg` and `ff3_reg` flip-flops, respectively.
- If any input reaches the output port having only combination logic, SpyGlass generates the output port as `related_port` <*`output-port-name`*> in the SGDC file. For example, the `in3` port reaches the `op2` output port, which has combinational logic only.

Validating Assumptions on Abstract View in SpyGlass Lint

At this stage, read in the abstract views of lower level blocks and their port interfaces and validate them to check if top-level constraints are matching with the block-level constraints under which abstraction occurred.

The following figure shows the process of validation in SpyGlass Lint solution. If you have an enhanced abstract model, specify the `use_block_interface` project file option.

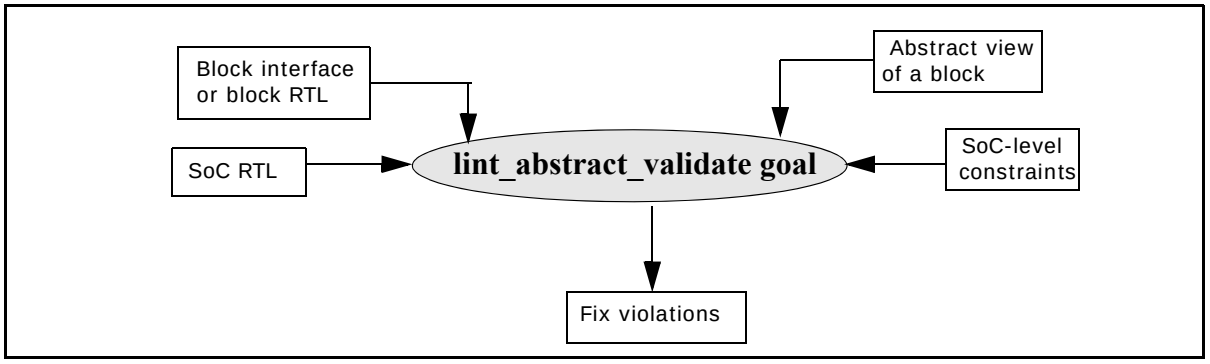


FIGURE 3.

Importing Abstract View

Create an SoC-level SGDC file to read (import) abstract view created in the *Example - Generating an Abstract View in SpyGlass Lint* step.

```
// File: top.sgdc

current_design top
sgdc -import blockA blockA/blockA_abstract.sgdc
```

NOTE: *You must also import the port interface information of the block for which you have imported the* abstract view*. The port interface information is imported in the Creating a Project File Used during Validation step.*

Creating a Project File Used during Validation

Create the following project file in which you read the abstract view of `blockA` and its port interface information in the form of its RTL:

```
# File: top.prj


##Data Import Section
read_file -type verilog top.v


## Import the 'abstract view' of the blocks
read_file -type sgdc top.sgdc
```

```
## Read the port interface of the abstract blocks
read_file -type verilog ./blockA/blockA_port_interface.v

##Common Options Section
set_option language_mode mixed
set_option projectwdir .
set_option current_methodology $SPYGLASS_HOME/GuideWare/
2015.12/soc/rtl_handoff/
set_option top top
```

Performing Validation

Run the `lint_abstract_validate` goal to validate the sanity of the abstract view with respect to the top-level test environment.

```
spyglass -project top.prj -batch -goal lint/
lint_abstract_validate
```

Viewing Validation-Related Messages

Look for the violation messages of the *LINT_sca_validation* rule in the GUI or the moresimple.rpt report.

The *LINT_sca_validation* rule reports a violation if the simulated value reaches the top-level net connected to a block-level port but no `set_case_analysis` constraint is specified on the block-level port, or vice-versa. This indicates that the abstract view is not suitable to be used at top-level.

Fix such violations, and regenerate the abstract view and validate it again.

Using the Abstract View in SpyGlass Lint

Once validation is successfully performed (block versus top-level constraints), you can perform SpyGlass Lint analysis at the top-level.

The following figure shows the process of validation in SpyGlass Lint. If you have an enhanced abstract model, specify the `use_block_interface` project file option.

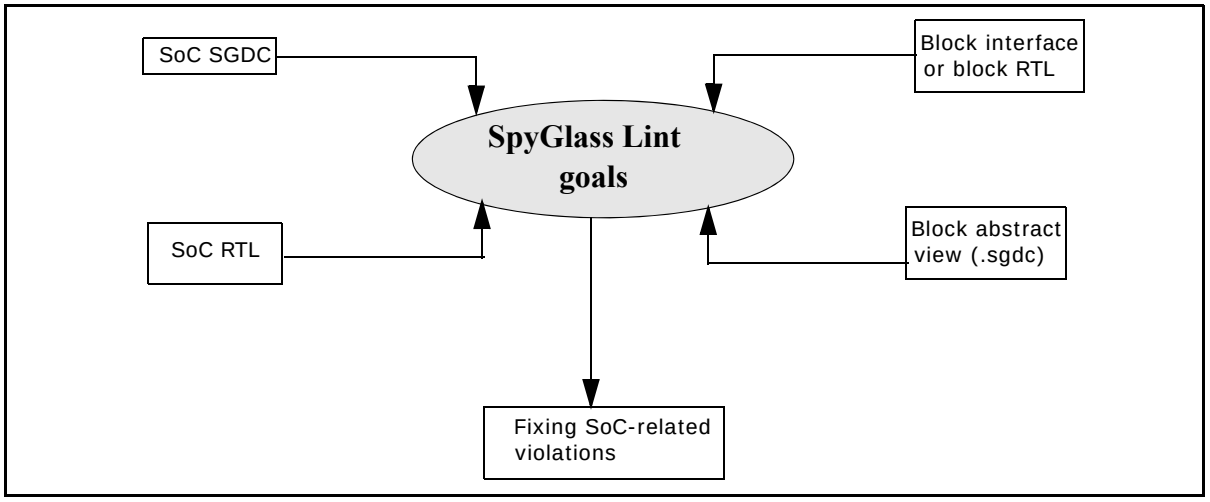


FIGURE 4.

For details on the steps during this stage, see the *Using the Abstract View during SoC-Level Verification* section in the SoC Methodology Guide.

Using the Automatic SoC Flow in SpyGlass Lint

Performing SoC verification involves multiple steps, such as verifying each block in SoC, generating an abstract view for each block, validating abstract views with SoC and performing SoC verification using the validated abstract views. This process requires complex file handling. In case of multiple instances of a block, input files for each instance should be tracked.

To simply the above process, SpyGlass Lint provides *The Automatic SoC Flow* (also known as the single push button flow). In this flow, you provide necessary inputs once and then SpyGlass automatically performs the following steps of SoC verification:

1. Generate block-level SGDC files by performing top-down constraint migration of blocks.
2. Verify each block by using the generated block-level SGDC.
3. Generate abstract views of the verified blocks.

4. Verify the generated abstract views at the SoC level.

This section covers the following topics:

- *Invoking the Automatic SoC Flow*
- *Configuration File for the Automatic SoC Flow*
- *Incremental Run of the Automatic SoC Flow*
- *Output of the Automatic SoC Flow*

Invoking the Automatic SoC Flow

Prerequisite

You must first load a design in *sg_shell* by using the `open_project` Tcl command.

Invoke the Automatic Flow

You can invoke the Automatic Flow the following ways in *sg_shell*:

- *Method 1: Configuration File*
- *Method 2: Non-Configuration File*

Method 1: Configuration File

By specifying the following command, you can provide the configuration file. In this method, the mode is `top_down` by default.

```
sg_shell> auto_soc -configfile <configuration-file>
```

For details on the `-configfile` argument of the above command, see *Configuration File for the Automatic SoC Flow*.

Method 2: Non-Configuration File

When you do not have a configuration file, by default, all the blocks instantiated directly in top are abstracted. In this method, you have the option of:

- Changing the mode to bottom up by using the `mode` argument
- Specifying a parallel file by using the `parallelfile` argument.

Example 1

By specifying the following command, you can provide the output directory and the parallel file. In this method, the mode is `top_down` by default.

```
sg_shell> auto_soc -outdir <output-directory> -parallelfile
<lsf-file> -current_product lint
```

Where, <lsf-file> should have the following format:

```
LOGIN_TYPE: lsf
MAX_PROCESSES: 2
LSF_CMD: /delsoft/software/lsf/7.0/linux2.6-glibc2.3-x86_64
/bin/bsub -I -q normal -R "rusage[mem=4000]"
```

Here, LOGIN_TYPE accepts only the lsf value.

Example 2

In this example, the mode is explicitly set to `bottom_up` and a parallel file is not provided. By default, the mode is `top_down`.

```
sg_shell> auto_soc -outdir <output-directory>
-mode bottom_up -current_product lint
```

Example 3

By specifying the following command, you can provide the output directory, the parallel file, and the mode. In this example, the mode is explicitly set to `bottom_up`. By default, the mode is `top_down`.

```
sg_shell> auto_soc -outdir <output-directory> -parallelfile
<lsf-file> -mode bottom_up -current_product lint
```

The Automatic SoC Flow

The following figure shows the automatic SoC Flow:

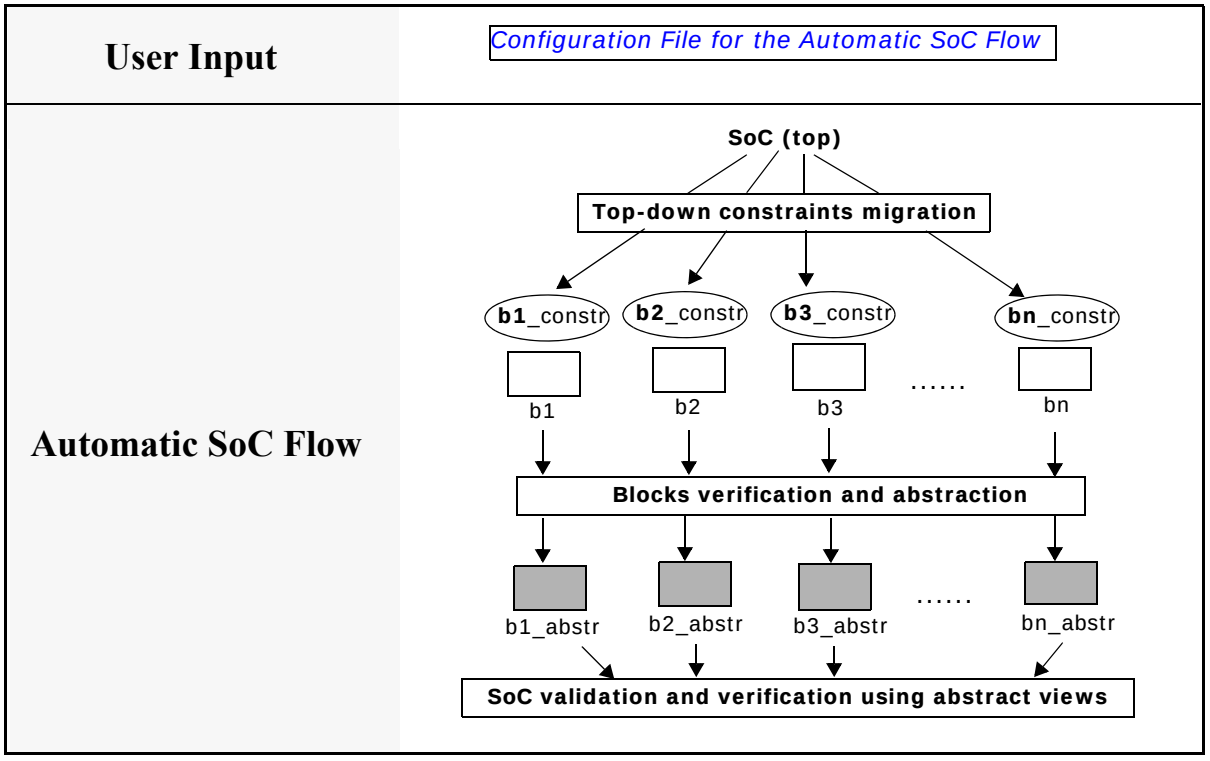


FIGURE 5. The Automatic SoC Flow

Configuration File for the Automatic SoC Flow

The configuration file provides the following information:

- Block-level and top-level goals to run
- Block-level setup information, such as parameters, input SGDC, and block abstract views
- Block-level project files
- Mode of run, that is bottom-up or top-down
- Location of output

To create a configuration file, use the `-config_file` option of the `auto_soc` Tcl command.

Sections in a Configuration File

The configuration file is divided into three top-level sections defining the following scopes:

- *Product Scope*
- *Top-Level Scope*
- *Block-Level Scopes*

Each of the above scope is defined by using a particular command. Within a scope, you can specify different specifications as described in the following table:

TABLE 1 Commands of a Configuration File

Specification	Corresponding Command
Define a verification methodology and/or goal to run.	`soc_arg current_methodology <name>` `soc_arg current_goal <goal-name>`
Define product-specific parameters and options.	`soc_arg set_parameter <name> <value>`
Define block-level project file.	`soc_arg set_project_file <prj-file>` Top-down constraints migration and project file generation is not done for the block specified with this command.
Define block-level SDC and/or SGDC file	`soc_arg sgdcfile <sgdc-file>`
Define a run mode	`soc_arg mode bottom_up` OR `soc_arg mode top_down` **(default)** If you specify *top_down*, top-level constraints are migrated to the block (top-down constraints migration). If you specify *bottom_up*, you must specify SGDC file for the block. In this case, SpyGlass performs block verification directly without performing top-down constraints migration.
Disable specific steps	• `soc_arg disable top_down` Disables the top-down constraints migration. In this case, you should specify block-level SGDC files. • `soc_arg disable verif` Disables block or top-level verification.
Define the location of the output of SoC verification	`soc_arg outdir <directory-path>`

Specification	Corresponding Command
Enable the parallel run of block-level modules	`soc_arg num_parallel_runs <number-of-block-level-modules>` For example, if you specify the value 4 to this command, *The Automatic SoC Flow* is run for any four blocks in parallel. If you do not specify this command, *The Automatic SoC Flow* of blocks occur serially.

Enable the LSF mode for parallel blocks run	`soc_arg parallelfile <lsf-file>`
	
	Where <lsf-file> is the LSF file containing the following format:
	`LOGIN_TYPE: lsf`
	`MAX_PROCESSES: 2`
	`LSF_CMD: /delsoft/software/lsf/7.0/linux2.6-glibc2.3-x86_64/bin/bsub -I -q normal -R "rusage[mem=4000]"`
	
	LOGIN_TYPE is static and it only accepts the lsf value.

See *Configuration File Format*.

Product Scope

This section defines the product under which *The Automatic SoC Flow* should be done.

Use the following command to define SpyGlass Lint scope:

```
current_product Lint
```

Use different *Commands of a Configuration File* to specify settings under this scope.

Top-Level Scope

This section defines the top-level block (SoC) for which *The Automatic SoC Flow* should be done:

Use the following command to define the top-level scope:

```
soc_top <top-name>
```

Use different *Commands of a Configuration File* to specify settings under this scope.

Block-Level Scopes

This section defines settings, such as parameters and SGDC files for a particular block. Repeat this section for each block.

Use the following command to define a block-level scope:

```
soc_block <block-name> [<inst-name>]
```

Use different *Commands of a Configuration File* to specify settings under this scope.

Configuration File Format

Following is the sample configuration file:

```
current_product Lint

soc_arg outdir run_soc_dir
#soc_arg mode bottom_up

soc_top <TOP-NAME>

soc_arg current_methodology <METHODOLOGY-NAME>

soc_arg current_goal  <GOAL-NAME>

soc_block <BLOCK1-NAME> <INSTANCE-NAME>

soc_arg current_methodology <METHODOLOGY-NAME>

soc_arg current_goal <GOAL-NAME>

soc_arg sgdcfile <block.sgdc>

soc_block <BLOCK2-NAME> <INSTANCE-NAME>

soc_arg current_methodology <METHODOLOGY-NAME>
```

```
soc_arg current_goal <GOAL-NAME>
```

Configuration File Without soc_block

If you do not specify `soc_block` in a configuration file, *The Automatic SoC Flow* performs top-down constraints migration for all the blocks at the top-level module.

For example, consider the following configuration file:

```
current_product lint
soc_arg outdir ./Work/Linux4
soc_arg parallelfile lsf
soc_top top
soc_arg current_goal lint/lint_rtl
soc_arg current_methodology $::env(SPYGLASS_HOME)
/GuideWare2.0/soc/rtl_handoff
```

In the above file, `soc_block` is missing. Therefore, all the top-level blocks are picked for top-down constraints migration. In addition, during block abstraction, all the goals available for `soc_top` are run.

Incremental Run of the Automatic SoC Flow

Incremental run of *The Automatic SoC Flow* enables you to incrementally analyze specific blocks as and when they are modified.

To incrementally analyze specific blocks, use the following command:

```
auto_soc -configfile <Tcl-file> -incremental “<space-
separated-block-list>”
```

The following example shows the usage of the above command:

```
auto_soc -configfile cfg.tcl -incremental “b1 b2 b3 b4”
```

Running *The Automatic SoC Flow* incrementally is required in the following cases:

- You make updates in the top-level RTL, project file, or SGDC file.

  Performing incremental run in this case invokes *The Automatic SoC Flow*

again including top-down constraints migration.

- You update the project file of a specific block or the RTL/SGDC of specific blocks.

Output of the Automatic SoC Flow

This section covers the following topics:

- *Directories Created in the Automatic SoC Flow*
- *Run Results in the GUI in the Automatic SoC Flow*
- *The HTML Report Generated in the Automatic SoC Flow*

Directories Created in the Automatic SoC Flow

The following table summarizes the directories created in *The Automatic SoC Flow*:

Lint/block_run	<block-name>_<instance-name>	abstract_view	Contains the abstract view of the block
		project_files	Contains project-file related information of the block
		top_down_gen_files	Contains the SGDC file generated for the block in the top-down flow

Lint/soc_run/	<top-level-module-name>	top_down	Contains files, such as project files and SGDC files used in the top-down flow
		verif	Contains top-level verification files, such as project files and SGDC files
<top>_lint_VERIF/<top>/html_reports/	dashboard.html	Shows consolidated run results of the top-down flow	
Lint/_spyglass_internal	error.log	Shows SpyGlass fatal and run errors	
Lint/_spyglass_internal	run.log	Shows information of the steps executed.	

Run Results in the GUI in the Automatic SoC Flow

SpyGlass generates the run results of different blocks in the GUI under the *Hierarchical Flow* tab that appears after *The Automatic SoC Flow* is complete.

The *Hierarchical Flow* tab is shown in the following figure:

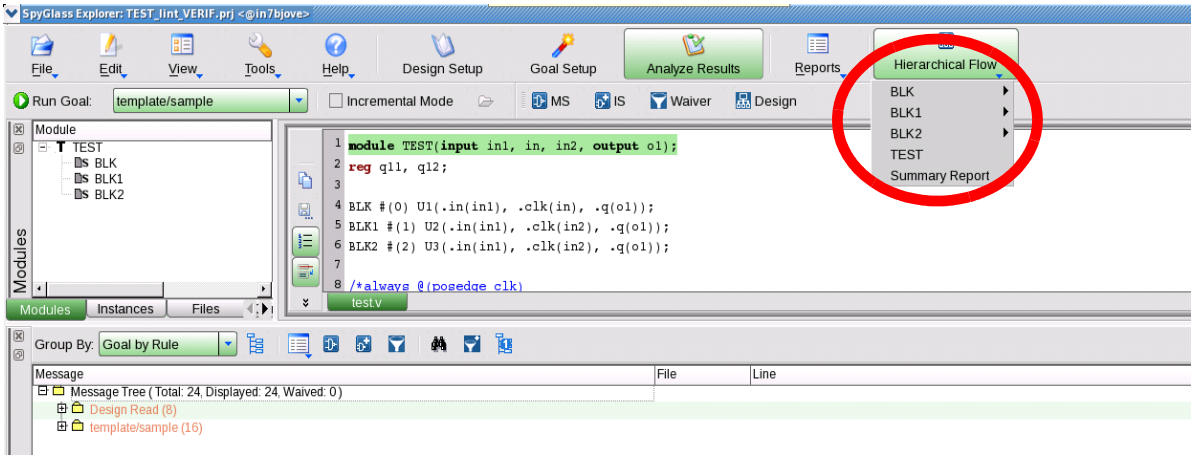


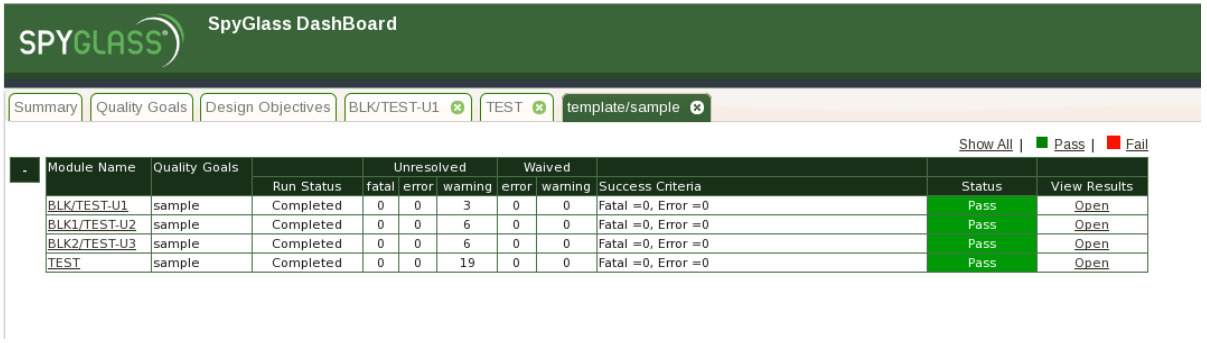
FIGURE 6. The Hierarchical Flow menu

The **Hierarchical Flow** menu enables you to load the verification run results of the top-level and the block.

The HTML Report Generated in the Automatic SoC Flow

SpyGlass generates the *dashboard.html* report in the <top>_lint_VERIF/<top>/html_reports/ directory after *The Automatic SoC Flow* is complete. This report shows the consolidated run results of the top-down flow.

The following figure shows this part of the report:



Module Name	Quality Goals		Unresolved			Waived				
		Run Status	fatal	error	warning	error	warning	Success Criteria	Status	View Results
BLK/TEST-U1	sample	Completed	0	0	3	0	0	Fatal =0, Error =0	Pass	Open
BLK1/TEST-U2	sample	Completed	0	0	6	0	0	Fatal =0, Error =0	Pass	Open
BLK2/TEST-U3	sample	Completed	0	0	6	0	0	Fatal =0, Error =0	Pass	Open
TEST	sample	Completed	0	0	19	0	0	Fatal =0, Error =0	Pass	Open

FIGURE 7. The dashboard.html Report

For more details, you can view the detailed results in the GUI. To launch the GUI, click the **Open** link in the **View Results** column. The GUI launches automatically, enabling you to use the debugging capabilities of the GUI.