cādence™

Conformal® Equivalence Checking Command Reference

Conformal L, Conformal XL, and Conformal GXL

Product Version 19.2
November 2019

Contents

About This Manual

This manual documents commands and modeling messages for the following Encounter® Conformal® Equivalence Checking solutions:

- Conformal L

  Conformal L has equivalency checking capabilities with functional checks for ASIC design flows.

- Conformal XL

  Conformal XL includes Conformal L and extends equivalency checking capabilities to datapath synthesis and layout.

- Conformal GXL

  Conformal GXL includes Conformal XL and extends equivalency checking capabilities to digital custom logic and custom memories.

- Conformal ECO

  Conformal ECO Designer offers functional ECO analysis, optimization, and generation capability.

  (Requires a Conformal ECO XL or GXL license)

- Conformal Low Power

  Conformal Low Power enables low power equivalence and functional checks for isolation cells, level-shifter cells, and state retention cells.

  (Requires a Conformal LP/LPXL or LPGXL license)

Audience

This manual is written for experienced designers of digital integrated circuits who must be familiar with RTL, synthesis, and design verification; as well as having a solid understanding of UNIX and Tcl/Tk programming.

Related Documents

For more information about the Conformal family of products, see the following documents. You can access these and other Cadence documents with the Cadence Help online documentation system. For a complete list of documents provided with this release, see the CDSDoc library.

- *User Guide for Conformal Equivalence Checking*

  Describes how to install, configure, and use Conformal to verify RTL, gate, or transistor-level designs.

Syntax Conventions

Convention	Definition
`Bold Case`	Indicates the command name.
`UPPERCASE`	Indicates the required minimum character entry.
`< >`	Indicates required arguments. Do not type the angle brackets.
`[ ]`	Indicates optional arguments. Do not type the square brackets.
`|`	Indicates a choice among alternatives. Do not type the vertical bar.
`\`	The backslash character (\) at the end of a line indicates that the command you are typing continues on the next line.
`...`	Indicates multiple entries of an argument.
`*`	Indicates that the entry can use the wildcard (*) to represent zero or more characters.

2

Command Reference

This chapter describes the Encounter® Conformal® commands. The commands are presented in alphabetical order.

This chapter also includes the following sections:

- Command Syntax
- Wildcards
- Using UNIX Commands with Conformal

Command Syntax

- Conformal commands are *not* case sensitive.
- For every Conformal `ADD` command, there are corresponding `DELETE` and `REPORT` commands. For example:

  ```
  ADD OUTPUT EQUIVALENCES
  ```

  ```
  DELETE OUTPUT EQUIVALENCES
  ```

  ```
  REPORT OUTPUT EQUIVALENCES
  ```

- Conformal commands adhere to the "3-2-1" rule, which reduces the number of characters you must type.
  - ❑ 3: Type the leading three characters of the first term.
  - ❑ 2: Then type the leading two characters of the second term.
  - ❑ 1: End with the leading character of the third term.

  In some cases, you must use more characters to resolve ambiguity. In this manual, the minimal sets of characters you must type are shown as *uppercase* letters in the syntax.

  When you use the 3-2-1 rule in conjunction with the syntax guide to resolve any possible ambiguity, you reduce the number of characters in a command, as the following example shows:

  ```
  ADD OUtput Equivalences
  ```

  becomes

  ```
  add ou e
  ```

- Reduce the number of characters you type for command options to the characters shown in uppercase in the syntax, as the following example shows:

  ```
  add output equivalences out10 out20 -module sub_mod1 -revised
  ```

  becomes

  ```
  add ou e out10 out20 -m sub_mod1 -r
  ```

Searching the Help Database for Specified Strings

Use the SEARCH command to search s the Help database of commands and options for matches to strings you specify.

Viewing TCLmode Help Information

The `HELP` command also displays a list of Conformal TCLmode commands. While in TCLmode, use the following syntax:

```
HELp
```

To view command usage for a specific command, use the `HELP` command followed by the command name.

```
HELp [command_name]
```

System prompt and command example:

```
TCL_SETUP> help set_current_module
```

Viewing Conformal UNIX-Style Man Pages

Conformal includes a `man` directory housed in: `<install_dir>/doc/mann/`.

Important

Observe the following requirements for viewing UNIX-style man pages:

You must type the entire command name.

Do not apply the 3-2-1 rule (described below).

Do replace each space in the command name with an underscore (_).

1. To access this resource from your UNIX shell, add the following variable:

   ```
   % setenv MANPATH "<install_dir>/doc:$MANPATH"
   ```

2. Type the following:

   ```
   man command_name
   ```

   For example:

   ```
   man read_design
   man set_system_mode
   ```

Wildcards

On an as-needed basis, Cadence adds wildcard pattern-matching support to Conformal commands. The syntax convention that alerts you to wildcard support is the asterisk (`*`).

If you use a pattern where a filename or design object is expected, Conformal Equivalence Checker expands the pattern using the same conventions as in the UNIX shell.

- Triggering pattern matching for filenames

  To trigger pattern matching for filenames, a string must include at least one asterisk (`*`), question mark (`?`), or a pair of square brackets (`[ ]`).

- Triggering pattern matching for design objects

  To trigger pattern matching for design objects, a string must include at least one asterisk (`*`) or question mark (`?`).

In arguments that are considered patterns, the following characters have special meaning: `^`, `{`, `}`, `[`, `]`, `?`, `*`. The dash (`-`) also has special meaning when it falls between square brackets.

Note: When you use wildcards for design objects, a wildcard can match a string that includes the hierarchical delimiter (`/`). For example, the pattern `*[10]` matches the design object `a/b/c[10]`.

When you use wildcards for filenames, every wildcard applies to part of a single directory or filename (this convention is the normal UNIX convention). For example, the pattern `*.v` does not match the filename `a/b/c.v`.

Special Characters for Filename and Design Object Pattern-Matching

Wildcard Character	Definition	Example
`?`	Match any single character.	`a?c` matches: `aac`, `abc`, `a4c`, `a?c`
`*`	Match any (possibly empty) string.	`a*c` matches the following: `ac`, `abc`, `a*c`

Special Characters for Filename and Design Object Pattern-Matching

Wildcard Character	Definition	Example

`[ ]`
 	
That is, "[" followed by characters and "]"
	Match any single character listed between the square brackets: "`[`" and "`]`".
	
	If the first character is "`^`", Conformal Equivalence Checker matches any single character not listed between the brackets.
	
	If the list shown between the brackets includes `x-y`, Conformal Equivalence Checker matches all characters in the range x–y.
	
	To match square brackets, you must include the escape character immediately preceding the square bracket.
	
	To be matched, the characters "`-`" and "`]`" must appear first in the list (possibly after `^`).
	
	Note: For design objects, recall that a string triggers pattern matching with an asterisk or question mark. In those cases, this convention applies.
		For filenames:
		
		`a[145]` matches the following:
		
		`a1`
		
		`a4`
		
		`a5`
		
		For design objects:
		
		`a*[145]` matches the following:
		
		`ab1`
		
		`a34`
		
		`at5`
		
		`a*\[145\]` matches the following:
		
		`ab[145]`
		
		`a3[145]`
		
		`at[145]`

`^`	At the beginning of the pattern, the character "`^`" negates the result of the match:	`^a*` matches any name that does not begin with `a`.

`\`	Matches only the character that follows the "`\`" character.	`a\[10]` matches the following:
		
		`a[10]`
		
		But it does not match:
		
		`a1`
		
		`a0`

Special Characters for Filename and Design Object Pattern-Matching

Wildcard Character	Definition	Example

`{p1,p2,...}`	Matches any string matched by any of the sub-patterns listed.	`design/{top,sub{5,11}}/*.v` matches the following:
		`design/top/a.v`
		`design/sub5/b.v`
		`design/sub11/c.v`
		Braces can nest.
		`a/{d{e,f},g{h,i}}_0` matches the following:
		`a/de_0`
		`a/df_0`
		`a/gh_0`
		`a/gi_0`

Using UNIX Commands with Conformal

To execute a UNIX command from within LEC or an LEC command script, start the line with an exclamation point "!" or with the `SYSTEM` command. When you execute commands in this way, they display to the standard output, and LEC records them in a log file, if one is active.

Using the -all Option

This option applies within the given defaults. For example, the syntax for the `ADD OUTPUT EQUIVALENCES` command is as follows:

```
ADD OUtput Equivalences
   <primary_pin primary_pin*...>
   [-Invert <primary_pin*...>]
   [-ROot | -Module <module_name*> | -All]
   [-Golden | -Revised | -Both]
```

In the above syntax, `-golden` is a default. Therefore, if you type the command with primary pin names and the `-all` option, but no other option, this command specifies output pin equivalences on all output boundary module pins *in the Golden design.*

Using the -both Option

When you use this option in conjunction with a specific name (for example, a pin name), that name must appear in both designs. Otherwise, LEC returns an error message. For example, the syntax for the `ADD PIN EQUIVALENCES` command is as follows:

```
ADD PIn Equivalences
   <primary_pin primary_pin*...>
   [-Invert <primary_pin*...>]
   [-ROot | -Module <module_name*> |-All]
   [-User | -Hier]
   [-Golden | -REvised | -Both]
```

Notice `-both` in the above syntax. If you specify three primary pins (for example, `a1`, `a2`, and `a`) and the `-both` option; all three pin names must exist in *both* the Golden and Revised designs. If they do not, LEC returns an error message.

If you specify `a1`, `a2`, `a` and include the `-revised` option; LEC applies equivalence to the pins *in the Revised design only* (even if these three pins also exist in the Golden design).

Saving the Command's Output to a File

To save the command's output to a file, use the command line > operator. This works for all Conformal commands.

For example, to save the default output of the `REPORT GATE` command to a file named `gate.out`, you would run the following:

```
    report gate > gate.out
```

You can also use the >> operator to append output text to an existing file.

Note: Although some commands include a `-file <filename>` type option to save the command's output to a file, you should use the command line > operator.

ABSTRACT LOGIC

```
ABSTract LOGic
     [-All]
     [-ASM | -NOASM]
     [-AUTO | -NOAUTO]
     [-REDUNDANT | -NOREDUNDANT]
     [-MODule <module_name>]
     [-NOPINDirection]
     [-NOPREResolve]
     [-POWER_VIEW]
     [-PURE]
     [-TEST_VIEW]
     [-UNRESOLVE_MOdule <module_name ...>]
     [-Golden | -Revised]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Performs functional analysis on circuit netlists, which can contain different devices, including transistors, gates, and state elements. The analysis abstracts the netlist into a logically-equivalent gate model. Use the abstracted model and compare it to the RTL model for complete functional verification. You can also write out the logic model and use it during high-performance simulation or fault grading.

Note: If neither the `-all` nor `-module` option is specified, Conformal abstracts the current root module and any modules that are instantiated under it.

Tcl Command

```
abstract_logic
```

Parameters

`-All`	Abstracts logic information from all cells in the database, including cells that are not used by the current root module.
`-ASM`	Enables the Advanced State-element Modeling (ASM) algorithm. This helps to analyze loop structure to produce better modeling of state elements, such as D-Latch, DFF, and bus-keeping I/O logic. *This is the default.*

`-NOASM`	Disables the Advanced State-element Modeling (ASM) algorithm. *Tip* If there are any unexpected results, you can use this option to revert back to the functionality of the 6.2 release and earlier.
`-AUTO`	Enables propagation of constants, pin constraints, non-inverted and inverted pin relationships across module boundaries. *This is the default.*
`-NOAUTO`	Do not enable propagation of constants, pin constraints, and non-inverted/inverted pin relationships across module boundaries.
`-REDUNDANT`	Removes three types of redundant inverters. First, remove the inverter whose fanin gate contains other inverters in the fanouts of the gate. Second, remove the inverters which sequentially connect together. Third, remove the inverters which connects to the input and output of a DFF/DLAT. *This is the default*.
`-NOREDUNDANT`	Do not enable the removal of redundant inverters.
`-MODule <module_name>`	Abstracts logic information from the specified module and its hierarchy.
`-NOPINDirection`	By default, the tool automatically assigns a pin direction (INPUT or OUTPUT) to bidirectional pins during abstraction. However, the tool cannot always determine a pin direction based on the design architecture. When this option is set, the tool will not automatically assign a pin direction to bidirectional pins during abstraction.
`-NOPREResolve`	Do not resolve any sub-modules. Be default, the tool resolves sub-modules that have a small number of devices during the abstraction process to form the circuit.
`-TEST_VIEW`	Performs structurally accurate abstraction. With this option, only limited boolean simplification is done for abstraction. As a result, the gate-level structure of the original logic is preserved as much as possible after abstraction.

`-POWER_VIEW`	Runs power-aware abstraction, where the connectivity of the power and ground pins are retained. Only limited boolean simplification is run to ensure that the abstraction results are as similar as possible to the original switch-level netlist.
`-Pure`	Performs basic gate abstraction without optimization. This is useful for circuit diagnosis.
`-UNRESOLVE_MOdule <module_name ...>`	Abstracts the logic information from specified modules first and does not flatten them to upper modules.
`-Golden`	Abstracts logic from the Golden design. *This is the default.*
`-Revised`	Abstracts logic from the Revised design.

Related Commands

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

MOS2BUFIF

READ PATTERN

REPORT ABSTRACT MODEL

REPORT CLOCK

REPORT MOS DIRECTION

REPORT NET ATTRIBUTE

ADD ALIAS

```
ADD ALias
     <aliasname> <string>
     (Setup / LEC Mode)
```

Adds alias names for quick command entry. Assign an alias to long command names and arguments to minimize typing and character entry.

If you add an alias with an alias name that already exists, Conformal accepts the new alias and returns a warning as shown in the following example:

```
// Warning: Alias 'myread' is already defined, will be replaced by the new
definition
```

For the greatest benefit, create aliases at the start of a Conformal session. Also, add aliases to an initial command file: `.conformal_lec`.

The CONFORMAL_RC Environment Variable

Conformal checks for the `CONFORMAL_RC` environment variable. If this variable is set, Conformal uses the file this variable refers to and does not search for other files.

If the `CONFORMAL_RC` variable is not set, Conformal continues the search as follows:

- First, the installation directory:

  `<install_dir>/share/cfm/lec/.conformal_lec`

- Second, the user's home directory: `~/.conformal_lec`
- Third, the current working directory: `./.conformal_lec`

If one or more of these initial command files exist, Conformal runs them in the order noted above. This process offers flexibility in the way you choose to use the initial command file. You can set up initial command files for any or all of the following purposes:

- A global initial command file for all users
- A global initial command file for an individual user
- An initial command file for a test case

Tcl Command

```
add_alias
```

Parameters

`name`	Specifies the name of the alias.
`string`	Specifies the command that the alias represents.

Related Commands

DELETE ALIAS

REPORT ALIAS

ADD BLACK BOX

```
ADD BLack Box
     [<[-Module | -Instance] [<module_or_instance_name* ...>] [-File <filename>]>
     | <-All [ | -Design | -Library]>][-Hier]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Specifies modules or instances that will be defined as blackboxes. These newly defined blackboxes are classified in the *User* class of blackboxes. Blackboxes already contained in the original design are classified in the *System* class of blackboxes.

Note: The wildcard (*) represents any zero or more characters in blackbox names.

Tcl Command

```
add_black_box
```

Parameters

`-Module`	Defines this list of module names as blackboxes. *This is the default.*
`-Instance`	Defines this list of instance names as blackboxes.
`<module_or_instance_name* ...>`	Defines the list of names of modules or instances. **Note:** Wildcard names are only supported for the module names. The wildcard (*) represents any zero or more characters in the blackbox module names.
`-File <filename>`	Specifies the name of the blackbox file. This file must contain only names of modules or instances, it is not a Verilog file. The names in the file are added to the `[name...]` list.
`-All`	Blackboxes all modules except the top module. `-All` applies within the given defaults.
`-Design`	(Used with the `-all` option only) Blackboxes all modules in the design. If you do not specify either `-design` or `-library`, blackboxing applies to both the library and the design.

`-Library`	(Used with the `-all` option only) Blackboxes all modules in the library. If you do not specify either `-design` or `-library`, blackboxing applies to both the library and the design.
`-Hier`	Used by the tool in hierarchical comparison dofiles to add hierarchically compared modules as blackboxes. *Note: This option is documented for informational purposes only and should not be used in a dofile.*
`-Golden`	Blackboxing applies to the Golden design only. *This is the default.*
`-Revised`	Blackboxing applies to the Revised design only.
`-Both`	Blackboxing applies to both the Golden and Revised designs.

Related Commands

DELETE BLACK BOX

REPORT BLACK BOX

ADD CELL PORT

```
ADD CEll POrt
     <-MODule <module_name*> ... >
     <[-POWer <port_name> ...][-GROund <port_name> ...]>
     [-GOLden | -REVised | -BOTH]
     (Setup Mode)
```

Adds definitions for power and ground ports.

Tcl Command

`add_cell_port`

Parameters

`-MODule <module_name*> ...`	Specifies the module, or modules, to which the power and ground port information applies. This accepts wildcards.
`-POWer <port_name> ...`	Specifies the names of power ports to be added.
`-GROund <port_name> ...`	Specifies the names of ground ports to be added.
`-Golden`	Adds ports to the Golden design. *This is the default.*
`-Revised`	Adds ports to the Revised design.
`-Both`	Adds ports to both the Golden and Revised designs.

Example

The following command creates a new port VDD, VDDL[0], VSS to the module LP_CELL and make VDD, VDDL[0] the power port and VSS the ground port:

```
add cell port -power VDD \VDDL[0] -ground VSS -module LP_CELL
```

ADD CLOCK

```
ADD CLock
     <0 | 1 | -STABLE_control>
     <net_name ...>
     [-Module <module_name>]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Defines a clock state where data can change. You can use this command to define:

- Pre-charge clock states for domino style circuits
- Stable nets for clock-gating modeling

Caution

When using the `ADD CLOCK` ***command with*** `set flatten model -gated_clock`***, there is an assumption that the ENABLE signal going into the AND gate of the clock cone is stable. Use with caution.***

Tcl Command

`add_clock`

Parameters

`0`	Specifies that the off-state of the clock pin is 0. This means that when the pin is low, pre-charge occurs.
`1`	Specifies that the off-state of the clock pin is 1.
`-STABLE_control`	Specifies the stable control nets for latch-free clock gate modeling.
`<net_name ...>`	Specifies the net name(s).
`-Module <module_name>`	Specifies that the defined clock pin is located in this module.
`-Golden`	Specifies that the clock is in the Golden design. *This is the default.*
`-Revised`	Specifies that the clock is in the Revised design.

`-Both`	Specifies that the clock is in both the Golden and Revised design.

Related Commands

ABSTRACT LOGIC

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

READ PATTERN

REPORT CLOCK

REPORT MOS DIRECTION

REPORT NET ATTRIBUTE

REPORT PIN DIRECTION

RESOLVE

SET ABSTRACT MODEL

ADD COMPARED POINTS

```
ADD COmpared Points
     <-All | -FILE <file_name> | <gate_id ... | <instance_pathname* ...> |
     <pin_pathname* ...>
       [-FRONTier]
       [-SUPport]
       [-Golden | -Revised]
       >
     >
     (LEC Mode)
```

Adds mapped points to the compare list. You can add compare points for all mapped points, or for a list of the gate ID numbers, instance paths, or pin paths.

If you add a compare point to the Golden design, the Conformal software also adds its mapped compare point from the Revised design. Alternately, if you add a compare point to the Revised design, the software also adds its mapped compare point in the Golden design.

Wildcard: The wildcard (*) represents any zero or more characters in instance or pin paths.

Tcl Command

`add_compared_points`

Parameters

`-All`	Adds "all" mapped points, excluding primary inputs, as compare points. `-All` applies within the given defaults.
`-FILE <file_name>`	Add compared points in batch mode from the specified file which is written by `write_compared_points`.
`<gate_id ...`	Adds the specified gate ID numbers as compare points. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`<instance_pathname*...>`	Adds the specified instance paths as compare points.
`<pin_pathname* ...>`	Adds the specified pin paths as compare points.

`-FRONTier`	Adds the specified key points and its frontier to the compare list. If no key points are specified, the frontier is computed from the existing compare points, and added to the compare list. No key points are added to the compare list if the frontier contains any unmapped key points.
`-SUPport`	Adds the specified key points and their support key points to the compare list. If no key points are specified, the support key points are computed from the existing compare points, and added to the compare list. No key points are added to the compare list if the support contains any unmapped key points.
`-Golden`	The gate ID numbers, instance paths, or pin paths are in the Golden design. *This is the default.*
`-Revised`	The gate ID numbers, instance paths, or pin paths are in the Revised design.

Examples

For a set of sample commands that shows this and related commands in context, see the example for the COMPARE command.

Related Commands

DELETE COMPARED POINTS

REPORT COMPARED POINTS

ADD CUT POINT

```
ADD CUt Point
     <pathname>
     [-Net | -Pin]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Adds a cut point to the specified net or pin path. This overrides automatic feedback loop cuts, which Conformal otherwise establishes on entering the LEC mode.

Tcl Command

```
add_cut_point
```

Parameters

`pathname`	Specifies the path that is the cut point of the feedback loop.
`-Net`	Specifies that the path is a net. *This is the default.*
`-Pin`	Specifies that the path is a pin.
`-Golden`	Applies the cut point to the Golden design. *This is the default.*
`-Revised`	Applies the cut point to the Revised design.
`-Both`	Applies the cut point to both the Golden and Revised designs.

Related Commands

DELETE CUT POINT

REPORT CUT POINT

REPORT PATH

ADD DONTTOUCH REGISTERS

```
ADD DOnttouch Registers
     <register_pathname* ...>
     [-SEQ_Constant | -SEQ_Merge]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Specified DFF/DLATs as don't-touch registers. Those don't-touch registers will not be modeled for sequential constant and/or sequential merge.

Tcl Command

`add_donttouch_registers`

Parameters

`<register_pathname* ...>`	
`-seq_constant`	Specified registers will not be modeled for sequential constant.
`-seq_merge`	Specified registers will not be modeled for sequential merge.
`-Golden`	Specified don't-touch registers to the Golden design.
`-Revised`	Specified don't-touch registers to the Revised design.
`-Both`	Specified don't-touch registers to both the Golden and Revised designs.

Related Commands

DELETE DONTTOUCH REGISTERS

REPORT DONTTOUCH REGISTERS

ADD DYNAMIC CONSTRAINTS

```
ADD DYnamic Constraints
     <0 | 1>
     <identifier ...>
     [-INStance | -Pin | -Net | -ID]
     [-Golden | -Revised | -Both]
     (LEC Mode)
```

Adds dynamic constraints for use with the `PROVE` command. Place constraints on the following:

- Hierarchical instance paths
- Hierarchical pin paths
- Hierarchical net paths
- Gate identification numbers

These constraints are either a 0-state or 1-state. Use this command as you diagnose and debug logic cones to help prove gate equivalence.

Tcl Command

```
add_dynamic_constraints
```

Parameters

`0`	Constrains the identifier to a 0-state.
`1`	Constrains the identifier to a 1-state.
`<identifier ...>`	If you do not specify one of the following options, Conformal automatically determines if the identifier is a number or a path. In the case of a number, Conformal uses the `-id` option; otherwise, Conformal searches for the gate with the `-instance`, `-pin`, or `-net` option; in this respective order.
`-INStance`	Hierarchical instance path. *This is the default.*
`-Pin`	Pin path, which is the module instance name concatenated with the pin name.

`-Net`	Net path, which is the instance name concatenated with the net name.
`-ID`	Identification number (ID) of a gate. The identification number is an integer assigned automatically by Conformal. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`-Golden`	Adds the dynamic constraints to the Golden design only. *This is the default.*
`-Revised`	Adds the dynamic constraints to the Revised design only.
`-Both`	Adds the dynamic constraints to both the Golden and Revised designs.

Examples

For a set of sample commands that shows this and related commands in context, see the example for the COMPARE command.

Related Commands

DELETE DYNAMIC CONSTRAINTS

PROVE

REPORT DYNAMIC CONSTRAINTS

ADD IGNORED GRID

```
ADD IGnored Grid
     <-Golden | -Revised>
     <-SUPply <supply_point_name> | -MODule <module_name> | -INStance
     <instance_name>>
     (Setup Mode)
```

Note: This is a Conformal Low Power command and an 1801 feature.

This command allows users to specify supply points to be ignored in power grid comparison, by either specifying a supply pin or supply port name, any supply points of a module, or any supply points of a module instance. Note that the downstream of an ignored supply point will also be ignored by such command.

Tcl Command

`add_ignored_grid`

Parameters

`-Golden`	Specifies the golden design netlist.
`-Revised`	Specifies the revised design netlist.
`-SUPply <supply_point_name>`	Specifies a supply pin in the design netlist or a virtual supply port created in the 1801 power intent.
`-MODule <module_name>`	Specifies a module name where all supply points on the instance of such module will be ignored.
`-INStance <instance_name>`	Specifies a design instance where all supply points on it will be ignored.

Related Commands

COMPARE POWER GRID

ADD IGNORED INPUTS

```
ADD IGnored Inputs
     <primary_pin* ...>
     [-ROot | -Module <module_name* ...> | -Instance <module_instance_name* ...>
     | -All]
     [-Golden | -REvised | -Both]
     (Setup Mode)
```

Specifies which input pins Conformal ignores during comparison. You can use this command when the input pins are part of a blackboxed module.

Note: Specified pins must be boundary module pins. Although boundary module pins are generally *not* compared points, they *are* compared points when the corresponding module becomes a blackbox.

For example, when Conformal compares two blackboxes and one of them has extra input pins, such as scan in and scan enable pins, use this command to tell Conformal to ignore these extra input pins during comparison.

Wildcard: The wildcard (*) represent any zero or more characters in ignored inputs and module names.

Tcl Command

```
add_ignored_inputs
```

Parameters

`<primary_pin* ...>`	Ignores this list of primary input pins (associated with the root module or the specified submodule).
`-ROot`	Ignores the specified input pins in the root module. *This is the default.*
`-Module <module_name*>`	Ignores the specified input pins in this module.
`-Instance <module_instance_name*>`	Ignores the specified input pins for the specified module instance(s).

`-All`	Ignores the specified input pins in "all" the modules, including the root module. `-All` applies within the given defaults.
`-Golden`	Ignores the specified input pins in the Golden design. *This is the default.*
`-REvised`	Ignores the specified input pins in the Revised design.
`-Both`	Ignores the specified input pins in both the Golden and Revised designs.

Related Commands

DELETE IGNORED INPUTS

REPORT IGNORED INPUTS

ADD IGNORED OUTPUTS

```
ADD IGnored Outputs
     <primary_pin*...>
     [-EQuivalences]
     [-Module  <module_name*>  | -All]
     [-Golden | -REvised | -Both]
     (Setup Mode)
```

Specifies which output pins Conformal ignores during comparison.

Note: Specified pins are boundary module pins. For example, when Conformal compares two modules and one of them has extra outputs, such as scan out pins, use this command to tell Conformal to ignore these extra output pins during comparison.

Wildcard: The wildcard (*) represents any zero or more characters in ignored outputs and module names.

Tcl Command

```
add_ignored_outputs
```

Parameters

`<primary_pin* ...>`	Ignores this list of primary output pins (associated with the root module or the specified submodule).
`-EQuivalences`	Ignores the specified output pins and their equivalences. The equivalences of a pin must be specified by the `ADD OUTPUT EQUIVALENCES` command prior to using this option. Equivalences created after using this option will not be ignored.
`-Module <module_name*>`	
	Ignores the output pins in the specified module. *The default is the root module.*
`-All`	Ignores the specified output pins in "all" the modules, including the root module. `-All` applies within the given defaults.
`-Golden`	Ignores the specified output pins in the Golden design. *This is the default.*

`-REvised`	Ignores the specified output pins in the Revised design.
`-Both`	Ignores the specified output pins in both the Golden and Revised designs.

Related Commands

ADD OUTPUT EQUIVALENCES

DELETE IGNORED OUTPUTS

REPORT IGNORED OUTPUTS

ADD INSTANCE ATTRIBUTE

```
ADD INstance Attribute
     <module_name> <instance_name>
     [<WEAK> | <DELETE>]
     [-Golden | -Revised]
     (Setup Mode)
```

Specifies how to treat an attribute for a gate or transistor primitive. Attributes can either be `WEAK` or deleted from the database for the purposes of a complete abstraction and comparison. However, newer abstraction capabilities can make the `WEAK` feature unnecessary.

Tcl Command

`add_instance_attribute`

Parameters

`<module_name>`	Applies the instance attribute to the specified module, which contains the instance.
`<instance_name>`	Applies the instance attribute to the specified instance.
`WEAK`	**Note:** This option applies to Conformal GXL. Specifies drive strength of `WEAK` on an attribute. In the case of multiple drivers, first the state of the node is determined by devices that are not WEAK (STRONG), then if there are none, or if all of the STRONG devices are disabled, the WEAK devices impact the net's function. This option affects extraction behavior and loop hand-ling.
`DELETE`	Removes the specified device from the circuit. **Note:** This option applies to Conformal GXL. *Tip* The preferred method is to use the `REMOVE` command.
`-Golden`	Applies the instance attribute to the Golden design. *This is the default.*

`-Revised`	Applies the instance attribute to the Revised design.

Related Commands

DELETE INSTANCE ATTRIBUTE

REMOVE

REPORT INSTANCE ATTRIBUTE

ADD INSTANCE CONSTRAINTS

```
ADD INstance Constraints
     <0 | 1>
     <<instance_pathname* ...> | <constraint_name> ... -Module <module_name*>>
     [-REPlace]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Places constraints on specified instance paths in either the Golden or Revised design. You can only place 0-state or 1-state constraints on the outputs of instances. You can only apply instance constraints to DFFs and D-latches inside the specified instance paths.

Wildcard: The wildcard (*) represents any zero or more characters in instance and module names.

Tcl Command

```
add_instance_constraints
```

Parameters

`0`	Constrains the specified instance paths to a 0-state.
`1`	Constrains the specified instance paths to a 1-state
`<instance_pathname* ... >`	Places the constraints on these instance paths. **Note:** The instances are either DFFs or D-latches.
`<constraint_name> ... -Module <module_name>*`	Applies the constraint to the specified module(s).
`-REPlace`	Changes the previously specified instance constraint.
`-Golden`	Applies the instance constraints to the Golden design. *This is the default.*
`-Revised`	Applies the instance constraints to the Revised design.
`-Both`	Applies the instance constraints to both the Golden and Revised designs.

Examples

The following two commands are equivalent:

```
add instance constraints 0 U1/U2/out_reg -revised
add instance constraints 0 out_reg -module U2 -revised
```

However, these will be reported differently when running the `REPORT INSTANCE CONSTRAINTS` command, because of the `-module` option.

```
// Command: report instance constraints
Constrained instances in Golden:
    0  /U1/U2/out_reg
    0  Module U2: Instance out_reg
```

Related Commands

DELETE INSTANCE CONSTRAINTS

REPORT INSTANCE CONSTRAINTS

ADD INSTANCE EQUIVALENCES

```
ADD INstance Equivalences
     <instance_pathname* ...>
     [-Invert <instance_pathname* ...>]
     [-MODULE <module_name> ]
     [-Pattern <pattern> ]
     [-HIERarchy]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Defines D-latches, DFFs or black boxes as equivalent or inverted equivalences. Use this command when you have one element in a design that corresponds to two or more elements in another design.

When used, this command includes instance equivalent results in the compare results. For example:

```
================================================================================
Compare results of instance/output/pin equivalences and/or sequential merge
================================================================================
Compared points      DFF       Total
--------------------------------------------------------------------------------
Equivalent           234       234
================================================================================
```

Effects on Comparison

This command affects comparisons when you use `add compared points -all`. In that situation, Conformal merges the instances specified with the `ADD INSTANCE EQUIVALENCES` command and then verifies them at the end of the comparison.

Wildcard: The wildcard (*) represents any zero or more characters in instance names.

Tcl Command

```
add_instance_equivalences
```

Parameters

`<instance_pathname* ...>`	
	Defines the group of instances that are equivalent. The first instance is the representative instance. The following instances are equivalent to the representative instance. This accepts wildcards. **Note:** The instances can be D-latches, DFFs or black boxes.
`-Invert <instance_pathname* ...>`	
	Defines a group of instances that is an inverted equivalence of the group that is equivalent. **Note:** The instances can be D-latches, DFFs or black boxes.
`-MODULE <module_name>`	
	Applies the instance equivalences to the specified module.The default is the root module.
`-Pattern <pattern>`	Applies the instance equivalences to registers which match the pattern. For example, if the <pattern> is "_rep\d+" LEC will merge <path>_rep1, <path>_rep2 ... to <path>
`-HIERarchy`	Applies instance equivalences to all the sub D-latches, DFFs and black boxes with corresponding sub keypoints in other instances if all the instances are the same module and not invert equivalences.
`-Golden`	Applies instance equivalences to the Golden design. *This is the default.*
`-Revised`	Applies instance equivalences to the Revised design.
`-Both`	Applies instance equivalences to both the Golden and Revised designs.

Related Commands

ADD COMPARED POINTS

DELETE INSTANCE EQUIVALENCES

REPORT INSTANCE EQUIVALENCES

SET FLATTEN MODEL

ADD INSTANCE RENAMING

```
ADD INstance Renaming
     <rule_name> <pattern> <substitution>
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Creates an instance renaming rule that will be used during the `MAP SUPPLY` command. Enables instance renaming for when the design hierarchy differs between the Golden and Revised designs. The rule will define how the full pathname of a supply keypoint in the Golden design will be renamed to find a corresponding supply key point on the Revised side.

Applies only to the Golden design. Regular expressions (such as wildcards) are not supported for *`pattern`*.

Tcl Command

```
add_instance_renaming
```

Parameters

`<rule_name>`	Specifies a unique rule name.
`<original_pattern>`	Specifies one full pathname in the Golden design that will be renamed. Regular expressions are not supported.
`<substitution>`	Specifies one full pathname in the Revised design which will replace the pathname specified by *`<original_pattern>`*

Example

The following indicates that all instances under "`a/b`" should be flattened to "`a`":

```
add instance renaming R1 "a/b" "a"
```

Related Commands

ADD MAPPING MODEL

DELETE INSTANCE RENAMING

DELETE MAPPING MODEL

MOS2BUFIF

REPORT INSTANCE RENAMING

ADD KEYPOINT INFO

```
ADD KEypoint Info
    <keyword> <value>
    <gate_id ...> | <instance_pathname* ...> | <pin_pathname* ...>
    [-Golden | -Revised]
    (Lec Mode)
```

Adds user-specified information to key points. This information is kept in the global database and can be queried using the `REPORT KEY POINT -info` command.

Tcl Command

```
add_keypoint_info
```

Parameters

`<keyword>`	Specifies the name of the property to be added to key points.
`<value>`	Specifies the value of the property to be added.
`-Golden`	Specifies that the key points are in the Golden design.
`-REvised`	Specifies that the key points are in the Revised design.

Related Commands

DELETE KEYPOINT INFO

REPORT KEY POINT

SET PROJECT

ADD LIBERTY_COMPARE FILTER

```
ADD LIberty_compare Filter
     [<filter_name>]
     [-ATTRIBUTE_NAME <name*>]
     [-DESCRIPTION <description>]
     [-GLOBAL | -WAIVE]
     [-IGNORE_PIN]
     [-LIB_CELL_NAME <name*>]
     [-LIB_PIN_NAME <name*>]
     [-OBJECT_TYPE <type>]
     [-VALUE <value*>] | [[-GOLDEN_VALUE <value*>] | [-REVISED_VALUE <value*>]]
     (Setup/LEC Mode)
```

Description

Note: This is a Conformal Low Power command.

Filters are a useful way to see only the data that you want displayed. Use this command to filter the results of the COMPARE LIBERTY command (for example, when reporting the results using the REPORT COMPARED LIBERTY command).

Note: At least one of the following options must be specified: -LIB_CELL_NAME, -LIB_PIN_NAME, -ATTRIBUTE_NAME, -VALUE, -GOLDEN_VALUE, -REVISED_VALUE.

Tcl Command

```
add_liberty_compare_filter
```

Parameters

`<filter_name>`	Specifies the name of the filter. The filter name must be specified first. If a filter name is not specified, the tool automatically generates a name.
`-ATTRIBUTE_NAME <name*>`	Filters out all differences for attributes with the specified name(s). Wildcards are supported.
`-DESCRIPTION <description>`	Specifies a description for the filter. This description is displayed when you use the REPORT LIBERTY_COMPARE FILTER command.
`-GLOBAL`	Specifies that the filter is global. *This is the default*.

`-WAIVE`	Specifies that the filter is a local filter.
`-IGNORE_PIN`	Ignores all pin differences if attribute has the value specified. The option `-attribute_name` and at least one value is required.
`-LIB_CELL_NAME <name*>`	Filter out differences for Liberty cells with the specified name(s). Wildcards are supported.
`-LIB_PIN_NAME <name*>`	Filter out differences for Liberty pins with the specified name(s). Wildcards are supported.
`-OBJECT_TYPE <type>`	Filters out differences of the specified Liberty object type. Currently Supported:
	`-LIB_CELL`: only applies to library cell
	`-LIB_PIN`: only applies to library pins (data or PG pins)
	`-LIB_PG_PIN`: only applies to PG pins (declared as pg_pin)
	`-LIB_DATA_PIN`: only applies to data pins
`-VALUE <value*>`	Filters out differences for which the compared values (Golden and Revised) match the specified value pattern. Wildcards are supported.
`-GOLDEN_VALUE <value*>`	Filters out differences for which the compared Golden value match the specified value.
`-REVISED_VALUE <value*>`	Filters out differences for which the compared Golden value match the specified value.

Example

- The following command filters out all differences related to attribute `switch_function`:

```
add liberty_compare filter -attribute_name switch_function
```

- The following command filters out all differences for attribute `related_ground_pin` where the compared value matches "`*VSS*`":

```
add liberty_compare filter -attribute_name related_ground_pin -value "*VSS*"
```

- The following command filters out all differences related to Liberty cells which match "`RAM*`":

```
add liberty_compare filter -lib_cell_name RAM*
```

- The following command filters out all differences related to Liberty pin which match "*clk*":

```
add liberty_compare filter -lib_pin_name *clk*
```

Related Commands

COMPARE LIBERTY

REPORT COMPARED LIBERTY

REPORT LIBERTY_COMPARE FILTER

ADD LOWPOWER CELLS

```
ADD LOwpower Cells
     <module_name*>
     [-isolation | -level_shifter
       | [-retention -attribute <attribute_name>] ]
     [-Both | -Golden| -Revised]
     (Setup Mode)
```

Note: This is a Conformal Low Power command.

Defines the low power attribute for specified modules.

Tcl Command

```
add_lowpower_cells
```

Parameters

`<module_name*>`	Applies the low power cell attribute to the specified module(s). Wildcards are accepted.
`-isolation`	Specifies the module as isolation cell. This is equivalent to the attribute `is_isolation_cell : true` in the liberty library.
`-level_shifter`	Specifies the module as level shifter cell. This is equivalent to the attribute `is_level_shifter : true` in the liberty library.
`-retention -attribute <attribute_name>`	Specifies the module as a state retention cell with its associated power gate cell attribute. This is equivalent to the attribute `power_gating_cell : ATTRIBUTE` in the liberty library.
`-Both`	Applies the low power cell attribute to both the Golden and Revised designs. *This is the default.*
`-Golden`	Applies the low power cell attribute to the Golden design.
`-Revised`	Applies the low power cell attribute to the Revised design.

Examples

- The following command assigns cell `fdf1a1` as a state retention cell with a `CLK_LOW` attribute:

  ```
  add lowpower cells fdf1a1 -retention -attribute CLK_LOW -revised
  ```

- The following command assigns cell `and2b1` as an isolation cell:

  ```
  add lowpower cells and2b1 -iso -revised
  ```

Related Commands

CHECK LOWPOWER CELLS

DELETE LOWPOWER CELLS

REPORT LOWPOWER CELLS

REPORT LOWPOWER DATA

SET LOWPOWER OPTION

ADD LP_CONTROL IGNORED

```
ADD LP_control Ignored
     <identifier>
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Ignores the specified low-power control signal(s) during comparison. The probe point for the ignored low-power control signal will not be created in LEC mode.

Users must enable `SET LOWPOWER OPTION -LP_CTRL_SIGNAL_COMPARED_PAIR` before `READ POWER INTENT` to use this command.

This command works after `READ POWER INTENT` for both designs.

Tcl Command

```
add_lp_control_ignored
```

Parameters

`<identifier>`	Ignores the specified low-power control signal. The identifier must be in the following formats: Isolation: <domain_name>.<strategy_name> Retention: <Save|Restore>_<domain_name>.<strategy_name> Power Switch: <control_pinname>.<strategy_name>
`-Golden`	Ignores the specified low-power control signal in the Golden design. *This is the default.*
`-Revised`	Ignores the specified low-power control signal in the Revised design.
`-Both`	Ignores the specified low-power control signals in both Golden and Revised designs.

Related Commands

SET LOWPOWER OPTION

ANALYZE LP_CONTROL PAIR

ADD LP_CONTROL PAIR

REPORT LP_CONTROL VERIFICATION

REPORT LP_CONTROL IGNORED

REPORT LP_CONTROL PAIR

ADD LP_CONTROL PAIR

```
ADD LP_control Pair
     <identifier1> <identifier2>
     [-Golden | -Revised]
     (Setup Mode)
```

Adds a compared pair of two specified low-power control signals.

Users must enable `SET LOWPOWER OPTION -LP_CTRL_SIGNAL_COMPARED_PAIR` before `READ POWER INTENT` to use this command.

This command works after `READ POWER INTENT` for both designs..

Tcl Command

```
add_lp_control_pair
```

Parameters

`<identifier1> <identifier2>`	
	Specifies two lower-power control signals as a compared pair. By default, the first identifier is from the Golden design, and the second identifier is from the Revised design. The identifier must be in the following formats: Isolation: <domain_name>.<strategy_name> Retention: <Save|Restore>_<domain_name>.<strategy_name> Power Switch: <control_pinname>.<strategy_name>
`-Golden`	When two low-power control signals are specified, indicates that both are from the Golden design. The first identifier is the representative.
`-Revised`	When two low-power control signals are specified, indicates that both are from the Revised design. The first identifier is the representative.

Examples

The following example sets the control signal of isolation strategy PD1.iso1 in the Golden design and the control signal of isolation strategy PD1.iso1 in the Revised design as a compared pair.

```
add lp_control pair PD1.iso1 PD1.iso1
```

The following example adds the control signal of PD1.iso1 and PD1.iso_dft in the Revised design as a compared pair. PD1.iso1 is the representative.

```
SETUP> add lp_control pair PD1.iso1 PD1.iso_dft -revised


Mapping result:
1-th mapped points:
G) + 15  CUT  /ISO_PD1.iso1
R) + 15  CUT  /ISO_PD1.iso1
(R) + 16  CUT  /ISO_PD1.iso_dft


Comparison result:

================================================================================
  Compared points      CUT       Total
  --------------------------------------------------------------------------------
-
  Equivalent           1         1

================================================================================
  Compare results of instance/output/pin equivalences and/or sequential merge

================================================================================
  Compared points      CUT       Total
  --------------------------------------------------------------------------------
-
  Equivalent           1         1

================================================================================
```

Related Commands

SET LOWPOWER OPTION

ANALYZE LP_CONTROL PAIR

ADD LP_CONTROL IGNORED

REPORT LP_CONTROL VERIFICATION

REPORT LP_CONTROL IGNORED

REPORT LP_CONTROL PAIR

ADD MAPPED INSTANCE

```
ADD MApped Instance
     <-Golden <instance path name>>
     <-Revised <instance path name>>
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command and an 1801 feature.

Design object names can be changed due to synthesis or implementation. This command specifies the mappings of instances from the golden design to the revised design.

Tcl Command

```
add_mapped_instance
```

Parameters

`-Golden`	Specifies the instance name in the golden netlist
`-Revised`	Specifies the instance name in the revised netlist

Examples

```
add mapped instance -golden u1/inst[0] -revised u1/inst_0_
```

ADD MAPPED POINTS

```
ADD MApped Points
     <<<gate_id | instance_pathname | pin_pathname>
       <gate_id | instance_pathname | pin_pathname>>
       |<-NET <net> <net>>
       |<-RULE <pattern> <substitution>>>
     [-INSTance | -NET]
     [-INPut_pin <Golden_pin> <Revised_pin>] ...
     [-OUTput_pin <Golden_pin> <Revised_pin>] ...
     [-NOINVert | -INVert]
     [-REPLACE_PIN_MAPPING]
     [-SKIP_AUTO_PIN_BINDING]
     [-Type <golden_type> <revised_type>]
     [-GOLden | -REVised]
     (LEC Mode)
```

When Conformal moves from Setup to LEC system mode, it automatically maps key points and places them in the *System* class of mapped points. If any additional mapped points are necessary, use this command, and Conformal will place them in the *User* class of mapped points.

Note: If you attempt to add mapped points that were already mapped, Conformal returns a warning message.

In the syntax shown below, the first `gate_id`, `instance_pathname`, or `pin_pathname` refers to the Golden design; the second argument refers to the Revised design.

The `-invert` option makes one mapped point inverted with respect to the other mapped point. The (-) sign represents an inverted-mapped point. The (+) sign represents a non-inverted mapped point.

Tcl Command

```
add_mapped_points
```

Parameters

`<gate_id>`	Adds this gate (identified by number) as a mapped point. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.

`<instance_pathname>`	Adds this instance path as a mapped point.
`<pin_pathname>`	Adds this pin path as a mapped point.
`-NET <net> <net>`	Uses the specified Golden and Revised net names to map key points together.
`-RULE <pattern> <substitution>`	Uses the instance or net renaming rule with the specified original pattern and substitution pattern to map key points together.
`-INSTance`	Specifies that the rule is an instance renaming rule. *This is the default.*
`-NET`	Specifies that the rule is a net renaming rule.
`-INPut_pin <Golden_pin> <Revised_pin> ...`	Maps the specified Golden and Revised blackboxed input pins. Multiples are permitted. However, you must list Golden and Revised pins in pairs *and* you must precede each pair with the `-input_pin` option. (See the example. For each pair that is listed, the first input pin is from the Golden design and the second input pin is from the Revised design.)
`-OUTput_pin <Golden_pin> <Revised_pin> ...`	Maps the specified Golden and Revised blackboxed output pins. Multiples are permitted. However, you must list Golden and Revised pins in pairs *and* you must precede each pair with the `-output_pin` option. (See the example. For each pair that is listed, the first output pin is from the Golden design and the second output pin is from the Revised design.)
`-NOINVert`	Do not invert the two mapped points with respect to one another. *This is the default.*
`-INVert`	Inverts the two mapped points with respect to one another.
`-REPLACE_PIN_MAPPING`	Replaces existing input and output pin mappings of blackboxes. This option is effective only when blackbox mapping is specified (through `-INPut_pin` or `-OUTpin`).
`-SKIP_AUTO_PIN_BINDING`	

	Skips automatic input and output pin binding for blackboxes. Use this option with specified input/output bin binding (through `-INPut_pin` or `-OUTpin`).
`-Type <golden_type>`	`<revised_type>`
	Specifies the types of key points that correspond to the given identifier. Where the type can be one of the following:
	`PI`: Primary inputs
	`E`: TIE-E gates
	`Z`: TIE-Z gates
	`DFF`: D flip-flops
	`DLAT`: D-latches
	`CUT`: Artificial gates inserted
	`BBOX`: Blackboxes
	`PO`: Primary Outputs
`-GOLden`	Applies this rule pattern substitution to the Golden design. *This is the default.*
`-REVised`	Applies this rule pattern substitution to the Revised design.

Examples

In the following two commands, `A1` is the instance path of a blackbox:

```
add mapped points A1 A1 -input_pin in1[0] inA[0] -input_pin in1[1] inA[1]
add mapped points A1 A1 -output_pin out1[0] outA[0] -output_pin out1[1] outA[1]
```

Related Commands

DELETE MAPPED POINTS

INVERT MAPPED POINTS

MAP KEY POINTS

READ MAPPED POINTS

REPORT MAPPED POINTS

REPORT UNMAPPED POINTS

SET MAPPING METHOD

SET NAMING RULE

ADD MAPPING MODEL

```
ADD MApping Model
     <module_name*> ...
     [-DESign]
     [-LIBrary]
     [-Inverted][-NONInverted]
     [-Golden | -Revised | -Both]
     [-Verbose]
     (Setup Mode)
```

The `ADD MAPPING MODEL` command specifies the phase information for the given modules. Use this command together with `SET MAPPING METHOD -PHASEMAPMODEL`.

Tcl Command

`add_mapping_model`

Parameters

`<module_name> ...`	Specifies the name(s) of the module(s). This accepts wildcards.
`-DESign`	Specifies that the phase information applies to the modules in the design space. When `-library` is not specified, `-design` is the default. When both `-library` and `-design` are not specified, they are both turned on by default.
`-LIBrary`	Specifies that the phase information applies to the modules in the library space. When `-design` is not specified, `-library` is the default. When both `-library` and `-design` are not specified, they are both turned on by default.
`-Inverted`	Specifies inverted phase information for the given modules. *This is the default*.
`-NONInverted`	Specifies non-inverted phase information for the given modules.
`-Golden`	Specifies that the phase information applies to the modules in the Golden design. *This is the default*.
`-Revised`	Specifies that the phase information applies to the modules in the Revised design.
`-Both`	Specifies that the phase information applies to the modules in both the Golden and Revised design.

`-Verbose`	Displays verbose information.

Examples

The following command specifies that module "`mymod`" in the library of the Golden design has an inverted phase.

```
MODE> add mapping model mymod -library
```

Related Commands

DELETE MAPPING MODEL

REPORT MAPPING MODEL

SET MAPPING METHOD

ADD MODULE ATTRIBUTE

```
ADD MOdule Attribute
     <module_name* ...>
     <-PIPELINE_Retime [-DFF2Buffer]
     | -ANALYZE_ECO_STRING <command string>
     | -COMPARE_Effort <Low | Medium | High | AUto | LIght | COMPlete | None>
     | -CPU_Limit #
     | -ECO_module
     | -FLAT_ECO_MODULE
     | -HIER_Compare <hier_compare_script>
     | -ISOlate_module <g_instance_name> <r_instance_name>
     | -NOBBOXEMpty
     | -NOFLatten
     | -NODYNAMIC_RESOLUTION>
     [-DONTTOUCH_ICG]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Defines the attributes for specified modules.

Tcl Command

`add_module_attribute`

Parameters

`<module_name* ...>`	Applies the attribute to the specified modules. This accepts wildcards.
`-PIPELINE_Retime`	Checks the specified modules for pipeline retiming (and remodel if pipeline retiming is detected). This option requires Conformal to check the module and remodel it if the Golden and Revised designs have pipeline-retiming. For advanced pipeline retiming, see ANALYZE RETIMING.
`-DFF2Buffer`	Changes registers to buffers.This option lets you compare models with no registers to those with pipeline registers inserted.

`-ANALYZE_ECO_STRING <command string>`	Replaces default `ANALYZE ECO` command in the hierarchical script generated using the `WRITE HIER_COMPAR DOFILE` command with `<command string>`. It requires with command `SET ECO OPTION -hier_flat` and the specified module has attribute by `SET ECO OPTION -flat_eco_module`.

Example:

```
add_module_attribute moduleA -
analyze_eco_string

'analyze_eco -hierarchical -ecopin_dofile
moduleA.ecopins moduleA.patch -preserve_clock -
replace'
```

The hierarchy dofile for the specified module is changed

From:

```
analyze_eco -hierarchical -ecopin_dofile
moduleA.ecopins.do moduleA.patch.v -replace
```

To :

```
analyze_eco -hierarchical -ecopin_dofile
moduleA.ecopins  moduleA.patch -preserve_clock
-replace
```

`-COMPARE_Effort`	Assigns a specified comparison effort level to the module. This option is generally applied to hierarchical comparisons where some modules need a higher compare effort than others. `Low` applies low effort to equivalency checking for the specified module. *This is the default.* `Medium` applies greater effort to equivalency checking for the specified module. `High` applies the maximum effort to equivalency checking for the specified module. `AUto` starts with low effort and automatically increases the compare effort when abort points are in the specified module. `Light` applies minimal effort to equivalency checking for the specified module. `COMplete` performs equivalency checking for each gate until the comparison results in an EQ or NEQ status. Note: The tool never aborts with this option; if the tool cannot return EQ/NEQ, the compare will continue indefinitely. `None` applies no compare effort to equivalency checking for the specified module.
`-CPU_Limit #`	Specifies a number of seconds for each module during hierarchical compare.This option decreases the amount of time Conformal spends comparing a particular module. Note that only the `COMPARE` and `ANALYZE ABORT -compare` commands are affected by the specifed CPU limit.
`-ECO_module`	Specifies ECO module(s); any mismatched ports on the specified module(s) are ignored when running the `WRITE HIER_COMPARE DOFILE` command.

`-FLAT_ECO_MODULE` Specifies flattened eco modules. With SET ECO OPTION -HIER_FLAT, replaces the default ADD COMPARED POINTS -all and COMPARE commands in the hierarchical script generated using the WRITE HIER_COMPARE DOFILE command with the following:

```
ANALYZE HIER_COMPARE -ECO_AWARE
ADD COMPARED POINTS -all
COMPARE
COMPARE ECO HIERARCHY
CHECK ECO SETUP
ANALYZE ECO -hierarchical -ecopin_dofile <mod-
ule>.ecopins.do <module>.patch.v -replace
```

`-HIER_Compare <`*`hier_compare_script`*`>`

For the specified module, replaces the default `ADD COMPARED POINTS -all` and `COMPARE` commands in the hierarchical script generated using the `WRITE HIER_COMPARE DOFILE` command with the specified `<hier_compare_script>`.

Note: This option is not supported if you want to perform hierarchical comparison using the `RUN HIER_COMPARE` command.

`-ISOlate_module <g_instance_name> <r_instance_name>`

Specifies the full path names of the Golden and Revised instances that need to be isolated. The specified module instances will then be isolated by the `ANALYZE HIER_COMPARE -isolate_module` command.

Isolating modules can help perform stand-alone analysis without reading the entire design.

Note: This option is available for beta-testing. The options may be changed prior to the final release. The name of this command is also subject to change.

`-NOBBOXEMpty` Un-blackboxes the specified empty module(s).

`-NOFLatten` Disables the specified module(s) from being flattened during the dynamic hierarchical run.

`-NODYNAMIC_RESOLUTION`

Does not perform dynamic flattening when the specified module(s) are the current root module(s).

`-DONTTOUCH_ICG` The ICG cell specified by `-donttouch_icg` would not perform gated-clock modeling.

`-Golden`	Specifies that the module attribute applies to the Golden design. *This is the default.*
`-Revised`	Specifies that the module attribute applies to the Revised design.
`-Both`	Specifies that the module attribute applies to both the Golden and Revised design.

Example

In the following command sequence, the hierarchical dofile script `hier.do` that is generated with the `write hier_compare dofile` command contains the commands `add partition points -all`, `set compare effort high`, `add compared points -all`, and `compare` for module `modA`, instead of the default `add compared points -all` and `compare` command sequence:

```
add module attribute modA -hier_compare "add partition points -all; set compare
effort high; add compared points -all; compare"
write hier_compare dofile hier.do -constraints
```

Related Commands

ANALYZE HIER_COMPARE

ANALYZE RETIMING

DELETE MODULE ATTRIBUTE

REPORT MODULE ATTRIBUTE

WRITE HIER_COMPARE DOFILE

ADD MOS DIRECTION

```
ADD MOs Direction
     <module_name> <instance_name> <net_name> |
     <-Module <module_name> | -All> [-FROM_Source_pin | -FROM_Drain_pin] >
     [-Golden | -Revised]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Adds unidirection to bidirectional MOS devices. This can change a `tranif0` into a PMOS, or change a `tranif1` to an NMOS.

Use the `REPORT MOS DIRECTION` command to display the list of all unidirectional and bidirectional transistor-MOS instances and their source and drain ports.

Use this command to resolve bidirectional transistor-MOS instances, when the `ABSTRACT` command is not able to fully resolve all bidirectional transistor-MOS instances not supported for comparison.

Tcl Command

```
add_mos_direction
```

Parameters

`<module_name>`	Adds unidirection to the specified module.
`<instance_name>`	Adds unidirection to the specified instance.
`<net_name>`	Specifies the source net name, which is the input pin of the MOS device.
`-Module <module_name>`	Adds unidirection to all bidirectional MOS devices in this module.
`-All`	Adds unidirection to all bidirectional MOS devices. `-All` applies within the given defaults.
`-FROM_Source_pin`	Specifies that all source pins are inputs. *This is the default.*
`-FROM_Drain_pin`	Specifies that all drain pins are inputs.

`-Golden`	Applies the MOS direction to the Golden design. *This is the default.*
`-Revised`	Applies the MOS direction to the Revised design.

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

READ PATTERN

REPORT CLOCK

REPORT MOS DIRECTION

REPORT NET ATTRIBUTE

REPORT PIN DIRECTION

RESOLVE

ADD NAME ALIAS

```
ADD NAme Alias
     <filename ...>
     [-GOLden|-REVised]
     [-REPlace]
     (Setup / LEC Mode)
```

Specifies a JSON data file that contains name aliases for changing keypoint names during mapping. These aliases are enabled with the "SET MAPPING METHOD -alias" command.

The JSON data file must use following format for the array of name aliases:

```
[{"m":"<module_name>","t":"ins","n":"<name>","a":"<alias>"},...]
```

The tool will change keypoint "name" in "module_name" into "alias".

For multibit cells, use the following format:

```
[{"m":"<module_name>","t":"mb_ins","n":"<name>","pin":"<pin_name>","a":"<alias>"}
,...]
```

The tool will change the name of the driver DFF/DLatch of pin "pin_name" into "alias".

Tcl Command

```
add_name_alias
```

Parameters

`<filename>`	Specifies the JOSN file that contains the name aliases.
`-GOLden`	Use the name aliases for the Golden design.
`-REVised`	Use the name aliases for the Revised design.
`-REPlace`	Clear any previous name alias data before adding new name aliases.

Related Commands

DELETE NAME ALIAS

REPORT NAME ALIAS

SET MAPPING METHOD

REPORT GATE

WRITE MAPPED POINTS

ADD NET ATTRIBUTE

```
ADD NEt Attribute
     <VDD | GND | CLOCK0 | CLOCK1 | DYNSTate>
     <net_name>
     [-Module <module_name>]
     [-Golden | -Revised]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Defines pre-charge nets; power and ground; or with `DYNSTate`, defines a dynamic latch state.

If you do not use the `-module` option with this command, Conformal applies the attribute to every module on the specified side (Golden or Revised).

Tcl Command

```
add_net_attribute
```

Parameters

`VDD`	Specifies that the net has a VDD attribute.
`GND`	Specifies that the net has a GND attribute.
`CLOCK0`	Specifies that the net has a Clock-0 attribute. This option places an off-state of 0 at the specified net. The off-state is defined as the value at which the clock port is inactive.
`CLOCK1`	Specifies that the net has a Clock-1 attribute. This option places an off-state of 1 at the specified net. The off-state is defined as the value at which the clock port is inactive.
`DYNSTate`	Specifies that the net is a dynamic state point, which Conformal GXL will abstract as a latch.
`<net_name>`	Specifies the transistor-MOS net name.

`-Module <module_name>`	Specifies that the specified module contains the transistor-MOS. If you do not use the `-module` option with this command, Conformal applies the attribute to every module on the specified side (Golden or Revised).
`-Golden`	Applies the net attribute to the Golden design. *This is the default.*
`-Revised`	Applies the net attribute to the Revised design.

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

READ PATTERN

REPORT CLOCK

REPORT MOS DIRECTION

REPORT NET ATTRIBUTE

REPORT PIN DIRECTION

RESOLVE

SET ABSTRACT MODEL

ADD NET CONSTRAINTS

```
ADD NEt Constraints
     <ONE_Hot | ONE_Cold | ZERO_ONE_Hot | ZERO_ONE_Cold>
     <net_pathname ...>
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Adds one-hot or one-cold constraints on specified net paths. The one-hot constraint lets only one net be at a 1-state and the remaining nets be at a 0-state. The one-cold constraint lets only one net be at a 0-state and the remaining nets be at a 1-state.

Tcl Command

```
add_net_constraints
```

Parameters

`ONE_Hot`	Places one-hot constraints on the specified net paths.
`ONE_Cold`	Places one-cold constraints on the specified net paths.
`ZERO_ONE_Hot`	Places zero-one-hot constraints on the specified net paths.
`ZERO_ONE_Cold`	Places zero-one-cold constraints on the specified net paths.
`<net_pathname ...>`	Places constraints on the listed net paths.
`-Golden`	Applies the net constraints to the Golden design. *This is the default.*
`-Revised`	Applies the net constraints to the Revised design.
`-Both`	Applies the net constraints to both the Golden and Revised designs.

Related Commands

DELETE NET CONSTRAINTS

REPORT NET CONSTRAINTS

ADD NOBLACK BOX

```
ADD NOblack Box
     <<module_name*> ...| -All>
     [ | -INCLUDE_SUBmodules | -SUBmodules_only]
     [-Golden | -Revised | -Both]
     [-VERbose]
     (Setup Mode)
```

Specifies the modules that are excluded from the hierarchical dofile script generation. Run this command before running the `WRITE HIER_COMPARE DOFILE` command.

Wildcard: The wildcard (*) represents any zero or more characters in module names.

Tcl Command

`add_noblack_box`

Parameters

`<module_name*>`	Do not include the listed modules in the hierarchical dofile script generation. This accepts wildcards.
`-All`	Do not include any instances or modules in the hierarchical dofile script generation *except* the top module. "All" applies within the given defaults.
`-INCLUDE_SUBmodules`	Adds the noblack box attribute to the specified module(s) and all its submodules in a recursive manner, such that they are not included in the hierarchical dofile script generation. For example, if module `mod1` instantiates another module `submod1`, which in turn instantiates another module `sub_submod1`, then this option adds the noblack box attribute to `mod1`, `submod1`, and `sub_submod1`.

`-SUBmodules_only`	Adds the noblack box attribute to the submodules of the specified module(s) in a recursive manner, such that they are not included in the hierarchical dofile script generation. For example, if module `mod1` instantiates another module `submod1`, which in turn instantiates another module `sub_submod1`, then this option adds the noblack box attribute to `submod1` and `sub_submod1`; the attribute is not added to `mod1`.
`-Golden`	Do not include the specified modules in the Golden design. *This is the default.*
`-Revised`	Do not include the specified modules in the Revised design.
`-Both`	Do not include the specified modules in either the Golden or Revised designs.
`-VERbose`	Displays any error/warning messages.

Example

For the following example, module `mod1` instantiates module `submod1`, which in turn instantiates module `sub_submod1`. The following command adds noblack box to `submod1` and `sub_submod1`:

```
add noblack box mod1 -submodules
```

Related Commands

DELETE NOBLACK BOX

REPORT NOBLACK BOX

WRITE HIER_COMPARE DOFILE

ADD NOTRANSLATE FILEPATHNAMES

```
ADD NOtranslate Filepathnames
     <filepath_names* ...>
     [ | -Library | -Design]
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Specifies library or design files, where modules defined in these files will not be translated when running the `READ LIBRARY` or `READ DESIGN` command. These modules will automatically become blackboxes.

This command is applied during initial parsing, so name matching applies only to original module names. For parameterized or VHDL generic modules whose names are determined and applied by the Conformal software after parsing and preprocessing, you must use the `ADD BLACK BOX` command.

The following are examples of modules that should not be compiled:

- RAM models
- Analog blocks
- Modules that contain non synthesizable RTL constructs

Wildcard: The wildcard (*) represents any zero or more characters in module names.

Tcl Command

```
add_notranslate_filepathnames
```

Parameters

`<filepath_names* ...>`	
	Specifies the file pathname, which can be directory names and Verilog filenames.The wildcard (*) and search path is supported.
`-Library`	Do not translate the modules in the specified library. If you do not specify `-library` or `-design`, Conformal applies this command to both the library *and* design modules.

`-Design`	Do not translate the modules in the specified design. If you do not specify `-library` or `-design`, Conformal applies this command to both the library *and* design modules.
`-Both`	Do not compile the specified modules in both the Golden and Revised libraries and/or designs. *This is the default.*
`-Golden`	Do not translate the specified modules in the Golden library or design.
`-Revised`	Do not translate the specified modules in the Revised library or design.

Related Commands

ADD NOTRANSLATE MODULES

ADD SEARCH PATH

DELETE NOTRANSLATE FILEPATHNAMES

DELETE NOTRANSLATE MODULES

REPORT NOTRANSLATE FILEPATHNAMES

REPORT NOTRANSLATE MODULES

ADD NOTRANSLATED LINES

```
ADD NOtranslated Lines
     <filename | partial_filename> [-lib libname]
     <[start:end | line] ... >
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Specifies lines to skip in a Verilog or VHDL file. This has the same effect as commenting out the lines in the file. With this command, you can instruct the Conformal software to skip certain lines in the design file without having to modify the file.

Tcl Command

`add_notranslated_lines`

Parameters

`<filename>`	Verilog or VHDL file. This option does not use the search path; the full path or relative path to the file is required.
`<partial_filename>`	Verilog file. This option searches files in the read design file list; The wildcard (*) represents any zero or more characters in partial filename.
`-lib libname`	Search filename or partial_filename in a specific library specified by `-library` option of `read design`.
`start:end`	Specifies the a line number range to skip. For example, `6:8` would skip lines 6, 7, and 8.
`line`	Specifies a line number to skip.
`-Golden`	Skips the specified lines in the Golden design. *This is the default.*
`-Revised`	Skips the specified lines in the Revised design.
`-Both`	Skips the specified lines in the Golden and Revised designs.

Example

The following commands skip lines 6 through 8 and line 17 in the `foo.v` file when reading in the design:

```
add notranslated lines foo.v 6:8 17
read design foo.v
```

As a result, lines 6, 7, 8 and 17 are skipped in the `foo.v` file:

```
(1) module test(clk, rst, set, d, z);
(2) input clk, set, rst, d;
(3) output z;
(4) reg z;
(5)
(6)    garbage 1...........
(7)    initial begin end
(8)    garbage 2...........
(9)
(10)   always @(negedge clk or negedge set or negedge rst) begin
(11)       if (!rst) z <= 1'b0;
(12)       else if (!set) z <= 1'b1;
(13)       else z <= d;
(14)   end
(15)
(17)   garbage 3....
(18)
(19) endmodule
```

Related Commands

READ DESIGN

READ LIBRARY

ADD NOTRANSLATE MODULES

```
ADD NOtranslate Modules
     <module_name* ...>
     [ | -Library | -Design]
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Specifies library or design modules that are parsed, but will not be translated when running the `READ LIBRARY` or `READ DESIGN` command. These modules automatically become blackboxes.

Note: The `ADD NOTRANSLATE MODULES` command is applied during initial parsing, so name matching applies only to original module names. For parameterized or VHDL generic modules whose names are determined and applied by Conformal after parsing and preprocessing, you must use the ADD BLACK BOX command.

Examples of modules that should not be compiled include:

- RAM models
- Analog blocks
- Modules that contain non-synthesizable RTL constructs

Wildcard: The wildcard (*) represents any zero or more characters in module names.

Tcl Command

```
add_notranslate_modules
```

Parameters

`<module_name* ...>`	Do not compile the listed library or design modules. This accepts wildcards.
`-Library`	Do not compile the specified library modules. If you do not specify `-library` or `-design`, Conformal applies this command to both the library *and* design modules.
`-Design`	Do not compile the specified design modules. If you do not specify `-library` or `-design`, Conformal applies this command to both the library *and* design modules.

`-Both`	Do not compile the specified modules in both the Golden or Revised libraries and/or designs. *This is the default.*
`-Golden`	Do not compile the specified modules in the Golden library or design.
`-Revised`	Do not compile the specified modules in the Revised library or design.

Related Commands

ADD BLACK BOX

DELETE NOTRANSLATE MODULES

READ DESIGN

READ LIBRARY

REPORT NOTRANSLATE MODULES

ADD OUTPUT EQUIVALENCES

```
ADD OUtput Equivalences
     <primary_pin> <primary_pin*> ...>
     [-Invert <primary_pin* ...>]
     [-ROot | -Module <module_name*> | -All]
     [-User | -Hier]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Specifies output pin equivalences or inverted pin equivalences on output boundary module pins. Conformal uses the equivalences when the parent module is being compared and the subsequent module is a blackbox. The first specified output pin is the representative pin. The remaining primary output pins refer to the representative pin in that added equivalence group.

Effects on Comparison

This command affects comparisons when you use `add compared points -all`. In that situation, Conformal merges the output pins specified with the `ADD OUTPUT EQUIVALENCES` command and then verifies them at the end of the comparison.

Wildcard: The wildcard (*) represents any zero or more characters in output boundary module pin and module names.

Tcl Command

`add_output_equivalences`

Parameters

`<primary_pin> <primary_pin*> ...`

Specifies a list of output boundary module pins that are equivalent. The first output pin is classified as the representative pin.

The wildcard (*) is supported for the second pin.

`-Invert <primary_pin* ...>`	Specifies a list of output boundary module pins that have inverted equivalences with respect to the representative pin. These inverted output pins are identified as (-) with the `REPORT OUTPUT EQUIVALENCES` command. This accepts wildcards.
`-ROot`	Adds output equivalences to the root module output pins. *This is the default.*
`-Module <module_name*>`	Adds output equivalences to the specified module output pins.
`-All`	Adds output equivalences to all modules within the given defaults.
`-User`	Includes the added output equivalences in the comparison when the corresponding module is compared. *This is the default.*
`-Hier`	Do not include the added output equivalences in the comparison when the corresponding module is compared.
`-Golden`	Adds output equivalences to the Golden design. *This is the default.*
`-Revised`	Adds output equivalences to the Revised design.
`-Both`	Adds output equivalences to both the Golden and Revised designs.

Related Commands

ADD COMPARED POINTS

DELETE OUTPUT EQUIVALENCES

REPORT OUTPUT EQUIVALENCES

ADD OUTPUT STUCK_AT

```
ADD OUtput Stuck_at
     <0 | 1>
     <primary_pin* ...>
     [-ROot | -Module <module_name*> | -All]
     [-User | -Hier]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Places values at the output boundary module pins. This value has no effect on the current specified module comparison. However, Conformal uses the value when it compares the parent (or higher) module.

Note: This command is limited to blackboxes. For a flat run, if the module is not blackboxed, all applied constraints will not take effect.

Wildcard: The wildcard (*) represents any zero or more characters in boundary module pin and module names.

Tcl Command

```
add_output_stuck_at
```

Parameters

`0`	Specifies that the output `stuck_at` value is 0.
`1`	Specifies that the output `stuck_at` value is 1.
`<primary_pin* ...>`	Applies the `stuck_at` value to this list of output boundary module pins. This accepts wildcards.
`-ROot`	Applies the output `stuck_at` value to the root module boundary pin. *This is the default.*
`-Module <module_name*>`	Applies the output `stuck_at` value to the specified module boundary pin. This accepts wildcards.
`-All`	Applies the output `stuck_at` value to all output boundary module pins.

`-User`	Includes the added output stuck_at constraints in the comparison when the corresponding module is compared. *This is the default.*
`-Hier`	Do not include any added output stuck_at constraints in the comparison when the corresponding module is compared.
`-Golden`	Applies the output `stuck_at` value to the Golden design. *This is the default.*
`-Revised`	Applies the output `stuck_at` value to the Revised design.
`-Both`	Applies the output `stuck_at` value to both the Golden and Revised designs.

Related Commands

DELETE OUTPUT STUCK_AT

REPORT OUTPUT STUCK_AT

ADD PARTITION KEY_POINT

```
ADD PArtition Key_point
     [-Pin | -Instance]
     <<keypoint_name ...> [-Golden |-Revised]
        | <<gname0> <rname0> <gname1> <rname1> ... -Pair>>
     [-ALL_pattern | -ONE_Cold | -ONE_Hot ]
     (Setup Mode)
```

Specifies the pin or instance names that partition the design into different compare iterations when the normal comparison process cannot finish. Execute this command before the `WRITE PARTITION DOFILE` command.

- The `-all_pattern` option creates all possible binary combinations for n key points specified.
- The `-one_hot` option creates all combinations where only one key point has a 1-state and the rest have a 0-state.
- The `-one_cold` option creates combinations where only one key point has a 0-state and the rest have a 1-state.

Important

A maximum of 14 partition points can be added. If you try to add more than 14 partition points, then only the first 14 will be taken.

Tcl Command

```
add_partition_key_point
```

Parameters

`-Pin`	Specifies that the partition key point names are pin names. *This is the default.*
`-Instance`	Specifies that the partition key point names are instance names.
`<keypoint_name ...>`	Specifies a list of partition pin or instance key point names.
`-Golden`	Specifies that the partition key points are in the Golden design. *This is the default.*

`-Revised`	Specifies that the partition key points are in the Revised design.
`<gname0> <rname0> <gname1> <rname1> ... -Pair`	Individually specifies that the Golden and Revised partition key point names when the names differ between the Golden and Revised designs.
`-ALL_pattern`	Applies all possible combinations of constraints to the partition key point names. *This is the default.*
`-ONE_Cold`	Applies a one-cold constraint to the partition key point names.
`-ONE_Hot`	Applies a one-hot constraint to the partition key point names.

Related Commands

DELETE PARTITION KEY_POINT

REPORT PARTITION KEY_POINT

WRITE PARTITION DOFILE

ADD PARTITION POINTS

```
ADD PARtition Points
     < [-Net | -Pin | -GAte_instance | -ID]  identifier... [-Pair]
       | -MODule <module_name* ...>
       | -INstance <instance_pathname* ...>
       | -Datapath | -OUTPUT_TRIstate
       | -ALL
     >
     [-ABORT_cone [compare point ...] ]
     [-Cut | -NOCut]
     [-EFFORT <Low | Medium | High>]
     [-NAME]
     [-OUTPUT_port | -INPUT_port | -INTRA_operator]
     [-Golden | -Revised]
     [-Verbose]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Adds partition (cut) points to the design. If a location in one design is specified, the Conformal software will attempt to find the corresponding point in the other design. If found, it adds a cut point to both the Golden and Revised designs. These cut points are automatically mapped. This command automatically handles inverted points.

In the cases where the added cut points are not equivalent, you will need to diagnose them. If the nonequivalency is caused by the cut point, then you will need to delete that pair of cut points with the `DELETE PARTITION POINTS` command.

The `ADD PARTITION POINTS` command appends the existing compare list with the newly added partition points. In addition, it also performs comparison on some of the compare points to ensure that the newly added partition points do not cause any false nonequivalence.

Note: If a cut point has already been added to a location, a second cut point cannot be added to the same location. If a specified point has more than one corresponding point in the design, the software will not add a cut point.

Caution

Adding cut points in LEC mode causes flattened netlists to change. As a result, all the gate IDs are subjected to change. Adding cut points does not affect the existing compare points list; however, all the compare data is invalidated after adding cut points.

Tcl Command

`add_partition_points`

Parameters

`-Net`	Specifies that the identifier is a net pathname.
`-Pin`	Specifies that the identifier is a pin pathname.
`-GAte_instance`	Specifies that the identifier is a gate-instance pathname.
`-ID`	Specifies that the identifier is a gate identification number. This number is an integer assigned automatically by Conformal. ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`identifier`	Specifies the identifier for adding the partition point. If you do not specify `-Net`, `-Pin`, `-GAte_instance`, or `-ID`, the Conformal software automatically determines if the identifier is a gate-identification number, pin pathname, net pathname, or a gate-instance pathname.
`-Pair`	Allows specification of multiple pairs of identifiers, where the first identifier in each pair refers to the Golden design and the second identifier in each pair refers to the Revised design.
`-MODule <module_name* ...>`	Adds partition points to all the instances of the specified module(s).
`-INstance <instance_pathname* ...>`	Specifies the module instance pathname. By default, the partition points are all the output pins of the specified instance. **Note:** If an instance is removed during the Conformal flattening and modeling process, you cannot add partition points to the instance.

`-Datapath`	Adds partition points to all module instances containing datapath operators. The datapath operator is the multiplier (including the merged operator) in the Golden design. By default, the partition points are all the output pins of the datapath operator(s).
`-OUTPUT_TRIstate`	Specifies the partition points as the tri-state buffer at the primary output.
`-ALL`	Adds partition points to all possible locations. This is useful when resolving aborts.
`-ABORT_cone [compare point ...]`	Adds the specified partition points only in the fan-in logic cone of the specified compare points. If no compare point is specified, by default, the partition points are added in the fanin logic cone of all the compare points with abort or unknown results. **Note:** You must add compare points first to use this option.
`-CUT`	Generates the cut gates in the flattened netlists for the feasible partition points in Golden and Revised design. *This is the default.*
`-NOCUT`	Do not generate any cut gates in flattened netlists.
`-EFFORT <Low | Medium | High>`	Specifies the effort level for adding partition points. Increasing the effort level from `Low` to `High` will improve the quality of partition points being added, but will result in increased time. The default effort level is `Medium`.

`-NAME`	Adds partition points around instances using a name-based algorithm. This is faster than the default algorithm which adds partition points based on function.
	
	To perform a flat run with hierarchical partitioning, add partition points to all module instances using the following command:
	
	`add partition points -instance * -name -input -output`
	
	To get better datapath quality, add partition points to all module instances containing datapath operators using the following command:
	
	`add partition points -datapath -name -input -output`

`-OUTPUT_port`	Specifies the partition points as all the output pins of the module instance or datapath operator. *This is the default*.
`-INPUT_port`	Specifies the partition points as all the input pins of the module instance or datapath operator.
`-INTRA_operator`	Specifies the partition points as the gates inside the datapath operator.
`-Golden`	Specifies that the partition points are from the Golden design. *This is the default.*
`-Revised`	Specifies that the partition points are from the Revised design.
`-Verbose`	Reports the partition points with correspondences in the other design.

Examples

- The following command specifies partition points in Golden and Revised design by pin pathnames:

```
add partition points -pin i0_gold/sum[17]   i0_rev/z[17] -pair
```

- The following command specifies the partition points in the Golden design. The command automatically finds the corresponding gates in the Revised design, and if found, physically adds the cut gate pair in the flattened netlist.

```
add partition points -pin i0/sum[17] i0/sum[16] -golden
```

- The following command specifies the partition points from the output pins of multiplier operators in the Golden design. The command automatically finds the correspondence gates in the Revised design, and adds partition point pairs as cut gates in the flattened netlists.

```
add partition points -datapath
```

- The following command specifies an instance of datapath operator in the Golden design and adds some gates inside the operator as partition points:

```
add partition points -instance i0 -intra_operator
```

- The following command specifies all the datapath operators in the Golden design and adds some gates inside the operators as partition points:

```
add partition points -datapath -intra_operator
```

- The following command adds partition points for all the tristate buffers at the primary outputs:

```
add partition points -output_tristate
```

- The following command adds partition points to all instances of the `DW_` module:

```
add partition points -module DW_*
```

- The following command adds partition points at all the possible locations in the fan-in logic cone of the compare points with abort or unknown compare results:

```
add partition points -all -abort_cone
```

Related Commands

ADD COMPARED POINTS

DELETE PARTITION POINTS

REPORT PARTITION POINTS

ADD PHYSICAL CELLS

```
ADD PHysical Cells
     <module* ...>
     [-BOTh | -GOLden | -REVised ]
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Manually adds physical cells to the `physical_cell_list` created by the `EXTRACT PHYSICAL CELL` command.

Tcl Command

```
add_physical_cells
```

Parameters

`<module_name* ...>`	
	Adds the specified module(s) to the `physical_cell_list`, which is created by the `EXTRACT PHYSICAL CELL` command. Simple wildcard matching is supported.
`-BOTh`	Looks for the module(s) in both the Golden and Revised designs. *This is the default.*
`-GOLden`	Add physical cells in the Golden design.
`-REVised`	Add physical cells in the Revised design.

Related Commands

DELETE PHYSICAL CELLS

EXTRACT PHYSICAL CELLS

REPORT PHYSICAL CELLS

ADD PIN BINDING

```
ADD PIn Binding
     <golden_pin> <revised_pin>
     [-MOdule <golden_module_name> <revised_module_name>]
     [-INvert]
     (Setup Mode)
```

`ADD PIN BINDING` specifies a pair of pin names for:

- ❑ Primary inputs and outputs mapping while the module is a root module
- ❑ Inputs and outputs mapping of a black box

The pin binding process is in the mapping process which occurs automatically during the system mode switch from Setup to LEC.

Tcl Command

`add_pin_binding`

Parameters

`-MOdule`	Specify that the pin binding rules apply to the specified module pairs. Default module pair is the golden and revised top modules.
`-INvert`	Indicates the phases of the two pins are inverted..

Related Commands

DELETE PIN BINDING

REPORT PIN BINDING

ADD PIN CONSTRAINTS

```
ADD PIn Constraints
     <0 | 1 | ONE_Hot | ONE_Cold | ZERO_ONE_Hot | ZERO_ONE_Cold>
     <primary_pin* ...>
     [-REPlace] [-ROot | -Module <module_name*> | -All]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Constrains primary input pins to a logic value or relationship. The supported constraints are:

- 0-state
- 1-state
- One-hot
- One-cold
- Zero-one-hot
- Zero-one-cold
- Base 2, 8, 10, and 16 Verilog constants

The one-hot constraint lets only one pin be at a 1-state and the remaining pins be at a 0-state. The one-cold constraint lets only one pin be at a 0-state and the remaining pins be at a 1-state.

Wildcard: The wildcard (*) represents any zero or more characters in primary input and module names.

Tcl Command

```
add_pin_constraints
```

Parameters

`0`	Constrains pins to a constant 0 value.
`1`	Constrains pins to a constant 1 value.
`ONE_Hot`	Specifies that only one of the pins can have a high value.

`ONE_Cold`	Specifies that only one of the pins can have a low value.
`ZERO_ONE_Hot`	Specifies that all pins can have a low value, but only one can have a high value.
`ZERO_ONE_Cold`	Specifies that all pins can have a high value, but only one can have a low value.
`<primary_pin*...>`	Specifies that a list of primary input names that will be constrained to a certain state. (These primary inputs are from either the Golden or Revised design.) This accepts wildcards.
`-REPlace`	Changes the previously specified pin constraint.
`-ROot`	Applies the constraints to the root module(s). *This is the default.*
`-Module <module_name*>`	Applies the constraints to the specified module(s). The constraints are applied when the specified module is the root module or when it is blackboxed. This accepts wildcards. Note: In a flat comparison, to specify a constraint for a submodule, you must use the ADD PRIMARY INPUT command and specify a pin or net pathname. This cuts the existing driver of the net/pin and forces a new value to be propagated. If the submodule net/pin is floating, use the ADD TIED SIGNALS command instead.
`-All`	Applies to all root or blackboxed modules (does not apply to sub-modules during a flat comparison).
`-Golden`	Specifies that the primary input names are from the Golden design. *This is the default.*
`-Revised`	Specifies that the primary input names are from the Revised design.
`-Both`	Specifies that the primary input names are from both the Golden and Revised designs.

Related Commands

ADD PRIMARY INPUT

ADD TIED SIGNALS

DELETE PIN CONSTRAINTS

REPORT PIN CONSTRAINTS

ADD PIN EQUIVALENCES

```
ADD PIn Equivalences
     <primary_pin> <primary_pin* ...>
     [-INPUT_OUTPUT]
     [-Invert <primary_pin* ...>]
     [-ROot | -Module <module_name*> | -All]
     [-User | -Hier]
     [-Golden | -REvised | -Both]
     (Setup Mode)
```

Defines the relationship between pins; specifies whether module pins are equal or inverted. Use this command to complete logic abstraction or to resolve differences in logic during comparisons.

Note: Pin equivalences are considered only when the pin equivalences are on a root module. If a submodule has black-box pin equivalences, the Conformal Equivalence Checker checks to see if they are true when you use the `ADD COMPARE POINT -all` command.

Wildcard: The wildcard (*) represents any zero or more characters in primary input and module names. Wildcards are not valid for buses.

Tcl Command

`add_pin_equivalences`

Parameters

`<primary_pin> <primary_pin* ...>`	
	Specifies a list of primary input pins that are equivalent. The first input pin is classified as the representative pin. The wildcard (*) is supported for the second pin.
`-INPUT_OUTPUT`	Specifies blackbox module pin equivalences for either of the following conditions:

1. Feedthrough equivalence, where the input port is the representative.

2. Feedback equivalence, where the output port is the representative.

For example:

```
add pin equivalence input_pin -INPUT_OUTPUT
output_pin -module mod_A -revised
```

`-Invert <primary_pin* ...>`

Specifies that the primary pins are inverted with respect to the referenced primary pin. Use the `REPORT PIN EQUIVALENCES` command to identify the inverted primary input pins. The (-) denotes inverted pins. This accepts wildcards.

`-ROot` Applies the pin equivalences to the root module. *This is the default.*

`-Module <module_name*>`

Applies the pin equivalences to the specified module. *The default is the root module.*

You can also use this command on buses, by defining the bus range using `<module_name[msb:lsb]>`.

`-All` Applies the pin equivalences to "all" the modules. `-All` applies within the given defaults.

`-User` When you use this option, Conformal includes these added pin equivalences in the comparison, when the corresponding module is a blackbox. (This option does not apply to primary inputs.) *This is the default.*

`-Hier` When you use this option, Conformal does not do the additional check associated with the `-user` option.

`-Golden` Specifies that the pin equivalences are from the Golden design. *This is the default.*

`-Revised` Specifies that the pin equivalences are from the Revised design.

`-Both`	Specifies that the pins specified as equivalent exist in both the Golden and Revised designs. **Note:** If you use the `-both` option, every primary input pin you list must exist in *both* designs (Golden and Revised). Otherwise, Conformal returns an error message.

Example

The following example implies that `p1`, `p2`, and `p3` are equivalent and are inverted to `p4`, `p5`, and `p6`:

```
add pin equivalences p1 p2 p3 -inv p4 p5 p6
```

Related Commands

DELETE PIN EQUIVALENCES

REPORT PIN EQUIVALENCES

ADD POWER_INTENT_COMPARE FILTER

```
ADD POwer_intent_compare Filter
     [<filter_name>]
     [-DESCRIPTION <description>]
     [-DESIGN_OBJECT_FILTER <filter_expression>]
     [-FILTER <filter_expression>]
     [-REGEXp]
     [[-VALUE <value*>]| [[-GOLDEN_VALUE <value*>] | [-REVISED_VALUE <value*>]]]
     | <<[-OBJECT_TYPE <type>]
     |   [-COMMAND_NAME <name>][-OBJECT_NAME <name*>][-OPTION_NAME <name*>]>
     | < [-ATTRIBUTE_NAME <name*>] [-OBJECT_NAME <name*>] >>
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Filters are a useful way to see only the data that you want displayed. Use this command to filter the results of the `COMPARE POWER INTENT` command (for example, when reporting the results using the `REPORT COMPARED INTENT` command).

Tcl Command

```
add_power_intent_compare_filter
```

Parameters

`filter_name`	Specifies the name of the filter. The filter name must be specified first. If a filter name is not specified, the tool automatically generates a name.
`-DESCRIPTION <`*`description`*`>`	Specifies a description for the filter. This description is displayed when you use the `REPORT POWER_INTENT_COMPARE FILTER` command.

`-DESIGN_OBJECT_FILTER <filter_expression>`

Filters only design object compared points for which the specified `filter_expression` evaluates to true. The syntax for `filter_expression` is the same as for the Tcl find command. Use command '`report compared intent -attributes`' see the available attributes. See Example section for the example.

`-FILTER <filter_expression>`	Filters only compared points for which the specified `filter_expression` evaluates to true. The syntax for `filter_expression` is the same as for the Tcl `find` command. Use command '`report compared intent -attributes`' see the available attributes. See Example section for the example.
`-REGEXp`	Specifies that the pattern matching operators used inside filter expression (such as =~) should use regular expressions as patterns, rather than glob-style patterns. This option only has any effect if combined with `-FILTER` and/or `-DESIGN_OBJECT_FILTER`.
`-VALUE <value*>`	Filters out differences for which the compared value matches the specified value pattern. Wildcards are supported.
`-GOLDEN_VALUE <value*>`	Filters out differences for which the compared Golden value match the specified value.
`-REVISED_VALUE <value*>`	Filters out differences for which the compared Revised value match the specified value.
`-OBJECT_TYPE <name*>`	Filter out differences for the specified power intent object type. For a list of supported options, see the section Power Intent Object Types in the `REPORT POWER INTENT` command page.
`-COMMAND_NAME <name>`	Filter out differences for power intent objects related to the specified command.
`-OBJECT_NAME <name*>`	Filter out differences for power intent objects with the specified name(s). Wildcards are supported.
`-OPTION_NAME <name*>`	Filter out differences for options with the specified name(s). Wildcards are supported.
`-ATTRIBUTE_NAME <name*>`	Filters out all differences for attributes with the specified name(s). Wildcards are supported.

Example

- The following command filters out all differences related to power switches:

```
add power_intent_compare filter -object_type power_switch
```

- The following command filters out all differences for port states where the compared value matches "`*onH 1.2*`":

  ```
  add power_intent_compare filter -object_type port_state -value "*onH 1.2*"
  ```

- The following command filters out all differences related to option `-applies_to` of isolation strategies:

  ```
  add power_intent_compare filter -object_type isolation -option_name \
    -applies_to
  ```

- The following command filters out all differences related to attribute "`UPF_related_ground_port`":

  ```
  add power_intent_compare filter -attribute UPF_related_ground_port
  ```

- The following filters all failed isolation design object compared points that are named `'inst2/count[2]'`

  ```
  add power_intent_compare filter -object_type isolation \
    -design_object_filter "object_name==inst2/count[2]"
  ```

- The following filters all failed isolation design object compared points that are inside instance '`inst2`':

  ```
  add power_intent_compare filter -object_type isolation \
    -design_object_filter "object_name==~inst2/*" -regexp
  ```

- The following filters the `connect_supply_net` compared points that are connected to design macro instances '`macro1`':

  ```
  add power_intent_compare filter -command_name connect_supply_net \
    -design_object filter "object_name = ~macro1/*"

  or

  add power_intent_compare filter -object_type connect_supply_net \
    -design_object filter "object_name = ~macro1/*"
  ```

Note: To view what available attribute name of `connect_supply_net`, use `'report compared intent -connect_supply_net -attributes'`

- The following filters the `set_isolation` compared points that is named as `PD2.iso_rule2` power intent object at the golden side:

  ```
  add power_intent_compare filter -command_name set isolation\
    -filter "golden_command_name == PD2.iso_rule2"

  or

  add power_intent_compare filter -object_type set_isolation \
    -filter "golden_command_name == PD2.iso_rule2"
  ```

Note: To view what available attribute name of `set_isolation`, use `'REPORT COMPARED INTENT -set_isolation -attributes'`

Related Commands

COMPARE POWER INTENT

DELETE POWER_INTENT_COMPARE FILTER

REPORT COMPARED INTENT

REPORT POWER INTENT

ADD PRIMARY INPUT

```
ADD PRimary Input
     <pathname* ...> [-Net] | <pathname* ...> -Pin>
     [-Cut | -NOCut]
     [-LIBRARY][-NOLIBRARY]
     [-MODULE <module_name* ...>]
     [-SET_TCL_VARIABLE <variable_name>]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Adds a new primary input pin to a specified net or pin name. This new primary input pin is classified in the *User* class of primary inputs. (The original primary inputs of the design are classified in the *System* class of primary inputs.) Use the `-cut` option if the added primary input is the only driver to the net or pin. Otherwise, along with the other original drivers, the net becomes a wired net.

Wildcard: The wildcard (*) represents any zero or more characters in net paths.

Tcl Command

```
add_primary_input
```

Parameters

`<pathname*> -Net`	Adds the primary input to the specified net path. `-Net` *is the default.*
`<pathname*> -Pin`	Adds the primary input to the specified pin path.
`-Cut`	Cuts the other original drivers of the specified path and allow only the newly added primary input as the driver of the net or pin. *This is the default.*
`-NOCut`	Do not cut the other original drivers of the specified net or pin name. This option makes the new net a wired net.
`-LIBRARY`	Specifies that the specified modules names are in the library space. The default is in both library and design spaces.
`-NOLIBRARY`	Specifies that the specified modules names are in the design space. The default is in both library and design spaces.

`-MODULE <module_name* ...>`

	Adds primary inputs to the specified module(s)
`-SET_TCL_VARIABLE <variable_name>`	
	Places all the newly added primary inputs into the specified Tcl variable as a list. This list can be directly used in the `ADD PIN CONSTRAINTS` command.
`-Golden`	Adds the new primary input in the Golden design. *This is the default.*
`-Revised`	Adds the new primary input in the Revised design.
`-Both`	Adds the new primary input in both the Golden and Revised designs.

Example

For example, the following adds primary inputs to pin 'SI' of modules whose name contains string "FSD", but in the library space only. Then, it adds pin constraint 1 on all these newly added inputs.

```
add primary input SI -module *FSD* -library -both -set_tcl_variable
tclmode
add_pin_constraint 1 $pinlst -both
```

Related Commands

DELETE PRIMARY INPUTS

REPORT PRIMARY INPUTS

ADD PRIMARY OUTPUT

```
ADD PRimary Output
    <net_pathname*>
    [-Golden | -Revised | -Both]
    (Setup Mode)
```

Adds a new primary output pin to a specified net name. This new primary output pin is classified in the *User* class of primary outputs. (The original primary outputs of the design are classified in the *System* class of primary outputs.) This command is used for diagnosis when an internal value can be observed at a primary output.

Wildcard: The wildcard (*) represents any zero or more characters in net paths.

Tcl Command

```
add_primary_output
```

Parameters

`<net_pathname*>`	Adds the primary output to the specified net path. This accepts wildcards.
`-Golden`	Adds the new primary output in the Golden design. *This is the default.*
`-Revised`	Adds the new primary output in the Revised design.
`-Both`	Adds the new primary output in both the Golden and Revised designs.

Related Commands

DELETE PRIMARY OUTPUTS

REPORT PRIMARY OUTPUTS

ADD PROBE POINT

```
ADD PRobe Point
     <pathname> <probe_name> [-Net] | <pathname> <probe_name> -Pin
     [-Golden | -Revised | -Both]
     [-Strict | -NOStrict]
     [-Verbose]
     (Setup Mode)
```

Adds a probe point to the specified path of a net or a pin of a module instance. The probe point is named by the user-specified name. The probe point is an observation point to check the logic cone driving the specified path. Exactly name-matched probe points between designs will be automatically mapped when mapping. A probe point is a subset of CUT point that does not involve actually cutting the net. Users can compare and diagnose the mapped probe points. Note that probe points cannot be added within a blackbox.

Tcl Command

`add_probe_point`

Parameters

`<pathname> <probe_name> -Net`	Adds a probe point with the specified name at the specified net pathname. *This is the default*.
`<pathname> <probe_name> -Pin`	Adds a probe point with the specified name at the specified pin pathname. Note that only the pin pathname of a module instance can be specified.
`-Golden`	Adds the probe point in the Golden design. *This is the default*.
`-Revised`	Adds the probe point in the Revised design.
`-Both`	Adds the probe points in both Golden and Revised designs.
`-Strict`	Specifies the added the probe point is a strict probe point. Not-mapped strict probe point will cause dynamic flattening in hierarchical flow, hierarchical verification to report "Inconclusive", and report verification to report "INCOMPLETE" and verification to be. *This is the default*.

`-NOStrict`	Specifies the added the probe point is a non-strict probe point. Not-mapped non-strict probe point will not cause dynamic flattening in hierarchical flow.
`-Verbose`	Prints out additional messages when adding a probe point.

Example

For example, the following commands add a probe point named probe_1 at net path /inst1/SO in the golden design, add a probe point named probe_1 at net path /inst_sub/SO in the revised design, and add two probe points named probe_2 at module instance pin /inst2/SE in both designs. The probe points named probe_1 and probe_2 are mapped once the system mode is changed to LEC mode, and compared by 'add compared point probe*' and 'compare' commands.

```
add probe point /inst1/SO probe_1 -golden
add probe point /inst_sub/SO probe_1 -revised
add probe point /inst2/SE probe_2 -both
set system mode lec
add compared points probe*
compare
```

Related Commands

ADD RENAMING RULE

```
ADD REnaming Rule
     [[-PIN_MULTIDIM_TO_1DIM]
        [-ADD | -NOADD]
        [-NOASCEND | -ASCEND]
        [-VERBOSE] ]
     [<rule_name> <string> <string>
     [[ -MAp ]
        [-TYpe < PI | E | Z | DFf | DLat | CUt | BBox | PO> ...
          |-NOTYpe <PI | E | Z | DFf | DLat | CUt | BBox | PO> ...]
        |-LIMIT_MODule <module_name>
        |-MOdule
        |-PIn
        |[-INSTance | -NOINSTance]
        [-BBox <module_name>]]
     [-REPlace]
     [-Golden | -Revised | -BOth] ]
     (Setup / LEC Mode)
```

Specifies renaming rules for key point mapping, module renaming (when reading in the library and designs for hierarchical comparisons), pin renaming for blackboxes, and instance renaming for module pairing (when writing the hierarchical dofile).

Use the REPORT RENAMING RULE command to display the list of all renaming rules. Conformal applies renaming rules sequentially, in the order they were added.

Key Point Mapping

When you define renaming rules in the Setup system mode, they guide the automatic mapping process that occurs during the system mode switch from Setup to LEC. When you are in the LEC system mode, and find that the key point mapping is not complete, define additional renaming rules and repeat key point mapping to improve the mapping results. The automatic mapping process refers to the naming specified by the final renaming rules.

Module Renaming

You must use this command before the `WRITE HIER_COMPARE DOFILE` command. It helps map modules together for hierarchical comparisons.

Pin Renaming

This command applies to the specified blackbox or to all blackboxes, which is the default.

Renaming Rule Structure

When defining renaming rules, the first string specifies the pattern to be matched; the second string specifies how Conformal is to rename or make substitutions. The first string can contain expressions of the following types:

`%d`	Matches one or more digits, `[0-9]+`
`%a`	Matches one or more alphabetical characters, `[a-zA-Z]+`
`%s`	Matches one or more digits or alphabetical characters, `[0-9a-zA-Z]+`
`%u`	Matches one or more alphabetical characters or underscores, `[a-zA-Z_]+`
`%w`	Matches one or more digits or alphabetical characters or underscores, `[0-9a-zA-Z_]+`
`x`	Matches character "`x`"
`abc`	Matches string "`abc`"
`.`	Matches any single character
`*`	Matches zero or more repetitions of the preceding expression
`+`	Matches one or more repetitions of the preceding expression
`^bol`	Matches "`bol`" only when it occurs at the beginning of a string
`eol$`	Matches "`eol`" only when it occurs at the end of a string
`x|abc`	Matches "`x`" or "`abc`"
`[oz!]`	Matches "`o`", "`z`", or "`!`"
`(pattern)`	Matches anything that is matched by "`pattern`" and renders it referable (through `@n`) in the substitution string

Any character can be preceded by the escape character "`\`" to cancel any special meaning it has. Use the escape character whenever any of the following special characters represents a simple character.

```
    %   .   *   +   ^   $   |   (   )   [   ]   \
```

The second string can contain expressions of the following types:

`abc`	Replaces string "`abc`" for each matched string

`@n`	Replaces the string that matches the nth `%d`, `%a`, `%s`, or a pattern enclosed in parentheses. The `n` is a digit other than `0`, and you can use `@{nn}` to refer to further matches (that is, `10...99`)
`#(expr)`	Where "`expr`" is an arbitrary expression that can only contain constant integers, `@n` expressions, and the operators `+`,`-`,`*`, `/` and `( )`

The following table shows implementation examples for various pattern-matching and substitution strings. For pattern-matching strings, use parentheses to group individual patterns into a single pattern, as demonstrated in the example `(ab|de)*`.

First String	Second String	Source	Result
`%a%d`	`@1[@2]`	`xyz123`	`xyz[123]`
`%a_%d`	`@1[@2]`	`arr_5`	`arr[5]`
`_z_`	`/`	`_z_top_z_inst`	`/top/inst`
`^abc`	`XYZ`	`abcabc`	`XYZabc`
`abc`	`XYZ`	`abcabc`	`abcXYZ`
`abc`	`XYZ`	`abcabc`	`abcabc`
`^abc`	`XYZ`	`abc`	`XYZ`
`[oz!]`	`$`	`aaoaazaa!aa`	`aa$aa$aa$aa`
`\%d\%s`	`AA`	`%a%d%s%b`	`%aAA%b`
`\(ab`	`ZZ`	`(abcd`	`xxZZcd`
`%s\.%d`	`@1[@2]`	`xx.123`	`xx[123]`
`(x|abc`	`YY`	`abcx abcx`	`YYYY YYYY`
`(ab|de)*`	`YY`	`ababdeab`	`YY`
`%s&d`	`@1[#(7-@2)]`	`abc3`	`abc[4]`
`%s%d`	`@1[#(2*@2)]`	`abc5`	`abc[10]`
`reg%d\[%d\]`	`reg[@2]`	`reg2[5]`	`reg[5]`

Important

Do not include the forward slash, "`/`", at the top level for either the first or second string.

For example, express `/top_module/adder/reg[5]` as `top_module/adder/reg[5]`.

Tcl Command

```
add_renaming_rule
```

Parameters

`-PIN_MULTIDIM_TO_1DIM`	
	Allows you to create renaming rules to map multidimensional array pins to one-dimensional array pins.
`-ADD`	Shows the pins found when adding the rules into the system. *This is the default.*
`-NOADD`	Shows the pins found without adding the rules.
`-NOASCEND`	Renames the pins in an descending order. *This is the default.*
`-ASCEND`	Renames the pins in an ascending order.
`-VERBOSE`	Shows the renaming patterns.
`<rule_name>`	Specifies a rule identification name assigned to a specific renaming rule.
`<string> <string>`	The first string represents the pattern to be matched. The second string represents the substitution pattern.
`-MAp`	Specifies that the renaming rule applies to key point mapping. Note that it also specifies an instance renaming rule with the same rule name and rule pattern. *This is the default.*
`-TYpe`	Renames all key points with the specified type. Renaming rules for instance renaming will not be created. The available types are as follows:

`PI`	Primary Inputs

`E`	TIE-E gates
`Z`	TIE-Z gates
`DFF`	D flip-flops
`DLAT`	D-latches
`CUT`	Artificial gates for breaking combinational feedback loops
`BBOX`	Blackboxes
`PO`	Primary Outputs

`-NOTYpe` Renames all key points except the specified types. Renaming rules for instance renaming will be created. The available types are as follows:

`PI`	Primary Inputs
`E`	TIE-E gates
`Z`	TIE-Z gates
`DFF`	D flip-flops
`DLAT`	D-latches
`CUT`	Artificial gates for breaking combinational feedback loops
`BBOX`	Blackboxes
`PO`	Primary Outputs

`-LIMIT_MODule <module_name>`

Specifies that the renaming rule only applies to the specified modules (including children). Where `<module_name>` can be a list of names or wildcard name, such as "mod*". The option is only for key point mapping.

`-MOdule` Specifies that the renaming rule applies to module renaming when the library and design are read in.

`-PIn` Specifies that the renaming rule applies to pin names of blackboxes.

`-INSTance` Specifies that the renaming rule applies to instance renaming for module pairing when writing the hierarchical dofile. Note that it also specifies a renaming rule for blackbox keypoints with the same rule name and rule pattern.

`-NOINSTance`	Do not specify instance renaming rules when specifying a mapping renaming rule.
`-BBox <module_name>`	Specifies that the pin renaming rules apply to the specified blackbox module.
`-REPlace`	Allows the redefinition of an existing renaming rule. *Tip* The difference between using `add renaming rule -replace` as opposed to using `delete renaming rule` followed by `add renaming rule` is that renaming rules that are redefined remain in the same position in the list of renaming rules. By deleting and adding a rule, the new definition will appear at the end of the list. In some cases, the order in which renaming rules are applied might affect the result.
`-Golden`	Specifies that the renaming rule applies to the Golden design. *This is the default.*
`-Revised`	Specifies that the renaming rule applies to the Revised design.
`-BOth`	Specifies that the renaming rule applies to both the Golden and Revised designs.

Example

In the following command, `y2[1:0][2:0]` in module `test2` is renamed `y2[5:0]`:

```
add renaming rule -pin_multidim_to_1dim
// Rule created for (test2) y2[1:0][2:0]
// Rule created for (test1) y1[1:0][1:0]
// Rule created for (top) y2[1:0][2:0]
// Rule created for (top) y1[1:0][1:0]
// Rule created for (top) ym[2:3][2:0][1:0]
// 5 rules created. Rules for top module must be manually validated.
```

You can use the `REPORT RENAMING RULE` command to view the added rules.

When in Tcl mode, it is recommended to wrap the pattern and substitution strings in curly braces to protect them from command substitution. For example, the following changes left square brackets to an underscore:

```
add_renaming_rule r1 {\[ _}
```

Related Commands

CHANGE NAME

DELETE RENAMING RULE

MAP KEY POINTS

READ DESIGN

READ LIBRARY

REPORT RENAMING RULE

SET MAPPING METHOD

SET NAMING RULE

TEST RENAMING RULE

ADD RETENTION MAPPING

```
ADD REtention_register Mapping
     <rule_name>
     <-Module <module_name*> | -Instance <instance_pathname*>
        | -NOTag | -Tag <tag_name1*> -Tag <tag_name2*> ... >
        <-NOAttribute | -Attribute <attribute name>>
     [-TYpe <ALL | DFF | DLAT>]
     (Setup / LEC Mode)
```

Note: This is a Conformal Low Power command.

Adds the state retention mapping rules for validation of technology mapping of the sequential elements (DFFs or DLATs) from RTL to gate-level, gate-level to gate-level, or RTL to RTL.

For a description of the default rules that are added by the system, see CHECK LOWPOWER CELLS.

Tcl Command

```
add_retention_register_mapping
```

Parameters

`<rule_name>`	Specifies a rule identification name assigned to a specific retention mapping rule.
`-Module`	Specifies the module name in the Golden design. All the DFFs or DLATs under the named module will be subjected to this rule.
`-Instance`	Specifies the instance pathname in the Golden design. Here instance pathname refers to only DFF or DLAT instances. The named DFF or DLAT instance(s) would be subjected to this rule. Wildcards (*) are supported for the instance pathname. When using a wildcard, it might point to multiple instances.
`-NOTag`	When the Golden side is an RTL design, this implies that all the DFFs or DLATs which do not have any tag-name associated with their 'process' or 'always' block will be subjected to this rule.

`-Tag`	Specifies the tag name in the RTL Golden side. The 'tag name' refers to the label names used with a) 'process' blocks in the VHDL RTL and b) 'always' blocks in the Verilog RTL. All the DFFs or DLATs under the tag-named block will be subjected to this rule. Wildcards (*) are supported.
`-Attribute`	Specifies the 'power gating cell attribute' for the DFFs or DLATs in the Revised netlist. Different power gating cell attributes are defined for different sets of retention cells in the Synopsys Liberty library format. When synthesized (technology mapped), the DFFs or DLATs in the Golden design (specified using module-name, instance-name, or tag-name) should have the named attribute in Revised netlist.
`-NOAttribute`	Specifies that, when synthesized (technology mapped), the DFFs or DLATs in the Golden design (specified using module-name, instance-name, or tag-name) should not have any attribute (power gating cell attribute) in Revised netlist. In other words, the specified DFFs or DLATs should be technology mapped as ordinary or non-retention cells.
`-TYpe <ALL | DFF | DLAT>`	Indicates the sequential element type on which to apply the retention mapping rule. `ALL` applies the retention mapping rule to all sequential elements (both DFF and DLAT type). This is the command default when you do not specify the `-TYpe` option. `DFF` applies the retention mapping rule to DFFs only, and `DLAT` applies the retention mapping rule to DLATs only.

Examples

- The following command verifies that all registers with a tag label `lp_sel*` are implemented with a state retention cell whose `power_gating_cell` attribute is `LPRET_DFF1`:

```
add retention mapping R0 -tag lp_sel* -attribute LPRET_DFF1
```

- The following command verifies that all registers in module `blockA` are implemented with a state retention cell whose `power_gating_cell` attribute is `LPRET_DFF1`:

```
add retention mapping R1 -module dma -attribute LPRET_DFF1
```

- The following command verifies that all registers with instance name `/U0/*/fifo_dma*` are implemented with a state retention cell whose `power_gating_cell` attribute is `LPRET_DFF2`:

```
add retention mapping R2 -instance "/U0/*/fifo_dma*" -attribute LPRET_DFF2
```

Related Commands

CHECK LOWPOWER CELLS

DELETE RETENTION MAPPING

REPORT RETENTION MAPPING

ADD RULE FILTER

```
ADD RUle Filter
     <filter_name> [<condition ...>]
     [-DESCription <description>]
     [-FILTER <filter_expression> [-REGEXP]]
     [-RULe <rule_name*>] ...
     [-REPlace]
     (Setup Mode)
```

Creates and modifies report rule filters.

Tcl Command

`add_rule_filter`

Parameters

`<filter_name>`	Specifies the name of the rule filter to add or apply. If `<condition>` is not specified, the rule filter must already exist.
`<condition> ...`	Specifies the condition used to determine if a rule occurrence should be filtered or not. If the condition is true, the rule occurrence will be filtered. A condition is required when adding a new rule filter.

For `<condition>`, specify one or more of the following filter conditions:

`-RULe <rulename*>`

Filters out all occurrences of specified rule(s).

`-SEVerity <Error | Warning | Note | Ignore>`

Filters out all occurrences of specified severity levels.

`-MESsage <pattern*>`

Matches rule occurrences whose verbose message matches the specified pattern. For the condition `-message <pattern>` to be true, the pattern must match the entire verbose message of the rule occurrence.

For example, if the pattern is based on some words in the middle of the message, the pattern should start and end with the asterisk (`'*'`) wildcard character.

Some characters (such as, `'*'`, `'?'`, `'\'`, `'['`, and `']'`) have a special meaning in wildcard patterns. To match these characters literally, escape them with a backslash ('`\`'). For example:

```
add rule filter rule1 -rule CPF_ISO1 -and -message \
   '*from ff1/out_f_reg\[0\]/Q*'
```

`-OBJect [-Hierarchical] [-Recursive] <name*>`

Matches rule occurrences related to objects that match the specified pattern.

This condition is only available for SDC rules.

`-Hierarchical` matches any instance in the design hierarchy against the specified pattern (analogous to the SDC command '`get_pins -hier <pinname>`')

`-Recursive` matches any object under an instance that matches the specified pattern (analogous to the Unix command '`grep -r ... <dirname>`')

`-Hierarchical` and `-Recursive` can be used together.

`-SDCMatch <pattern*>`

Matches rule occurrences related to SDC statements matching the specified pattern. Note: the string representing the SDC statement is the one displayed in the SDC command browser, not the one from the SDC file.

This condition is only available for SDC rules.

`-NOT <condition>`

Matches rule occurrences not matched by the sub-condition.

`<condition> [-AND] <condition>`

Matches rule occurrences matched by both sub-conditions.

`<condition> -OR <condition>`

	Matches rule occurrences matched by either sub-condition.
`( <condition> )`	
	Conditions can be grouped using parentheses to enforce precedence. White space is required around the parentheses.
`-DESCription <description>`	
	Specifies a description for the added rule filter.
`-FILTER <filter_expression>`	
	The rule filter will apply to occurrences for which the specified `filter_expression` evaluates to true.
	The syntax of the `filter_expression` is the same as for the `find` Tcl command.
`-REGEXp`	Specifies that the pattern matching operators used inside `filter_expression` (such as =~) should use regular expressions as patterns, rather than glob-style patterns.
`-RULe <rule_name*>`	Specifies the rule occurrences to which the filter should be applied. By default, `-rule` is part of the condition.
	This accepts wildcards.
`-REPlace`	Specifies that the filter name can be that of an existing filter, which is then modified.
`-INClude`	Specifies that the filter will cause any matching occurrence not to be filtered out, unless this is reversed by another filter down the list.

Examples

The following adds a filter called `f1` that filters out occurrences where the message contains the string "`special_cell`":

```
add rule filter f1 -message "*special_cell*"
```

The following adds a filter called `f2` that filters out occurrences of `PDM4d` where the message contains the string `"special_cell"`:

```
add rule filter f2 -rule PDM4d -AND -message "*special_cell*"
```

Related Commands

DELETE RULE FILTER

READ RULE CHECK

REPORT RULE FILTER

WRITE RULE CHECK

ADD SEARCH PATH

```
ADD SEarch Path
     <pathname ...>
     [ | -Design | -Library | -POWER_intent]
     [-RECursive]
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Defines additional search paths outside the current directory for filenames you use in the `READ DESIGN` and `READ LIBRARY` commands. This command is necessary because the default is to search for filenames in the current directory; but your design or library can include filenames that are housed in other directories. The default search path is the current working directory.

When you add multiple search paths to the list, Conformal does the search in the order that those paths were added to the list.

Use the `REPORT SEARCH PATH` command to display all search paths. Use the tilde character (~) to shorten the specified path.

Tcl Command

```
add_search_path
```

Parameters

`<pathname ...>`	Specifies the search path for filenames used in the `READ DESIGN` and `READ LIBRARY` commands.
`-Design`	The `READ DESIGN` command uses the specified search path.
	If you do not specify `-library` or `-design`, Conformal applies this command to *both* the `READ DESIGN` and `READ LIBRARY` commands.
`-Library`	The `READ LIBRARY` command uses the specified search path.
	If you do not specify `-library` or `-design`, Conformal applies this command to both the `READ DESIGN` and `READ LIBRARY` commands.
`-POWER_intent`	This is a low power command option. Specifies the search path for power intent files.

`-RECursive`	Adds the subdirectories of the specified directory into the search paths.
`-Both`	Specifies that the search path applies to both the Golden and Revised designs. *This is the default.*
`-Golden`	Specifies that the search path applies to the Golden design and library.
`-Revised`	Specifies that the search path applies to the Revised design and library.

Related Commands

DELETE SEARCH PATH

READ DESIGN

READ LIBRARY

REPORT SEARCH PATH

ADD SEQ_CORRESPONDENCE

```
ADD SEQ_CORRespondence
     <golden_identifier> <revised_identifier>
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Adds the Golden register and the Revised state point as the pair of sequential corresponding points.

After adding the sequential corresponding pairs, use the `ANALYZE RETIMING -general` command to retime the Revised design to the state points according to the sequential correspondence information.

Tcl Command

```
add_seq_correspondence
```

Parameters

`<golden_identifier>`	Specifies the sequential corresponding gate ID or instance pathname for the Golden register.
`<revised_identifier>`	Specifies the sequential corresponding gate ID or instance pathname for the Revised state point.

Example

The following commands add the sequential corresponding points and perform general retiming analysis:

```
add seq_corr reg1 g1
add seq_corr reg2 g2
analyze retiming -general -verbose
```

Related Commands

ANALYZE RETIMING

DELETE SEQ_CORRESPONDENCE

REPORT SEQ_CORRESPONDENCE

SET RETIMING OPTION

ADD SUPPLY

```
ADD SUpply
     <object_list>
     [-POWER | -GROUND]
     [-ROOT | -Module <name_list*> | -ALL]
     [-PORT | -GLOBAL ]
     [-Golden| -Revised | -Both]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Defines power and ground ports of a module or the global power and ground signals for the entire design.

Tcl Command

`add_supply`

Parameters

`<object_list>`	Specifies that a list of net or port names, each separated by a space, that will be specified as power or ground.
`-POWER`	Set the specified objects with power attribute. *This is the default.*
`-GROUND`	Set the specified objects with ground attribute.
`-ROOT`	Applies this supply attribute to the specified objects in the current scope and all hierarchy of this scope. *This is the default.*
`-Module <name_list*>`	Applies the attribute setting to the specified module. This accepts wildcards.
`-ALL`	Applies the attribute setting (`-power`/`-ground`) to the objects for all modules.
`-PORT`	The defined object(s) must be the port(s) at the root or the specified module level. *This is the default.*
`-GLOBAL`	The defined object(s) could be the port(s) and wire(s) in the hierarchy of the root or the specified module.

`-Golden`	Specifies that the listed names are from the Golden design. *This is the default.*
`-Revised`	Specifies that the listed names are from the Revised design.
`-Both`	Specifies that the listed names are from both the Golden and Revised designs.

Related Commands

DELETE SUPPLY

REPORT SUPPLY

ADD TIED SIGNALS

```
ADD TIed Signals
     <0 | 1>
     <name* ...>
     [-Net | -Pin]
     [-ROot | -Module <module_name*> | -All]
     [  | -Design | -Library]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Assigns the specified floating nets or pins to a 0-state or a 1-state in the Golden or Revised design. These tied signals are classified in the *User* class of tied signals. The original tied signals of the design are classified in the *System* class of tied signals.

Wildcard: The wildcard (*) represents any zero or more characters in net, pin, and module names.

Tcl Command

```
add_tied_signals
```

Parameters

`0`	Ties the floating nets or pins to a 0-state.
`1`	Ties the floating nets or pins to a 1-state.
`<name* ...>`	Specifies a list of names that correspond to either floating nets or floating pins where you intend to add the tied signal.
`-Net`	Specifies that the listed names are net names. *This is the default.*
`-Pin`	Specifies that the listed names are pin names.
`-ROot`	Specifies that the floating net or pin resides in the current root module. *This is the default.*
`-Module <module_name*>`	Specifies that the floating net or pin resides in the specified module. *The default is the root module.* This accepts wildcards.

`-All`	Applies the tied signals to "all" the modules. `-All` applies within the given defaults.
`-Design`	Applies the tied signals to the design. If you do not specify `-design` or `-library`, Conformal applies tied signals to both designs and libraries.
`-Library`	Applies the tied signals to the library. If you do not specify `-design` or `-library`, Conformal applies tied signals to both designs and libraries.
`-Golden`	Adds the tied signals to the Golden design. *This is the default.*
`-Revised`	Adds the tied signals to the Revised design.
`-Both`	Adds the tied signals to both the Golden and Revised designs.

Related Commands

DELETE TIED SIGNALS

REPORT FLOATING SIGNALS

REPORT TIED SIGNALS

ANALYZE ABORT

```
ANAlyze ABort
     [-All
       | <<gate_id> | <instance_pathname> | <pin_pathname> ...
         [-Golden | -Revised]> | -Number <number>]
     [-CLass <Abort | Notcompared | ALL>]
     [-PRESERVE_NET]
     [-SNAPSHOT_DIRectory <directory>]
     [-Summary | -COmpare [-THREADS <integer>]]
     [-Verbose]
     [-XORTREE]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Analyzes abort points and recommends actions to help solve the abort points. This command can also provide useful information for further abort investigation.

Tcl Command

`analyze_abort`

Parameters

`-ALL`	Analyzes all abort points. *This is the default*.
`<gate_id>`	Specifies the gate ID for abort analysis.
`<instance_pathname>`	Specifies the instance pathname for abort analysis.
`<pin_pathname>`	Specifies the pin pathname for abort analysis.
`-Golden`	Specifies whether the gate ID or pathname is in the Golden design. *This is the default*.
`-Revised`	Specifies whether the gate ID or pathname is in the Revised design.
`-Number <number>`	Specifies the number of abort points to analyze.

`-CLass <Abort | Notcompared | ALL>`

	Specifies the class of points to analyze. Select `Abort` to analyze abort points, `Notcompared` to analyze not-compared points, or `ALL` to analyze both abort and not-compared points.
`-PRESERVE_NET`	Reports the critical nets that should be preserved in the synthesis flow. Preserving these critical nets in the synthesis flow can also help resolve aborts.
`-SNAPSHOT_DIRectory <directory>`	
	When this option is specified, LEC will create a snapshot of RTL annotation to the user-specified directory.
`-Summary`	Prints out the summary count of the abort points. *This is the default*.
`-COmpare [-THREADS <integer>]`	
	Automatically compares the aborted points. Use this option when running the `ANALYZE ABORT` command. `-THREADS` specifies the number of threads for abort analysis. This supersedes any `SET PARALLEL OPTION` command's `-threads` setting. **Note:** The `-THREADS` option activates the use of parallel algorithms which could help solve more abort points. Depending on the complexity of the aborts, the run time may take longer or run indefinitely.
`-Verbose`	Prints out additional information, such as RTL statistics.
`-XORTREE`	Force to enable resolving xortrees' aborts under multiple threads.

Example

The following commands run abort analysis after the initial compare:

```
compare
analyze abort -compare
```

Related Commands

COMPARE

RUN HIER_COMPARE

SET ANALYZE OPTION

ANALYZE COMPARE

```
ANAlyze COMpare
     [-ABORT_Stop <integer>]
     [-RESOURCEFILE <filename>]
     [-THREADS <integer>[,<integer>]]
     [-NONEQ_Stop <integer>]
     [-VERbose]
     (LEC Mode)
```

Note: This requires a Smart LEC license.

Smart Compare is done through the analyze compare command. This command executes the entire comparison step in the most optimal turnaround time. Based on the design's characteristics, analyze compare automatically executes the most appropriate combination of commands and options to complete the comparison

Tcl Command

```
analyze_compare
```

Parameters

`-ABORT_Stop <integer>`	Stops the command after finding the specified number of abort points.
`-RESOURCEFILE <filename>`	Specifies the resource filename to analyze the datapath modules. This is used for DC netlists.
`[-THREADS <integer>[,<integer>]]`	Specifies the minimum and maximum number of threads. If only one number entered, this specifies both the minimum and maximum number of threads. For example, `-threads 2` specifies two threads; `-thread 2,4` specifies a minimum of two threads, and a maximum of four threads. If `-threads` is not specified, LEC honors the setting from `set_parallel_option -threads`.
`-NONEQ_STOP <integer>`	Stops the comparison after finding the specified number of nonequivalent points.

`-VERBOSE`	Displays detailed comparison information, such as datapath analysis, the circuit duplication in multithreaded comparison.

Example

To use Smart Compare in a hierarchical comparison, use `write_hier_compare_dofile -compare_string` or `analyze_hier_compare_dofile -compare_string` to replace the default compare command with the `analyze_compare` command. The following is a sample dofile for an RTL-to-synthesized gate netlist hierarchical comparison using `analyze_compare`.

Genus Synthesized Netlist

```
// Reads in the library, design, and resource files
read_design rtl.v -golden
read_design netlist.v -revised
read_implementation_information fv/top -golden fv_map -revised map_lecv
...
// Enables auto setup analysis
set_analyze_option -auto
// Creates the hierarchical dofile script that will compare the designs
write_hier_compare_dofile hier.do -compare_string "analyze_compare -verbose"
go_hier_compare hier.do
or
// Reads in the library, design, and resource files
read_design rtl.v -golden
read_design netlist.v -revised
read_implementation_information fv/top -golden fv_map -revised map_lecv
...
// Enables auto setup analysis
set_analyze_option -auto
set_flatten_model -enable_analyze_hier_compare
set_system_mode lec
// Creates the hierarchical dofile script that compares the designs
analyze_hier_compare -dofile hier.do -compare_string "analyze_compare -verbose"
go_hier_compare hier.do
```

DC Synthesized Netlist

```
// Reads in the library design files
```

```
read_design rtl.v -golden
read_design netlist.v -revised
...
// Enables auto setup analysis
set_analyze_option -auto
// Creates the hierarchical dofile script that will compare the designs
write_hier_compare_dofile hier.do -compare_string \
"analyze_compare -verbose -resourcefile resourcefile.name"
go_hier_compare hier.do
or
// Reads in the library design files
read_design rtl.v -golden
read_design netlist.v -revised
...
// Enables auto setup analysis
set_analyze_option -auto
set_flatten_model -enable_analyze_hier_compare
set_system_mode lec
// Creates the hierarchical dofile script that compares the designs
analyze_hier_compare -dofile hier.do -compare_string \
"analyze_compare -verbose -resourcefile resourcefile.name"
go_hier_compare hier.do
```

Related Commands

SET DATAPATH OPTION

SET PARALLEL OPTION

WRITE HIER_COMPARE DOFILE

ANALYZE CONSTANT INFORMATION

```
ANAlyze CONSTant INFOrmation
     [-CHECKconsistency [-verbose]]
     [-DEBUG -verbose]
```
(LEC Mode)

Analyze the sequential constant from guidance in LEC mode. This can help analyze the consistency and further use GUI to debug the source of the problem when sequential constant is not applied.

Tcl Command

```
analyze_constant_information
```

Parameters

`-CHECKconsistency [-verbose]`	Check if there is a register in the netlist that has a mismatched behavior to guidance information. ’Conflict’ means the LEC models it but has a different conclusion to guidance. ’Not applied’ means LEC does not model the register where guidance tells. ’Unreachable’ is the sub class of Not-applied, where the register is unreachable.
`-verbose`	Shows the detailed information.
`-DEBUG -verbose`	Reports the source of DFF that cannot be remodeled as sequential constant. The reported list is sorted by the size of support. Users can open the GUI to debug why it cannot be remodeled. **Note**: This is a debug utility. Do not include it in sign-off flow.

ANALYZE CONSTRAINT

```
ANAlyze COnstraint
     [-PROPagate]
     [-REPort]
     [-GOLden | -REVised]
```
(LEC Mode)

Note: This command requires a GXL license.

Analyzes constraints in the design. Conformal can create more accurate constraints to resolve some sequential verification issues.

Tcl Command

```
analyze_constraint
```

Parameters

`-PROPagate`	Create the constraints to the downstream logic of current constraints in the specified design.
`-REPort`	Report all constraints in the specified design.
`-GOLden`	Specifies that the analyzed netlist is the Golden design. *This is the default.*
`-REVised`	Specifies that the analyzed netlist is the Revised design.

ANALYZE DATAPATH

```
ANAlyze DAtapath
     [-MODULE [-RESOURCEFILE <filename>] [-ISOLATE_ABORT_MODULE] ]
     [-DIAGNOSIS]
     [-EFFort <MEDium | HIgh>]
     [-MERGE | -NOMERGE]
     [-NOADDERTREE | -ADDERTREE]
     [-NOSHARE | -SHARE]
     [-NOWORDLEVEL | -WORDLEVEL]
     [-NOFLOWGRAPH | -FLOWGRAPH]
     [-SHARE_OPerator <r1 r2 [.. rN]> ]
     [-THREADS <integer>[,<integer>]]
     [-Verbose]
     [-GOLDEN | -REVISED]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Analyzes datapath modules. Based on the results of the analysis, Conformal can automatically resolve multipliers, operator merging, and resource sharing problems.

Note: You cannot run datapath analysis without first mapping the Revised design key points to the Golden design key points.

By default, this command performs bit-level analysis.

Tcl Command

```
analyze_datapath
```

Parameters

`-MODULE`	Applies analysis on the datapath modules. The default is in the Revised design netlist.
`-RESOURCEFILE <filename>`	Specifies the resource filename to analyze the datapath modules. This is used for DC netlists.

`-ISOLATE_ABORT_MODULE`	
	Isolates the module which is aborted during module-based datapath (MDP) analysis. The module's gate-level netlist will be abstracted into RTL for comparison at the top module.
`-DIAGNOSIS`	Displays information that can help diagnose the low quality of datapath analysis.
`-EFFort <MEDium | HIgh>`	
	Specifies the effort level. Choose `MEDium` (the default), or `HIgh` to help provide better analysis of some multipliers, but might increase the analysis run time.
`-MERGE`	Applies the operator merging technique. *This is the default.*
`-NOMERGE`	Do not apply the operator merging technique.
`-NOADDERTREE`	Do not automatically add parentheses to the input operands of adder trees. *This is the default.*
`-ADDERTREE`	Automatically adds parentheses to the input operands of adder trees.
`-NOSHARE`	Do not apply the resource sharing technique. *This is the default.*
`-SHARE`	Analyzes the design for datapath resource sharing.
`-NOWORDLEVEL`	Do not apply word-level datapath analysis. *This is the default.*
`-NOFLOWGRAPH`	Do not apply flow-graph based datapath analysis. *This is the default*.
`-FLOWGRAPH`	Applies flow-graph based datapath analysis. This helps analyze complex and advanced datapath clustering, product-of-sum multipliers or datapath with control logics or constants.
`-WORDLEVEL`	Applies word-level datapath analysis. This helps analyze advanced word-level optimizations on designs with complex adder-tree clustering, product-of-sum, XOR-tree, or datapath with control logics or constants.
`-SHARE_OPerator`	Shares the specified operators. See the example for the recommended flow. **Note:** If this option is specified, only the sharing is performed, and does not run datapath analysis.
`-THREADS <integer>[,<integer>]`	

	Specifies the minimum and maximum number of threads for module-based datapath analysis. If only one number entered, this specifies both the minimum and maximum number of threads. Currently it is only effective with the `-MODULE` option. For example, '`-threads 2`' specifies two threads; '`-thread 2,4`' specifies a minimum of two threads, and a maximum of four threads.
`-Verbose`	Provides additional information.
`-GOLDEN`	Applies to the Golden design. *This is the default*.
`-REVISED`	Applies to the Revised design.

Examples

- The following commands show an example of the recommended flow when using the `-SHARE_OPerator` option:

```
analyze datapath -verbose -share_operator mult_30 mult_31
// Note: mult_30: shared
analyze datapath -verbose -share_operator mult_32 mult_33
// Note: mult_32: shared
analyze datapath -verbose
// Note: add_2(clustered): quality evaluated 70% success
// Note: mult_30: quality evaluated 80% success
// Note: mult_32: quality evaluated 100% success
```

- The following commands apply datapath module-based analysis followed by the datapath operator-level analysis:

```
analyze datapath -verbose -module -resourcefile resourcefile.name
analyze datapath -verbose
```

- The following is an example of what is displayed when using the `-diagnosis` option:

```
================================================================================
[GOLD] mult_26 (z = in1 * in2)
--------------------------------------------------------------------------------
Boundary           Match(%)     Bits(#)      Const(#)   Unreach(#)
--------------------------------------------------------------------------------
IN1 (unsigned)      100          16           0          0
IN2 (unsigned)      100          16           0          0
OUT (unsigned)      0            32           0          0
INTERNAL            8            132          15         -
--------------------------------------------------------------------------------
Note:
PPGEN boundary is not perfectly matched.
--------------------------------------------------------------------------------
// Note: mult_26: quality evaluated 10% success
```

Related Commands

ANALYZE MODULE

COMPARE

REPORT COMPARE DATA

REPORT DATAPATH OPTION

REPORT DATAPATH RESOURCE

REPORT HIER_COMPARE RESULT

REPORT MULTIPLIER OPTION

SET DATAPATH OPTION

SET MULTIPLIER IMPLEMENTATION

SET MULTIPLIER OPTION

WRITE HIER_COMPARE DOFILE

ANALYZE DC

```
ANAlyze DC
     <[-REMOVE_RANGE_CONSTRAINT]
     [-REMOVE_RANGE_CONSTRAINT_X]
     [-RESOLVE_RANGE_CONSTRAINT]
     [-REPORT_REDUNDANT_OVL_CONSTRAINT]
     [[-REPORT_STATUS | -REVIEW_CONSTRAINT] | [-VERBOSE]]
     >
     (LEC Mode)
```

Note: This is a Conformal XL command.

This command can improve compare performance by turning off DC gates. Having a large number of DC gates in your design can slow down the compare process, resulting in aborts.

DC gates are usually caused by range constraints (VHDL only) and explicit X assignments. This command focuses only on DC gates that are caused by range constraints.

This command can also detect redundant OVL assertion constraints in the Golden netlist.

You must use this command with one of the following options.

Tcl Command

```
analyze_dc
```

Parameters

`-REMOVE_RANGE_CONSTRAINT`	Turns off DC constraints that are due to VHDL range constraint and System Verilog assertions. Note: This option may cause false NEQs.
`-REMOVE_RANGE_CONSTRAINT_X`	Removes x-assignments modeled due to range constraints in VHDL. Note: This option may cause false NEQs.
`-REPORT_REDUNDANT_OVL_CONSTRAINT`	

	Reports any redundant OVL assertion constraints in the Golden netlist.
`-RESOLVE_RANGE_CONSTRAINT`	In some cases, turning off DC gates can result in false NEQs. For example, when the revised netlist is optimized according to DC space. Use this option to resolve false NEQs by restoring ALL DC gates in the Cone-Of-Influence (COI) of the corresponding compare points in the Golden netlist. **Note:** This option also restores all DC gates in the COI of the aborted compare point, as they might result in false NEQs later on in the comparison. You can use this option only if you used the `-remove_range_constraint` option and it resulted in a compare with at least one NEQ or abort point.
`-REPORT_STATUS`	Report the status of DC gates.
`-REVIEW_CONSTRAINT`	Reports always-on/off DC gates with related constraints that are recommended to be reviewed.
`-VERBOSE`	Provide detailed information when reporting.

Related Commands

COMPARE

READ DESIGN

ANALYZE DESIGN

```
ANAlyze DEsign
     [|-Golden] [-Revised | -Both]
     [|-SEQ_INPUT_INTERACT_STRuctural | -SEQ_INPUT_INTERACT_LOgical
     |-CLOCK_GATING_STRucture]
     [-SEQ_RESET_X]
     [-TRIGGER_EDGE_DIFF[|-HIer]]
     (LEC Mode)
```

Note: This is a Conformal XL command.

Analyzes designs and reports structures that should be warned.

Tcl Command

`analyze_design`

Parameters

`-Golden`	Applies analysis on the Golden design only.
`-Revised`	Applies analysis on the Revised design only.
`-Both`	Applies analysis on both the Golden and Revised design.
`-SEQ_INPUT_INTERACT_STRuctural`	Structurally analyzes the input cones overlapping of sequential gates. When a gate is reported, it implies zero-delay simulation mismatch can happen.
`-SEQ_INPUT_INTERACT_LOgical`	Logically analyzes the input cones overlapping of sequential gates. When a gate is reported, it implies that zero-delay simulation mismatch will happen. This option can take noticeable longer runtime.
`-CLOCK_GATING_STRucture`	Analyzes clock gating structure. Specifically, a DFF is reported if some deglitching DLATCH in its clock input cone can be on when the DFF clock is high.

`-SEQ_RESET_X`	Analyzes clock gating structure. Specifically, a DFF is reported if its clock gating enable signal depends on the DFF output.
`-TRIGGER_EDGE_DIFF`	Analyzes and reports corresponded points that come from negative/positive edge triggered RTL state elements and positive/negative edge triggered cells.
`-HIer`	Specifies that the report for `-TRIGGER_EDGE_DIFF` be for the whole design after the hierarchical comparison. By default, the report generated by `-TRIGGER_EDGE_DIFF` is for the top-level module only.

Example

The following shows an example of a report when running the ANALYZE DESIGN command:

- When used for zero-delay simulation mismatch check:

```
LEC> ANALYZE DESIGN -REV -SEQ_INPUT_INTERACT_LOGICAL
Sequential gates in revised design with input interaction:
ID        Type      Sig-pair    Com-sups    Com-gates  Gate-name
================================================================================
19        DLAT      (D, CLK)    2           1          out_reg[0]/U$1/i0
20        DLAT      (D, CLK)    2           1          out_reg[6]/U$1/i0
```

  where

  ID: Gate ID

  Type: Specifies whether the gate is DFF or DLATCH;

  Sig-Pair: The two overlapping input signals. Possible signals that can appear include D(data), CLK(clock), S(set), R(reset);

  Com-sprts: Number of common supports;

  Com-gates: Number of common gates;

  Gate-name: DFF name.

- When used for clock gating structure check:

```
LEC> ANALYZE DESIGN -REV -CLOCK_GATING_STRUCTURE
DFFs in revised design with bad clock structure:
ID        DLAT-num    Gate-name
================================================================================
8         2           data_reg_reg/TI_LACTHC_UDP3/U$1
```

  where DLAT-num is the number of dlatches in the clock input cone.

ANALYZE EXPRESSION

```
ANAlyze EXPression
     <<module_name*> | -SUBSTITUTED MODULES>
     [-GOLden|-REVised]
     (Setup Mode)
```

Note: This requires a Conformal XL license.

Transforms a gate-level module into its equivalent word-level model. In the DW verification flow, `analyze expression` can replace the substituted generic-gate model with the word-level RTL model, then datapath analysis can be applied to avoid aborts in the comparisons. For a complex DW module with sub-modules, `analyze expression` replaces the sub-modules with their respective word-level RTL models.

Note: In the DW verification flow, you should use `analyze expression` immediately after the module substitution command `write blackbox wrapper`.

Tcl Command

```
analyze_expression
```

Parameters

`<module_name*>`	Specifies the name(s) of the module(s). This accepts wildcards.
`-SUBSTITUTED_MODULES`	Specifies that expression analysis is performed only on modules previously replaced by `write blackbox wrapper` command.
`-GOLden`	Specifies that expression analysis is performed on the Golden netlist. *This is the default*.
`-REVised`	Specifies that expression analysis is performed on the Revised netlist.

ANALYZE EXTENDED MAPPING

```
ANAlyze EXtended Mapping
     <filename1> <filename2>
     -Output <filename3>
     [-REPlace] [-GZIP] [-VERBose]
     [-INPUT_SPECification <file1_spec_string> <file2_spec_string>]
     [-OUTPUT_SPECification <output_spec_string>]
```

This command analyzes two extended mapping information to generate a file that helps to enable complex flows.

This command does not require any design to be read in and does not check the instance name with LEC DB. The analysis uses the instance names in the files as unique keys to perform text analysis.

Tcl Command

`analyze_extended_mapping`

Parameters

`<filename1>`	Specifies the emap file name for the comparison between design1 and design2.
`<filename2>`	Specifies the emap file name for the comparison between design2 and design3.
`-Output`	Specifies the output file name for the comparison between design1 and design3.
`-REPlace`	Replaces the output file if the file exists.
`-GZIP`	Writes out the output file in GZIP.
`-VERBose`	Prints out additional information.
`-INPUT_SPECification`	

Specifies the specification of the file1 and file2. Use `-` as a separator to provide two strings for each file. LEC uses these strings to define the relationship across the files.

For example, `-INPUT_SPECification A-B B-C` indicates that file1 is `A-to-B` comparison and file2 is `B-to-C` comparison. *This is default*.

If you specify `-INPUT_SPECification` you must also specify `-OUTPUT_SPECification`.

`-OUTPUT_SPECification`

Specifies the specification of the output. Use `-` as a separator to specify the labels for the file.

For example, `-OUTPUT_SPECification A-C` to generate the emap file for `A-to-C` comparison. This is default.

If you specify `-OUTPUT_SPECification` you must also specify `-INPUT_SPECification`.

Related Commands

READ EXTENDED MAPPING

WRITE EXTENDED MAPPING

ANALYZE GATE

```
ANAlyze GAte
     <[identifier* ...] | [-NONEQ] | [-ABORT] >
     [-FILE <file_name> [-REPLACE]]
     [-CHECK_CONSTant | -NOCHECK_CONSTant]
     [-GOLden | -REVised]
     [-VERBOSE]
     (LEC Mode)
```

Note: This is a Conformal XL command.

Analyzes gates in the design. This command reports all the messages associated with a specific gate and provides additional analysis messages.

For more information on this command, refer to the 14.1 web interface document (the web interface is enable through the set web interface command) titled *Advanced Message Reporting*.

Tcl Command

```
analyze_gate
```

Parameters

`<identifier*...>`	Specifies the gate IDs to analyze. Multiple gates can be specified and wildcards are supported.
`-NONEQ`	Selects all nonequivalent points.
`-ABORT`	Selects all abort points.
`-FILE <file_name>`	Outputs the formalized results to the specified file. If the file exists, Conformal appends the output; otherwise, Conformal creates a new file with the specified name. Results are in the JSON file format.
`-CHECK_CONSTant`	Checks if a gate is a sequential constant gate. *This is the default*.
`-NOCHECK_CONSTant`	Does not check if a gate is a sequential constant gate.
`-GOlden`	Specifies that the identifiers are from the Golden design.
`-REvised`	Specifies that the identifiers are from the Revised design.

`-VERBOSE`	Provide detailed information when reporting.

ANALYZE HIER_COMPARE

```
ANAlyze HIER_compare
     [-APPEND_String <string>]
     [-COMPARE_String <string>]
     [-CONDitional]
     [-DOFile <hier_dofile>][-REPlace]
     [-ECO_aware]
     [-EFFort <Medium | High>]
     [-EXact_pin_match | -NOEXact_pin_match]
     [-HIERarchical | -FLATten]
     [-INPUT_OUTPUT_Pin_equivalence]
     [-KEEP_TOP_level_constraints | -NOKEEP_TOP_level_constraints]
     [-LEVEL <integer>]
     [-MODULE <Golden_module> <Revised_module>]
     [-NOCONstraints | -CONstraints]
     [-NOEXACT_MODULE_match | -EXACT_MODULE_match]
     [-NOFUNCTION_Pin_mapping | -FUNCTION_Pin_mapping]
     [-PREPEND_String <string>]
     [-THReshold <integer>]
     [-USAge]
     [-VERbose]
```
(LEC Mode)

Note: This requires a Conformal XL license.

Analyzes the modules and their instantiations in LEC mode to generate the hierarchical dofile script that verifies the two hierarchical designs (starting from the lower-level modules, progressing to the top root module).

To enable hierarchical dofile generation in LEC mode, you must use the following command in Setup mode.

```
SETUP> set flatten model -enable_analyze_hier_compare
```

You cannot use this command when performing custom analysis of switch-level networks that were enabled through the `SET XC` command

Tcl Command

```
analyze_hier_compare
```

Parameters

`-APPEND_String <string>`	Appends any string of commands to the hierarchical dofile script *after* key point comparison for each module. Use the semi-colon character (;) to separate commands. Use double quotes to surround each command (see Examples).
`-COMPARE_String <string>`	Replaces the default compare command with a string of compare commands in the hierarchical dofile script generation for each module. Use the semi-colon character (;) to separate commands. Use double quotes to surround each compare command (see Examples).
`-CONDitional`	Skips blackboxing for nonequivalent submodules during the hierarchical comparison. (The end result is that Conformal flattens these submodules.) To report the flattened modules, use the `report hier_compare result -flattened` command.
`-DOFile <hier_dofile>`	Specifies the name of the hierarchical dofile script that verifies design hierarchy. Use the tilde character (~) to shorten the path of the file.
`-REPlace`	Replaces the existing file.
`-ECO_aware`	This option requires an ECO license. Recognizes ECO-related changes. This option recognizes ports that would otherwise be ignored for non-ECO comparisons, thus facilitating the correct comparison between the Golden and Revised design.
`-EFFort <Medium | High>`	Specifies the effort level for generating the hierarchical comparison dofile. `Medium`: Default effort level. `High`: Provides better analysis of constraint extraction, but can increase the analysis run time.
`-EXact_pin_match`	Writes only those modules with matching pin names to the hierarchical dofile script. *This is the default*.

`-NOEXact_pin_match`	Writes out even those modules which have mismatched pin names to the hierarchical dofile script.
`-HIERarchical`	Includes all of the specified module's submodules in the hierarchical dofile script. *This is the default*.
`-FLATten`	Flattens all of the specified module's submodules, which means that the submodules are not included in the hierarchical dofile script.
`-INPUT_OUTPUT_Pin_equivalence`	Extracts input-output pin equivalences within a module and applies them to the hierarchical dofile script. This can be used when the design has feedthroughs or feedback buffers.
`-KEEP_TOP_level_constraints`	Automatically applies top-level pathname-based constraints to the appropriate submodules, during hierarchical comparison. Top-level constraints are supported for `ADD PRIMARY INPUT`, `ADD PIN CONSTRAINTS`, and `ADD INSTANCE CONSTRAINTS` commands. *This option is the default*.

During hierarchical comparison, if top-level constraints take effect for the specific submodule, they are enclosed within the following messages:

```
Applying top-level pathname-based constraints
...
End of top-level pathname-based constraints
```

On the other hand, if the the pathname-based constraints do not take effect when they should have, the following warning message is generated:

```
// Warning: Top-level constraints might not have
been fully applied
```

See example below.

`-NOKEEP_TOP_level_constraints`	
`-LEVEL <integer>`	Writes all modules to the hierarchical dofile script up to the specified hierarchical level. The root module is at level 1.

	Disables the application of top-level pathname-based constraints to the appropriate submodules, during hierarchical comparison.
`-MODULE <Golden_module> <Revised_module>`	
	Writes the specified Golden module and Revised module to the hierarchical dofile script. (The module names can be different.) This option writes out the specified modules, even if the ports do not match.
`-NOCONstraint`	Do not extract or propagate the root module constraints and equivalences. *This is the default*.
`-CONstraint`	Extracts and propagates the root module constraints and equivalences and adds these to the corresponding modules in the hierarchical dofile script.
`-NOEXACT_MODULE_match`	Writes out module pairs, with matching instance path names (irrespective of the module name), to the hierarchical dofile script. *This is the default*.
`-EXACT_MODULE_match`	Writes out only the module pairs, with matching module names and instance path names, to the hierarchical dofile script.
`-NOFUNCTION_Pin_mapping`	Do not extract the functional mapping for the submodule boundary ports with the correct phase. *This is the default*.
`-FUNCTION_Pin_mapping`	Extracts the functional mapping for the submodule boundary ports with correct phase. This option is useful when the netlist has gone through clock-tree-synthesis (CTS), which creates additional submodule ports with different polarity; or, if there is an inverter push across submodule boundaries without modifying the port names.
`-PREPEND_String <string>`	Appends any string of commands to the hierarchical dofile script *before* key point comparison for each module. Use the semi-colon character (;) to separate commands. Use double quotes to surround each command (see "Examples").
`-THReshold <integer>`	Analyzes only modules with primitive instances greater than the threshold number. The default value of threshold is 50 primitives.

`-USAge`	Executes the USAGE command after each comparison and at the end of the hierarchical comparison.
`-VERbose`	Provides additional information when analyzing the modules for hierarchical comparison.

Example

```
LEC> analyze hier_compare -dofile hier.do -replace
```

```
LEC> analyze hier_compare -dofile hier.do -append_string "report compare data
-class nonequivalent" -prepend_string "report unmapped points -notmapped;
analyze setup" -replace
```

The following is a sample dofile that reads in the two hierarchical designs, writes out the hierarchical dofile script, and compares design hierarchies:

```
set log file hier.log -replace
read library golden.lib -verilog -golden
read design golden.v -verilog -golden
read library revised.lib -verilog -revised
read design revised.v -verilog -revised
set flatten model -enable_analyze_hier_compare
set system mode lec
analyze hier_compare -dofile hier.do -replace -constraints
dofile hier.do
exit -force
```

In the following example, top-level pathname-based constraints are applied to the appropriate submodules, during hierarchical comparison.

```
-----------------------------------------------------------------------
In dofile
-----------------------------------------------------------------------
add primary input a0/b0/scan_en -net -Golden
add pin constraint 0 a0/b0/scan_en  -Golden
analyze hier_compare -dofile hier.do -replace -constraints -
keep_top_level_constraints -noexact_pin_match
dofile hier.do
...
-----------------------------------------------------------------------
During hierarchical comparison; part of logfile
-----------------------------------------------------------------------
// Running Module modB and modB
// Command: set root module modB -Golden
// Command: set root module modB -Revised
// Command: set module property -instance /a0/b0 -Golden
// Command: set module property -instance /a0/b0 -Revised
// Command: report black box -NOHidden
// Command: set system mode lec
```

```
Applying top-level pathname-based constraints
// Command: add primary input scan_en -Golden
// Command: add pin constraints 0 scan_en -Golden
End of top-level pathname-based constraints
// Processing Golden ...
// Modeling Golden ...
```

Related Commands

ADD NOBLACK BOX

ADD MODULE ATTRIBUTE

DELETE NOBLACK BOX

DOFILE

READ DESIGN

READ LIBRARY

REPORT HIER_COMPARE RESULT

REPORT MODULE ATTRIBUTE

REPORT NOBLACK BOX

RESET HIER_COMPARE RESULT

RUN HIER_COMPARE

SAVE HIER_COMPARE RESULT

SET FLATTEN MODEL

SET NAMING RULE

UNIQUIFY

WRITE HIER_COMPARE DOFILE

ANALYZE IMPLICATION

```
ANAlyze IMplication
     [-ADD <-ONE | -1 | -ZERO | -0> <GateID...>]
     [-Block <GateID ...>]
     [-CHECK_Constant <GateID ...> ]
     [-CHECK_Redundancy <GateID ...>]
     [-DELete <GateID ...>]
     [-DEPTH <depth>]
     [-ONE | -1 <GateID ...>][-ZERO | -0 <GateID ...>]
     [-Golden | -Revised]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Analyzes implication values on the design. If you assign value(s) on certain gate(s), this command shows what the necessary values are on other gates to satisfy the assignment. It can also show if a gate has redundant fanin and if a gate is a constant gate.

The results are displayed in the schematic view with the following colors:

- Blue: initial assignments
- Green: current implication values
- Red: gates on the conflict path
- Purple: location where conflict occurred

In the schematic view, you can also right click the gate and set a value. Holding the mouse pointer on a gate, an information box will show if this gate has redundant fanin and if it is a constant gate.

Tcl Command

```
analyze_implication
```

Parameters

`-ADD`	Specifies that the following assignment(s) will be added into previous assignment(s).
`-Block`	Blocks the gate(s) so that implication will not go across it.
`-CHECK_Constant`	Checks the specified gate to see if it is constant gate.

`-CHECK_Redundancy`	Checks the specified gate to see if it has redundant fanins.
`-DELete`	Specifies that the following gate(s) will be removed from previous assignment(s).
`-DEPTH`	Specifies the logic depth beyond which implication will not be performed. The default value is 10.
`-ONE | -1`	Specifies that the following gate(s) will be assigned one.
`-ZERO | -0`	Specifies that the following gate(s) will be assigned zero.
`-Golden`	Specifies that the gate IDs are from the Golden design. *This is the default.*
`-Revised`	Specifies that the gate IDs are from the Revised design.

Related Commands

READ DESIGN

READ LIBRARY

ANALYZE LP_CONTROL PAIR

```
ANAlyze LP_control Pair
     [-Verbose]
     (Setup Mode)
```

Automatically sets up matched low-power control signals between Golden and Revised designs as compared pairs, and checks if there are any mismatched low-power control signals.

Users must enable `SET LOWPOWER OPTION -LP_CTRL_SIGNAL_COMPARED_PAIR` before `READ POWER INTENT` to use this command.

This command works after `READ POWER INTENT` for both designs.

Tcl Command

```
analyze_lp_control_pair
```

Parameters

`-Verbose`	Displays automatically paired low-power control signals and not-paired low-power control signals.

Related Commands

SET LOWPOWER OPTION

ADD LP_CONTROL IGNORED

ADD LP_CONTROL PAIR

REPORT LP_CONTROL VERIFICATION

REPORT LP_CONTROL IGNORED

REPORT LP_CONTROL PAIR

ANALYZE MODULE

```
ANAlyze MOdule
     <module_name*> ...
     [-BREAK_ABORT]
     [-BREAK_NONEQ]
     [-EFFORT <Low | Medium | High>]
     [-MSB_TRUNCATIONS | -NOMSB_TRUNCATIONS]
     [-LSB_TRUNCATIONS | -NOLSB_TRUNCATIONS]
     [-PARTIAL_SUM_OUTPUTS]
     [-REPLACE | -NOREPLACE]
     [-TMPDIR <string>]
     [-GOLden | -REVised]
     [-VERBose]
     (Setup Mode)
```

This command requires a Conformal XL license.

Analyzes non-exact synthesis modules in the Revised netlist and replaces the corresponding RTL models in the Golden netlist with the synthesized modules if the synthesized modules can be proven equivalent using the specified comparison mode. The method of module correspondence between the Golden and Revised netlist is identical to hierarchical comparisons. Therefore, module renaming rules and uniquification can also be applied.

For `analyze module` to be successful, the modules must be hierarchically comparable and must be included in the hierarchical dofile script generation. Do not use any commands that may affect these requirements (such as `ADD NOBLACK BOX`) before `ANALYZE MODULE`.

Caution

The ANALYZE MODULE command should be used after uniquification. After running this command, all design level settings cannot be modified. These settings include, but are not limited to, pin constraints and renaming rules.

Tcl Command

```
analyze_module
```

Parameters

`<module_name*>`	Specifies the name(s) of the module(s). This accepts wildcards. For DW02_tree modules, the module name must contain the keyword `tree`. For DW02_multp modules, the module name must contain the keyword `multp`. For DW_squarep modules, the module name must contain the keyword `squarep`. The tool also checks the module's input and output interfaces to ensure they meet the specifications of DW02_tree, DW02_multp, or DW_squarep. If the analysis of one or more specified modules fails (for example, if the specified module cannot be analyzed), this command returns an error code after analyzing all specified modules. The tool also sets the command's error flag of exit code.
`-BREAK_ABORT`	Returns an error code when it encounters an abort module. The tool also sets the command's error flag of exit code.
`-BREAK_NONEQ`	Returns an error code when it encounters a nonequivalent module. The tool also sets the command's error flag of exit code.
`-EFFORT <Low | Medium | High>`	
	Specifies the amount of effort applied to constraint extractions. `Low` applies minimal effort to constraint extraction. *This is the command default*. `Medium` applies greater effort to constraint extraction. `High` applies the maximum effort to constraint extraction.
`-MSB_TRUNCATIONS`	Truncates the most significant bits (MSBs) of partial sum outputs when adding up for comparison. This option is helpful in proving modules equivalent when such MSBs are unreachable. *This is the default*.
`-NOMSB_TRUNCATIONS`	Keeps the MSBs of partial sum outputs when adding up for comparison, even if the MSBs are unreachable.

`-LSB_TRUNCATIONS`	Truncates the least significant bits (LSBs) of partial sum outputs when adding up for comparison. This option is helpful in proving modules equivalent when such LSBs are unreachable. *This is the default*.
`-NOLSB_TRUNCATIONS`	Keeps the LSBs of partial sum outputs when adding up for comparison, even if the LSBs are unreachable.
`-PARTIAL_SUM_OUTPUTS`	Compares the modules with their partial sums added. This is applicable only to DesignWare/ChipWare models with partial sums output.
`-REPLACE`	Performs the replacement. *This is the default*.
`-NOREPLACE`	Do not perform the replacement.
`-TMPDIR <string>`	Specifies the location for the temporary directory that is created during the analysis of the module. If this option is not specified, `ANALYZE MODULE` uses the temporary directory created for the `SET PARALLEL OPTION` command; if it does not exist, a temporary directory is created in the current working directory. *Tip*: When possible, use the `SET PARALLEL OPTION` command to designate the location of the temporary directory so that both the `SET PARALLEL OPTION` and `ANALYZE MODULE` commands use the same directory. If you use the `ANALYZE MODULE -tmpdir` command, it specifies a temporary directory location that will be used by only the `ANALYZE MODULE` command.
`-GOLden`	Replaces the module in the Golden netlist with the module from the Revised netlist if successful. *This is the default*.
`-REVised`	Replaces the module in the Revised netlist with the module from the Golden netlist if successful.
`-VERBose`	Provides additional information for the analysis.

Example

The following command analyzes modules `DW02_multp` and `DW02_tree` in the Revised netlist and replaces the corresponding RTL models in the Golden netlist with the synthesized modules if the synthesized modules can be proven equivalent using the partial sums outputs comparison mode:

```
analyze module *DW02_multp* *DW02_tree* *DW_squarep* -partial_sum_outputs
```

Related Command

WRITE HIER_COMPARE DOFILE

ANALYZE MULTIPLIER

```
ANAlyze MUltiplier
     [-NOCDP_INFO | -CDP_INFO]
     (LEC Mode)
```

Initiates an analysis of multiplier modules. Based on the results of the analysis, Conformal can automatically resolve architecture mismatches and operand swapping problems. Additionally, use the `-cdp_info` option if you want Conformal to let you know when Conformal XL will be helpful.

Use this command after switching from Setup to LEC mode:

```
set system mode lec
...
analyze multiplier -cdp_info
add compared points -all
compare
```

Tcl Command

```
analyze_multiplier
```

Parameters

`-NOCDP_INFO`	Do not display a message when Conformal XL can enhance multiplier analysis. *This is the default.*
`-CDP_INFO`	Displays a message when Conformal XL can enhance multiplier analysis.

Related Commands

ANALYZE DATAPATH

ANALYZE MODULE

REPORT DATAPATH OPTION

REPORT DATAPATH RESOURCE

REPORT MULTIPLIER OPTION

ANALYZE NETLIST

```
ANAlyze NEtlist
     [-ABSTRACT [HFA | LIBCELL | MUXDFF]
     [-VERBose]
     [-GOLden | -REVised]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Provides another view of the flattened netlist, which can help in datapath analysis, comparison, and diagnosis.

This command can modify the structure of the flattened netlist.

As the same Boolean function can have different representations in the netlist, this command provides a uniform structural transformation on the netlist: without changing the function for which the netlist represents.

Note: You should use this command only on gate-level netlists.

Tcl Command

```
analyze_netlist
```

Parameters

`-ABSTRACT`	Abstracts the identified function blocks in the flattened netlist.
`HFA`	Abstracts the half-adder or full-adder function blocks in the flattened netlist.
`LIBCELL`	Identifies the library cells in the flattened netlist and re-synthesizes them with a simpler circuit representation. This option is useful for netlists library cells that are not optimized.
`MUXDFF`	Re-synthesizes the logic cone of the data input of a DFF, such that the logic cone is similar to the one shown in the following figure. This option is useful for retimed netlist and has been integrated in `ANALYZE RETIMING` command.

`-VERBose`	Provides additional information.
`-GOLden`	Specifies the netlist to be analyzed is the Golden netlist. *This is the default.*
`-REVised`	Specifies the netlist to be analyzed is the revised netlist.

Examples

The following command abstracts half-adder or full-adder function blocks in the revised netlist.

```
LEC> ANALYZE NETLIST -ABSTRACT HFA -REVISED
```

The following command identifies library cells used in the golden netlist and re-synthesizes them with a simpler circuit transformations.

```
LEC> ANALYZE NETLIST -ABSTRACT LIBCELL -GOLDEN
```

Related Command

ANALYZE DATAPATH

COMPARE

ANALYZE NONEQUIVALENT

```
ANAlyze NOnequivalent
     [ | <gate_id> | <instance_pathname*> ...
       [-Golden | -Revised] ]
     [-Type <PO | DFF | DLAT | BBOX | CUT > ...]
     [-DFT_Constraint_diagnosis]
     [-SOURCE_diagnosis]
     [-Summary | -Verbose]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Helps identify the possible causes of nonequivalent compared points.

Tcl Command

`analyze_nonequivalent`

Parameters

`<gate_id>`	Analyzes nonequivalent compared points for the specified gate.
`<instance_pathname*> ...`	Analyzes nonequivalent compared points for the specified instance.
`-TYPE`	Analyzes nonequivalent compared points with the specified type. `PO`: Primary output `DFF`: D flip-flop `DLAT`: D-latch `BBOX`: Blackbox `CUT`: Compared points with artificial gates to break combinational loops
`-DFT_Constraint_diagnosis`	Only analyzes the non-equivalent DFFs in the DFT. Incorrect or missing DFT constraints that cause the non-equivalence will be reported.

`-SOURCE_diagnosis`	Analyze nonequivalent compared points using diagnosis strategy.
`-Golden`	Analyzes nonequivalent compared points in the Golden design. *This is the default*.
`-Revised`	Analyzes nonequivalent compared points in the Revised design.
`-Summary`	Provides a summary of the analysis. *This is the default*.
`-Verbose`	Provides additional information for each individual nonequivalent compared point.

Examples

The following shows an example of a report when running the `ANALYZE NONEQUIVALENT` command. The lines in bold indicate the cause of the problems:

```
LEC> analyze noneq 213
//Command analyze noneq 213
Analyzing nonequivalent compared points:
  (G) + 213 DFF /wbs/hvlen_reg[28]
  (R) + 6277 DFF /wbs/hvlen_reg[28]/U$1
  The clock of DFF in Golden is not gated.
  The clock of DFF in Revised is gated.
Analysis of nonequivalent compared points:
  Gated clock of of DFF or DLAT. (Occurrence: 1)
  Unknown reason. (Occurrence: 1)
LEC> analyze noneq 170 -revised
//Command analyze noneq 170 -revised
Analyzing nonequivalent compared points:
  (G) + 167 PO /wbm_sel_o[0]
  (R) + 170 PO /wbm_sel_o[0]
  Following constraints may be necessary:
    Constant 1: (G) 1026 DFF /wbm/sel_o_reg[0]
Analysis of nonequivalent compared points:
  Sequential constant. (Occurrence: 1)
  Unknown reason. (Occurrence: 1)
```

Clock Gating

You can fix the first problem in the report:

```
  The clock of DFF in Golden is not gated.
  The clock of DFF in Revised is gated.
```

by running the following command in Setup mode:

```
set analyze option -auto
```

or the following command in LEC mode:

```
analyze setup
```

Sequential Constant

You can fix the second problem in the report:

```
  Following constraints may be necessary:
    Constant 1: (G) 1026 DFF /wbm/sel_o_reg[0]
```

by running auto analysis in Setup mode with the following commands:

```
set analyze option -auto
set flatten model -seq_constant
```

or the following command in LEC mode:

```
remodel -seq_constant
```

Related Command

ANALYZE SETUP

ANALYZE PARTITION

```
ANAlyze PArtition
     [-DATAPATH_TYPEs <<MUL | DIV | ADD | SUB> ...>]
     [-DOFile <filename>][-Replace]
     [-EFFORT [MEDIUM | HIGH]]
     [-GROUPs <id ...>]
     [-KEYPOINTs <identifier ...>]
     [-MAX_KEYPOINTs <integer>]
     [-MAX_PARTITIONs <integer>]
     [-NOABORT_STOP | -ABORT_STOP]
     [-NOINTERNAL | -INTERNAL]
     [-VERBose]
```
(LEC Mode)

This command requires a Conformal XL license.

This command derives partition groups by analyzing the datapath resources in the fanin-cones of not-compared or abort compared points. A partition group consists of three parts:

- A set of partition key points
- Partitions of the functional space of the partition key points
- A set of compared points suitable for the partitions.

Based on the derived partition groups, the tool can generate a dofile that performs functional partitioned comparisons (you can the -keypoints option of this command in place of the `ADD PARTITION KEY_POINT` and `WRITE PARTITION DOFILE` commands.

Tcl Command

```
analyze_partition
```

Parameters

`-DATAPATH_TYPEs <<MUL | DIV | ADD | SUB> ...>`

Specifies the type(s) of datapath resources to be analyzed for functional partitioning. Currently, the supported datapath types are `MUL` (multiplier), `DIV` (divider, modulus, and remainder), `ADD` (adder), and `SUB` (negator and subtractor). By default, all the above types of datapath resources are analyzed. This option has no effect if you use the `-keypoints` option.

`-DOFile <filename>`	Specifies the name of the partition dofile script that performs functional partitioned comparisons. Use the tilde character (~) to shorten the path of the file.
`-Replace`	Replaces the existing file.
`[-EFFORT [MEDIUM | HIGH]]`	Specifies the abort analysis effort level; `medium` is the default. Effort `high` enables abort analysis targeted at partitioned comparisons. For more information, refer to the 13.1 web interface article titled *Enhanced Functional Partition Execution*.
`-GROUPs <id ...>`	Specifies the partition groups that are used to generate the partition dofile script. If not specified, all the derived partition groups are used to generate the partition dofile script.
`-KEYPOINTs <identifier ...>`	Generates partitions based on the user-provided partition key points that are in the Golden netlist. You can use this option in place of the `ADD PARTITION KEY_POINT` and `WRITE PARTITION DOFILE` commands.
`-MAX_KEYPOINTs <integer>`	Specifies the maximum number of partition keypoints in a partition group. The default value is 8.
`-MAX_PARTITIONs <integer>`	Specifies the maximum number of partitions in a partition group. The default value is 256.
`-NOABORT_STOP`	Disables the adaptive partitioned comparison flow. *This is the default*.
`-ABORT_STOP`	Enables the adaptive partitioned comparison flow. In this flow, if there are aborts in a partitioned comparison, the rest of the partitioned comparisons are skipped. For more information, refer to the 13.1 web interface article titled *Enhanced Functional Partition Execution*.
`-NOINTERNAL`	Generates partitions based on key points. *This is the default*.
`-INTERNAL`	Generates partitions based on internal points and key points. Note that you can only run the dofile created by this option in the same LEC session.

`-VERbose`	Control the output information for resource sharing and derived partition groups.

Examples

The following illustrates various command line examples:

```
LEC> analyze partition -verbose
LEC> analyze partition -verbose -dofile partition.do -replace
LEC> analyze partition -dofile partition.do -max_keypoints 30 -max_partitions 1000
LEC> analyze partition -dofile partition.do -datapath_types mul div -replace
```

The following illustrates a sample script that performs functional partitioned comparisons:

```
add compared points -all
analyze partition -verbose -dofile partition.do -replace
dofile partition.do
```

Related Commands

ADD COMPARED POINTS

ADD PARTITION KEY_POINT

DOFILE

WRITE PARTITION DOFILE

ANALYZE POWER ASSOCIATION

```
ANAlyze POwer Association
     [-Module <module_name* ...>]
     [-OUT_Dofile <dofile_name> | -OUT_CPF <filename>]
     [-Golden | -Revised]
     [-REPlace]
     (Setup / Verify Mode)
```

Note: This requires a Conformal GXL license.

Analyzes the module's SPICE netlist and identifies the power/ground pin for each input and output pin with which it is associated.

Notes:

- The power/ground pin definition can come from LEF or SPICE.
- Run the `SET SPICE OPTION -NOBULK` command before reading in the SPICE design to maintain the connectivity of power/ground ports.
- Each input or output pin can have only one associated power/ground pin. Multiple power/ground pin association are ignored.

Tcl Command

`analyze_power_association`

Parameters

`-Module <module_name*> ...>`	Specifies the module(s) to be analyzed. Without this option, all modules in the design are analyzed. This accepts wildcards.
`-OUT_Dofile <dofile_name>`	Specifies the file to output the `ADD POWER ASSOCIATION` command.
`-OUT_CPF <filename>`	Specifies the output CPF file.
`-Golden`	Analyze the Golden design. *This is the default*.

`-Revised`	Analyze the Revised design.
`-REPlace`	Replaces the specified `-OUT_Dofile` or `-OUT_CPF` file if it already exists.

Related Commands

READ DESIGN `-spice`

SET SPICE OPTION

ANALYZE PROJECT

```
ANAlyze PRoject
     [-HIER_TO_FLAT
        [-VERbose <MAPPED_POINTS | UNMAPPED_POINTS | COMPARED_POINTS |
                  COMPARE_DATA [-CLASS <EQuivalent | INVequivalent
                  | NONEQuivalent | ABort | NOTcompared>]>]
        [-MAPPING_FILE <filename> [-REPlace]]
     ]
     (Setup/LEC)
```

Note: This command requires a Conformal XL license.

This command analyzes the information of multiple LEC runs for the specified LEC project. You must use the `SET PROJECT NAME` command before this command.

Tcl Command

`analyze_project`

Parameters

`-HIER_TO_FLAT`	Analyzes the hierarchical comparison information in the current run directory and generates a mapping and compare data report of the flattened run. This report is in the same format as the output of the `REPORT MAPPED POINTS -SUMMARY` and `REPORT COMPARE DATA -SUMMARY` commands. Applies to both Setup/LEC mode.
`-VERbose`	Displays a detailed report for the criteria.
`-CLASS`	Displays the specified class of compared points: ■ EQuivalent: Equivalent points ■ INVequivalent: Inverted-equivalent points ■ NONEQuivalent: Nonequivalent points ■ ABort: Aborted points ■ NOTcompared: All points not compared
`-MAPPING_FILE`	Writes mapped point information to a file.

`-REPlace`	Replace existing file.

Related Commands

DELETE PROJECT

SET PROJECT NAME

SET PROJECT OPTIONS

SET PROJECT PROPERTY

ANALYZE REDUNDANCY

```
ANAlyze REDundancy
     <[Identifier <-ONE | -ZERO | -UNREACH>] |
     [-Operand_isolation]>
     [-EFFORT <Low | Medium | High> ]
     [-Golden | -Revised]
     [-NOCommit | -Commit]
     [-Verbose]
     (LEC)
```

Analyzes the flattened netlist to find and resolve redundancy within signals and operand isolation cells. This command works for both hierarchical and flat comparisons.

For more information on how and when to use this command, launch the web interface using the `SET WEB_INTERFACE ON` command.

Tcl Command

`analyze_redundancy`

Parameters

`Identifier`	Specifies the analysis points. The identifier can be a gate, identification number, instance path, or pin path.
	Note: Identification numbers change from one version of Conformal to another. Always use the full path within your dofiles, especially if you are rerunning a design using a different tool version.
`-ONE`	Proves whether the specified locations have redundant value one.
`-ZERO`	Proves whether the specified locations have redundant value one.
	Note: If `-ONE`, `-ZERO`, or `-UNREACH` is not specified, the tool proves redundant value zero first, and then redundant value one.
`-UNREACH`	Proves whether the specified locations are unreachable to keypoints.

`-Operand_isolation`	Analyzes the netlist for redundancies in operand isolation cells. Operand isolation analysis should be done before datapath analysis. Note: This option will set `-commit`, and automatically analyze both the Golden and Revised designs. Currently, this option is supported for only RC synthesized netlists. In addition, this option can only be used on operation isolation cells that have been preserved in the synthesis netlist---this option does not support cells that have been ungrouped or that have gone through boundary optimization.
`-EFFORT`	Specifies the analysis effort level. Note: The `medium` and `high` levels provide better analysis, but can also increase the analysis run time.
`-Golden`	Specifies that the analyzed netlist is the Golden design.
`-REvised`	Specifies that the analyzed netlist is the Revised design.
`-Commit`	Remodels the signal with the proved redundant value. Note: This option is default when using `-Operand_isolation`.
`-Diagnosis`	Provides diagnosis information.
`-Verbose`	Provides verbose information.

Examples

- The following command sequentially checks the fanouts of a specified gate. Using this command, the tool finds that gate u2 has two fanouts with redundant value of one.

```
LEC> analyze redundancy -rev u2 -one
// Note: 2 net(s) have redundant value 1.
LEC> analyze redundancy -rev u2 -one -verbose
// Note: GATE 'u2' to GATE 'u5' has redundant value 1
// Note: GATE 'u2' to GATE 'u4' has redundant value 1
// Note: 2 net(s) have redundant value 1.
LEC> analyze redundancy -rev u2 -one -commit -verbose
// Note: GATE 'u2' to GATE 'u5' has redundant value 1
// Note: GATE 'u2' to GATE 'u4' has redundant value 1
// Note: 2 net(s) have redundant value 1 (committed).
```

- Operand isolation analysis should be done before datapath analysis. The following illustrates typical usage in the flow for invoking operand isolation analysis:

```
set system mode lec
analyze redundancy -operand_isolation -commit -revised
analyze datapath -module -verbose
analyze datapath -verbose
add compare point -all
compare
```

- The following performs redundancy analysis on the isolation cells in the revised netlist, and remodels the cells using the proved redundant values:

```
analyze redundancy -operand_isolation -commit -revised
```

- The following command reports whether operand isolation occurs in the synthesis netlist:

```
LEC> analyze redundancy -operand_isolation -verbose
// Operand Isolation Analysis Status
=================================================================
Design_Mod  Operand_Instance    Data_Width   Analysis_Quality
-----------------------------------------------------------------
top         RC_OI_HIER_INST10       16            100%
            RC_OI_HIER_INST11       16            100%
            RC_OI_HIER_INST12       16            100%
            RC_OI_HIER_INST9        16            100%
-----------------------------------------------------------------
Total isolation cells detected                       4
Total isolation cells analyzed successfully          4
Operand isolation analysis success rate           100%
=================================================================
```

Related Commands

ANALYZE ABORT

ANALYZE DATAPATH

ANALYZE RESULTS

```
ANAlyze RESults
     -LOgfiles <filename> ...
     [-Explain]
     [-GZIP_suffix <pattern> ...]
     [-Verbose]
     (Setup / LEC Mode)
```

Analyze results from LEC log files and produces a detailed summary. The report provides recommendation on suitable number of workers for parallelization of module comparison.

Tcl Command

`analyze_results`

Parameters

`-LOgfiles <filename> ...`	Specify log file names to perform results analysis.
`-Explain`	Provides more explanations in the analysis report.
`-GZIP_suffix <pattern> ...`	Specify one or more filename ending pattern to treat as gzipped files. *The default is ".gz"*.
`-Verbose`	Display all verbose messages.

Examples

The following sets an LEC log file and specifies that log file for results analysis at the end of the run.

```
set_log_file -replace lec_run.log
usage -auto -elapse
...
analyze_results -logfile lec_run.log
exit
```

Related Commands

SET LOG FILE

RUN HIER_COMPARE

GO HIER COMPARE

ANALYZE RETIMING

```
ANAlyze REtiming
     [-ABSTRACT_MUXDFF | -NOABSTRACT_MUXDFF]
     [-DIAGNOSIS [<identifier>] [-BACKWARD] ]
     [-GENERAL [<identifier* ...> [-BACKWARD]] ]
     [-MERGE | -NOMERGE]
     [-PIPELINE [<identifier* ...>] [-BACKWARD] [-FIXNAMEmapped][-REPEAT] ]
     [-PREDICT <identifier> [-BACKWARD] ]
     [-VERBose]
     [-BOth | -GOLden | -REVised]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Initiates pipeline retiming, retiming for Genus synthesized netlist, or retiming for designs that are combinationally equivalent. Normally, you use this command to retime a Revised design to match a referenced Golden design. If this command is successful, you can use the `COMPARE` command to ensure that the two designs are equivalent. If this command is unsuccessful, then the original retiming you performed is incorrect.

For more guidelines and examples, see the "Advanced Capabilities" chapter in the *Conformal Equivalence Checking User Guide*.

Tcl Command

`analyze_retiming`

Parameters

`-ABSTRACT_MUXDFF`	Moves registers along with their extracted synchronous enables. *This is the default.*
`-NOABSTRACT_MUXDFF`	Do not move registers along with their extracted synchronous enables.

`-DIAGNOSIS [<identifier>] -BACKWARD`

Checks if the design is pipeline retimed. Use `<identifier>` to check if a particular register can be retimed a step forward or backward. If the retime movement cannot succeed, the reason for the failure is reported. By default, this diagnoses the forward retiming step.

Note: This option will not change the netlist. It only provides information about the specified retiming step.

`-BACKWARD` specifies that the backward retiming step is diagnosed.

`-GENERAL <identifier* ...> -BACKWARD`

Retimes registers in the design to state points. Use this option to specify the state points or use the added sequential corresponding points as the state points.

You can specify an `<identifier>` as the state point(s) or use the added sequential corresponding points as the state points.

Use the `-BACKWARD` option to move registers backward to the specified state points. By default, registers are moved forward to the specified state points.

`-MERGE`	Specifies that equivalent registers are merged after registers are moved, including inverted-equivalent registers. This helps to reduce the unmapped register key points and the resulting false nonequivalences. *This is the default.*
`-NOMERGE`	Disables the merging of equivalent registers after they are moved.

`-PIPELINE <identifier* ...>`

Moves all registers to the primary output side of the design as much as possible, or use the `<identifier>` option to specify one or more registers, separated by a space. This accepts wildcards.

`-BACKWARD` moves registers backward to the primary input side of the design as much as possible. With this option, pipeline backward retiming can be performed on either all registers or a selected set of registers.

`-FIXNAMEmapped` moves only registers that are not mapped by name. During retiming analysis, registers that are mapped by name are fixed in their original position.

`-REPEAT` will continue to move registers, even after the internal retiming movement limit is reached.

`-PREDICT <identifier* ...> -BACKWARD`

Predicts if registers can be retimed to the state point specified by `<identifier>`.

Use the `-BACKWARD` option to predict backward retiming. By default, this predicts forward retiming.

`-VERbose`	Prints additional information, including a list of current tasks and statistics.
`-BOTh`	Executes this command on both the Golden and Revised designs. *This is the default.*
`-REVised`	Executes this command on the Revised design and uses the Golden design as the reference design.
`-GOLden`	Executes this command on the Golden design and uses the Revised design as the reference design.

Examples

- The following command initiates backward pipeline retiming for register `r1` and registers whose identifiers begin with `r2`, such as `r2a` and `r21`, in the Revised design:

```
analyze retiming -pipeline r1 r2* -revised -backward
```

- The following command manually retimes to the specified state points:

```
analyze retiming -general retime_state_point* -backward -verbose
```

Related Commands

ADD SEQ_CORRESPONDENCE

ADD COMPARED POINTS

ADD MODULE ATTRIBUTE

COMPARE

DELETE SEQ_CORRESPONDENCE

REPORT SEQ_CORRESPONDENCE

SET RETIMING OPTION

ANALYZE SEQUENTIAL CONSTANTS

```
ANAlyze SEQuential Constants
     [-AUTO]
     [-ALLOW_ADDMAP]
     [-COMPare]
     [-RESTORE_COMPARE | -NORESTORE_COMPARE]
     (LEC / Setup Mode)
```

Compare the original trigger function of registers which are remodeled into sequential constants by LEC.

Their mapping is based on original mapping and name mapping. Note that the compared netlists during sequential constants' comparison are not the final netlist remodeled by LEC. Besides, for this kind of comparison, few remodeling behaviors will be modified.

Tcl Command

```
analyze_sequential_constants
```

Parameters

`-AUTO`	Automatically runs sequential constants' comparison after analyze_setup. For this kind of comparison, it may affect some remodeling. This option needs to be executed in setup mode.
`-ALLOW_ADDMAP`	Allow that LEC automatically adds extra related mapping for comparing sequential constants. Notice it may bring extra remodeling and comparing in regular flow. Default is not to allow.
`-COMPARE`	Manually compares sequential constants in LEC mode. It needs to turn on -`AUTO` first.
`-RESTORE_COMPARE`	Restore to normal compare mode after comparing sequential constants. *This is the default*.
`-NORESTORE_COMPARE`	Do not restore to normal compare mode. Users need to use this option with `-COMPARE` for diagnosing. After diagnosing, use `-RESTORE_COMPARE` to restore to normal compare mode.

Related Commands

ANALYZE SETUP

```
ANAlyze SEtup
     [-EFFORT <Medium | High | Ultra>]
     [-CUT | -NOCUT]
     [-DELETE_UNREACHable | -NODELETE_UNREACHable]
     [-GATED_Clock | -NOGATED_Clock]
     [-LATCH_TRANSPARENT | -NOLATCH_TRANSPARENT]
     [-NOABSTRACT_GATED_CLOCK | -ABSTRACT_GATED_CLOCK]
     [-NOFORCE | -FORCE]
     [-NOLATCH_NO_HOLDING | -LATCH_NO_HOLDING]
     [-NOLIBRARY_VERIFICATION | -LIBRARY_VERIFICATION]
     [-NOMODIFY_MAP | -MODIFY_MAP]
     [-NOPHASE_MAPPING | -PHASE_MAPPING]
     [-NOREPORT_MAP | -REPORT_MAP]
     [-REPORT_RUNTIME]
     [-SEQ_MERGE | -NOSEQ_MERGE]
     [-SEQ_REDUNDANT | -NOSEQ_REDUNDANT]
     [-TRANSFORM_SET_DOMINANT | -NOTRANSFORM_SET_DOMINANT]
     [-VERBose]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Analyzes the netlists and sets up the flattened design for accurate comparison. This helps avoid false nonequivalences. This command can also analyze and remodel the following commonly-encountered setup issues: sequential constants, sequential merging, loop cutting, clock gating, and phase mapping.

Note: To resolve sequential constant optimization with this command, you must use the following command:

```
set flatten model -seq_constant
```

Tcl Command

```
analyze_setup
```

Parameters

`-EFFORT <Medium | High | Ultra>`

Specifies the effort level for automatic setup.

Medium is the default.

`-CUT`	Analyzes loop cutting. When creating flattened netlists, the Conformal software breaks all loops by inserting CUT gates, which could cause false non equivalences. With this option, you can resolve these false non equivalences. *This is the default.*
`-NOCUT`	Do not analyze loop cutting.
`-DELETE_UNREACHable`	Deletes functionally mapped unreachables if they are non-equivalent. *This is the default*.
`-NODELETE_UNREACHABLE`	Do not delete functionally mapped unreachables.
`-GATED_Clock`	Remodels gated-clock logic of the clock port of a DFF. *This is the default.*
`-NOGATED_Clock`	Do not remodel gated-clock sequential instances.
`-LATCH_TRANSPARENT`	Converts transparent D-latches into buffers. *This is the default*.
`-NOLATCH_TRANSPARENT`	Automatic analysis will not convert transparent D-latches into buffers.
`-NOABSTRACT_GATED_CLOCK`	Do not abstract gated-clock. *This is the default*.
`-ABSTRACT_GATED_CLOCK`	Analyzes the complex gated-clock structure and abstracts it into a simple, general structure.
`-NOFORCE`	`ANALYZE SETUP` is skipped when it is called after all mapped points are added as compared points, and all compared points are either equivalent or abort. *This is the default*.
`-FORCE`	Forces the tool to run `ANALYZE SETUP`, regardless of the compare status.
`-NOLATCH_NO_HOLDING`	Do not remodel latches that do not have holding functions. *This is the default*
`-LATCH_NO_HOLDING`	Remodel latches that do not have holding functions.
`-NOLIBRARY_VERIFICATION`	Do not perform setup analysis for library cell verification. *This is the default.*
`-LIBRARY_VERIFICATION`	Performs setup analysis for library cell verification.
`-NOMODIFY_MAP`	Do not modify mapping. *This is the default.*

`-MODIFY_MAP`	Synthesis optimizations, such as gated clocks or sequential constants, can sometimes lead to the alteration of the original functional mapping. This command enables LEC to automatically modify these mappings after remodeling.
`-NOPHASE_MAPPING`	Disables phase adjustment for DFF/D-LATCH mapping during analyze setup. *This is the default.*
`-PHASE_MAPPING`	Enables phase adjustment for DFF/D-LATCH mapping during analyze setup. Note: If the design has an inverter push, use `SET MAPPING METHOD -phase` for the first mapping (prior to `SET SYSTEM MODE lec`) because correct mapping is critical for modeling. Executing `SET MAPPING METHOD -phase` after entering LEC mode cannot guarantee that all mapping issues will be resolved.
`-NOREPORT_MAP`	Do not report the mapping results after completing automatic analysis. *This is the default.*
`-REPORT_MAP`	Reports the mapping results after completing automatic analysis.
`-REPORT_RUNTIME`	Reports the runtime usage of each remodeling stage in the command.
`-SEQ_MERGE`	Enables the merge of DFF/D-LATCH. *This is the default*.
`-SEQ_REDUNDANT`	Automatic analysis removes redundant fan-out gates from DFFs and DLATs. *This is the default*.
`-TRANSFORM_SET_DOMINANT`	Automatic analysis to balance the structure of set-dominant DFFs and DLATs during automatic setup. *This is the default*.
`-NOTRANSFORM_SET_DOMINANT`	Disables structure balancing of set-dominant DFFs and DLATs during automatic setup.
`-NOSEQ_REDUNDANT`	Automatic analysis will not remove redundant fan-out gates from DFFs and DLATs.
`-NOSEQ_MERGE`	Disables the merge of DFF/D-LATCH.
`-VERBose`	Provides additional information.

Related Commands

ANALYZE NONEQUIVALENT

SET ANALYZE OPTION

ANALYZE SEQUENTIAL DUPLICATION

```
ANAlyze SEQuential DUPlication
     [-WRITE_LIST <filename> [-REPlace]]
     [-FILE <filename> [-ALL | -FEEDback | -CONVergence]]
     [-REVised | -GOLden]
     [-VERBose]
     (LEC Mode)
```

This command performs two checks for sequential duplication sets in the netlist.

- Feedback check: duplicated registers have no functional dependence on other registers within the duplication set.
- Convergence check: any comparison point depends on, at most, one register in the duplication set.

Tcl Command

```
analyze_sequential_duplication
```

Parameters

`-WRITE_LIST <filename>`	Write the merge points to the specified file.
`-REPlace`	Replaces the existing file.
`-FILE <filename>`	Specifies the duplication set.
`-ALL`	Perform both feedback and convergence validation. *This is the default*.
`-FEEDback`	Perform feedback validation.
`-CONVergence`	Perform convergence validation.
`-REVised`	Indicates revised design to validate. *This is the default*.
`-GOLden`	Indicates Golden design to validate.
`-VERBose`	Provides verbose information.

Related Commands

ANALYZE X

```
ANAlyze X
     [ -ALL | <identifiers> | -Source <filename> <line> <column> ]
     [-Level <num> ]
     [ -Verbose ] [ -FILE <annotation filename> ] [-REPlace]
     [ -APPLY <annotation filename> [-NOIFDEF] ]
     (LEC Mode)
```

This command requires an XL license.

This command analyzes X assignments in the golden RTL design to find corresponding functions in the revised gate-level design. Replacing X assignments with these functions maintains the equivalence between RTL and gate-level designs.

This command can automatically write out analyzed functions into an annotation file and apply them to the RTL design to replace X assignments. Applied RTL should be in the same location recorded in the annotation file.

In the ECO flow, replacing X assignments in the new RTL (R2) can help to reduce synthesis differences in G2 and can help achieve a smaller ECO patch.

Note: The golden flattened netlist will be changed after calling this command to analyze X-assignments.

Tcl Command

`analyze_x`

Parameters

`-ALL`	Analyze all X assignments in the golden RTL design.

`<identifiers>`	Analyze X assignments for the specified don't care gates. The identifiers of gates can be the instance path name or identification number. The identifiers can be an ID or instance path name. For example, a gate with an ID of 89 or gate-name of `/top/n44[0]` (as reported by `REPORT GATE`) can be specified as either of the following:
	`analyze x 89`
	`analyze x /top/n44[0]`

`-Source <`*`filename`*`> <`*`line`*`> <`*`column`*`>`	
	Analyze X assignments in the specified locations of the RTL source file. If omitting the column, all X assignments in the specified line of file are targets. If omitting the line and column, all X assignments in the specified file are targets.
`-Level <`*`num`*`>`	Sets limitation of traversed circuit levels for analyzing functions and expressions. If level is 0, the possible analyzed functions will be limited to 0, 1, or a net. If level is 1, it allows 1 operator in analyzed functions. Higher level takes long runtime and higher success rate. Default is 14 according to performance trade-off.
`-Verbose`	Displays a report of the analyzed expressions.
`-FILE <`*`annotation filename`*`>`	
	Write out expressions for analyzed X assignments with their location in the RTL source file.
`-REPlace`	Replaces any existing annotation files with the same name.
`-APPLY <`*`annotation filename`*`>`	

Apply the specified annotation file to replace X assignments that have the same locations as the ones specified in the annotation file.

Note: Although original files are kept (renamed from "*.v" into "*.v~"), it is recommended that you manually back up your RTL files before applying any annotation files.

The applied code will replace X with its analyzed expression in "`` `ifdef LEC_ANAX_FUNCTION``" and keep original line in "`` `else``", such as the followings:

```
`ifdef LEC_ANAX_FUNCTION

  o = { a & b, 1'b1 };

`else

  o = 2'bxx;

`endif
```

`-NOIFDEF`	Replaces X assignments directly (no `` `ifdef``)

Examples

For example, given RTL "`o = s ? 1'bx : a;`" and a gate-level design "`or ( o, s, a );`", you can:

- Use the following commands in LEC mode to get analyzed functions of X assignments.

```
LEC> analyze x -all -verbose -file xfunction.json
// Remodel 1 X assignments
(test.v:4,16) X is converted to:
{ 1'b1 }
```

- Use the following commands to apply the annotation file "`xfunction.json`" to replace X-assignments in RTL code automatically:

```
LEC> analyze x -apply xfunction.json
// Succeed to apply X in (test.v:4,16)
```

Where the RTL code becomes the following:

```
`ifdef LEC_ANAX_FUNCTION
 assign o = s ? { 1'b1 } : a;
  `else
   assign o = s ? 1'bx : a;
  `endif
```

APPLY GUIDED TRANSFORMATIONS

```
APPly GUided Transformations
     [-EFFort < Low | Medium | High | SUper | ULtra>]
     [-Module <module_name ...>]
     (Setup Mode)
```

Applies the datapath transformations, recorded in the OVF (Open Verification Format) guide file(s), to the LEC Golden design database that is created from RTL. For an overview of the OVF flow, refer to the *Interfacing between RTL Compiler and Conformal User Guide* (this document is available within the RTL Compiler document set). This command is a part of the RC-LEC Scripted Verification Flow, where the transformations are used as a strategy to resolve aborts. For more information on this flow, refer to the 13.1 web interface documented *Scripted RC Verification.*

To commit the transformations recorded in the guide file(s), LEC can create the Golden design database with a higher structural similarity to the netlist synthesized by the synthesis tool, which may help resolve abort points or enhance comparison performance.

By default, this command validates all transformations before committing them to the Golden design database. This ensures that the Golden design functionality must not divert from the original interpretation of the Golden RTL design.

You can use the `-module` option to selectively choose one or more modules to apply the transformation: instead of the entire design.

Tcl Command

```
apply_guided_transformations
```

Parameters

`-EFFort < Low | Medium | High | SUper | ULtra>`	
	Specifies the effort for validating transformations. Default is `LOW`.
`-Module <module_name ...>`	
	Specifies the modules for which to apply transformations. If this option is not specified, all transformations for the entire design might be applied.

Related Commands

READ GUIDE FILE

REPORT GUIDE INFORMATION

ASSIGN PIN DIRECTION

```
ASSign PIn Direction
     <IN | OUT | IO>
     <module_name*>
     <pin_name* ... >
     [-FROM_DIr <IN | OUT | IO>]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Defines a module pin's direction. SPICE netlist ports do not have direction, unless you supply `*.pininfo <pin>:<direction>` as a CDL comment and read it in as an inout. Abstraction analyzes the circuit and assigns pin direction: when determinable. In some cases, you need to assign pin direction manually to complete abstraction.

Note: You can use this command instead of the `ADD MOS DIRECTION` command to assist abstraction.

Wildcard: The wildcard (*) represents any zero or more characters in module/pin names.

Tcl Command

```
assign_pin_direction
```

Parameters

`IN`	Specifies that the assigned pin direction is input.
`OUT`	Specifies that the assigned pin direction is output.
`IO`	Specifies that the assigned pin direction is I/O.
`<module_name*>`	Specifies that the pin resides in the specified module. To specify all modules, use `*`.
`<pin_name*... >`	Assigns a direction to the specified pin. Specifies that the assigned direction of all pins in module `clka` is input.
`-FROM_DIr`	Only pins which have the direction specified by this option's argument will be redirected.
`-Golden`	Assigns the pin direction to the Golden design. *This is the default.*

`-Revised`	Assigns the pin direction to the Revised design.
`-Both`	Assigns the pin direction to both the Golden and Revised designs.

Examples

```
assign pin direction in mux2p sela -revised
assign pin direction out mux4p y -golden
assign pin direction in mux2p sela y -both
//Assigns direction to pins sela and y.
assign pin direction OUT * VDD
//Specifies that the assigned direction of pins named VDD in all modules is output.
assign pin direction IN mod pin_* -from_dir IO
//Changes all IO pins whose name matches 'pin_*' in
//module 'mod' to IN pins on the golden side.
```

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

READ PATTERN

REPORT CLOCK

REPORT MOS DIRECTION

REPORT NET ATTRIBUTE

REPORT PIN DIRECTION

RESOLVE

BACKWARD

```
BACkward
     [<integer>]
     (LEC Mode)
```

Reports fanin gate information from the currently displayed flattened gate information. The fanin gate you choose with this command becomes the current flattened gate. Use this command to trace gates in place of repeatedly using the `REPORT GATE` command.

Note: This command does not report gates at the design level.

Tcl Command

```
backward
```

Parameters

`<integer>`	Specifies which fanin gate is reported. The value 1 denotes the first fanin. *The default value is 1.*

Related Commands

FORWARD

REPORT GATE

BREAK

```
BREak
        (Setup / LEC Mode)
```

Terminates the dofile script and returns you to the system mode prompt.

Related Commands

CONTINUE

DOFILE

SET DOFILE ABORT

CHANGE GATE TYPE

```
CHAnge GAte Type
    <identifier>
    [-Type <gate_type>]
    [-Help]
    [-Golden | -Revised]
    (LEC Mode)
```

Changes the gate type of a selected object in the schematic viewer.

Note: You cannot use this command on pins and nets.

Note: If you invoke the Flatten Schematics window for a candidate gate (such as *Corresponding Supports* and *Compared Points*) from the Diagnosis Manager, you cannot change its gate type.

Tcl Command

```
change_gate_type
```

Parameters

`<identifier>`	Specifies the gate ID or instance pathname.
	If you do not specify one of the following options, Conformal automatically determines if the identifier is a number or a path. In the case of a number, Conformal uses the `-id` option; otherwise, Conformal searches for the gate with the `-instance`, `-pin`, or `-net` option; in this respective order.
`-Type <gate_type>`	Changes the specified gate type, which can be `AND`, `NAND`, `OR`, `NOR`, `XOR`, `XNOR`, `BUF`, or `INV`.
	Note: Not all types are available for all objects
`-Help`	Returns a list of potential candidate gate types that can be applied.
`-Golden`	Specifies that the identifier is in the Golden design. *This is the default.*
`-Revised`	Specifies that the identifier is in the Revised design.

Related Commands

REPORT GATE

CHANGE NAME

```
CHAnge NAme
     <filename>
     [-Summary | -Verbose]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Converts netlist net names, port names, and cell names back to their original names. Thus, Conformal does key point mapping faster and more efficiently. The most common use of this command is to change the names in a post synthesis netlist back to their original, pre-synthesis forms.

After Conformal reads in the file containing the original names and new names, it makes the conversion. Generally, the synthesis tool you have used generates the file describing the changes. Consult the specific vendor's tool documentation for additional change name information.

The format of the change name file is as follows:

- The change name information is preceded by dashes.
  Note the dashes in the example below that separate column headers and the change name information. Conformal recognizes this file as a change name file format when it detects the dashes.
- The "Design" column consists of module names.
  For example, see `mod0_module` in the sample Change Name file below.
- The "Type" column consists of elements that belong to the specified design (that is, cell, port, and net).
- The "Object" is the original name. Conformal will change the name back from the "New Name" to the original name.
- The "New Name" is the name assigned by the synthesis tool

```
Design           Type  Object                     New Name
----------------------------------------------------------
mod0_module      port  port0_input1               port0_1
mod0_module       net   net0_output                net0_t
mod1_module      port  port1_input1a              port_1a
mod1_module       cell  net1_subinstantiation      net1_n
```

Tcl Command

`change_name`

Parameters

`<filename>`	Specifies the file that contains name changes.
`-Summary`	Prints out the summary count of the number of name changes. *This is the default.*
`-Verbose`	Prints out a message for each name change.
`-Golden`	Changes names in the Golden design. *This is the default.*
`-Revised`	Changes names in the Revised design.
`-Both`	Changes names in both the Golden and Revised designs.

Related Commands

ADD RENAMING RULE

DELETE RENAMING RULE

REPORT RENAMING RULE

CHECK LOWPOWER CELLS

```
CHEck LOwpower Cells
```
(*LEC Mode*)

Note: This is a Conformal Low Power command.

This command performs two checks, both of which apply to all mapped state elements (DFF and DLAT). Power domains are compared; if they are not equivalent, the design may not be functionally equivalent. State retention strategy is also compared; sometimes, differing retention strategies are applied when the Golden and Revised power intent is different.

Use this command after keypoint mapping.

Use the `REPORT LOWPOWER DATA -verbose` command to diagnose failures.

Tcl Command

```
check_lowpower_cells
```

Related Commands

COMMIT POWER INTENT

READ POWER INTENT

REPORT LOWPOWER CELLS

REPORT LOWPOWER DATA

SET LOWPOWER OPTION

CHECK MAPPING SETUP

```
CHEck MApping Setup
     [-ALIAS]
     [-CASE_sensitive]
     [-MAPPING_FILE_QUALITY]
     [-SUMMARY]
     [-MAPped | -UNMAPped | -VALID | -INVALID]
     [-ERROR <number>]
     (Setup/LEC Mode)
```

This command reports the quality of the mapping file without having to go into LEC mode. This command should be invoked after elaborating both designs and reading the mapping file with `set_analyze_option -mapping_file`.

Tcl Command

```
check_mapping_setup
```

Parameters

`-ALIAS`	Checks and reports the quality of alias mapping. Name alias are specified using the `ADD NAME ALIAS` command.
`-CASE_sensitive`	Checks if the keypoints mapping are case sensitive.
`-MAPPING_FILE_QUALITY`	Reports the quality of mapping file for mapping two designs. All valid, invalid, mapped, and unmapped data are reported in default.
`-SUMMARY`	Instructs the command to print only a summary report. otherwise, by default, a detailed listing of the names in the mapping file will be reported.
`-MAPPED`	Instructs the command to report only names that can be mapped by the file.
`-UNMAPPED`	Reports the points that cannot be mapped by the file. Use this option to reduce the size of the report.
`-VALID | -INVALID`	Reports the valid/invalid names.

`-ERROR <number>`	This option will error out and stop to run dofile when the total number of invalid entries in the mapping file is more than or equal to the number <number>.

Related Command

SET ANALYZE OPTION

CHECK VERIFICATION INFORMATION

```
CHEck VErification Information
     [-IMPLEMENTATION | -VERIFICATION]
     [-TYPE MERGE | CONSTant | ALL]
     [-VERbose]
     (Setup/LEC Mode)
```

This command checks the guild information quality, contains verification information and implementation information.

Tcl Command

`check_verification_information`

Parameters

`-IMPLEMENTATION`	
	Checks the implementation information. The implementation is read by `READ IMPLEMENTATION INFORMATION`.
`-VERIFICATION`	Checks the verification information. The implementation is set by `SET IMPLEMENTATION INFORMATION`.
`-TYPE`	Specifies the type of information that would be checked. The default is `ALL`.
`-VERBOSE`	Displays the detailed information of the check result..

Related Command

SET IMPLEMENTATION

READ IMPLEMENTATION INFORMATION

CHANGE NET TYPE

```
CHAnge NEt Type
     <TRI | TRI0 | TRI1 | TRIREG | TRIAND | TRIOR>
     <net_name>
     [-Module <module_name>]
     [-Golden | -Revised]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Modifies the database so that it appears that the changed net types were declared in the original Verilog netlist.

Tcl Command

`change_net_type`

Parameters

`TRI`	Specifies a tristate net type. *This is the default*.
`TRI0`	Specifies a net of type `TRI0`.
`TRI1`	Specifies a net of type `TRI1`.
`TRIREG`	Specifies a net of type `TRIREG`.
`TRIAND`	Specifies a net of type `TRIAND`.
`TRIOR`	Specifies a net of type `TRIOR`.
`-Module <module_name>`	Specifies the module name of the net to be changed. By default, this command changes the net in the root module.

Related Command

ADD NET ATTRIBUTE

CHECKPOINT

```
CHECKPOINT
     <checkpoint_file_name>
     [-PROTECT <password>]
     [-REPlace]
     (Setup/LEC Mode)
```

Creates a *checkpoint* file. A checkpoint file contains a complete snapshot of the Conformal run up until the `CHECKPOINT` command is issued. The tool preserves the entire memory contents of the Conformal run. The checkpoint file includes:

- Hierarchical and flattened databases
- Environment settings
- Constraints
- Verification results
- User-defined variables
- User-defined procedures

To restart a session using a checkpoint file, invoke Conformal using the `-RESTART_CHECKPOINT <checkpoint_file_name> [-protect <password>]` option. For example:

```
lec -restart_checkpoint mycpfile -protect mypass
```

Review the set of limitations at the end of this section before using the `CHECKPOINT` command.

For more information on the checkpoint and restart facility, refer to "Checkpoint and Restart Facility" in the *Conformal Equivalence Checking User Guide*.

Tcl Command

```
checkpoint
```

Parameters

checkpoint_file_name	Name of the checkpoint file.

`-protect <password>`	Encrypts the checkpoint file with the specified password. The password cannot lead with a dash ("-"); for example, `-mypassword` is not allowed.
`-REPlace`	Removes any existing file of the same name and replaces them with the specified file.

Limitations

- Checkpoint and restart works on only Linux, and only on the following platforms: 64-bit Linux kernel versions 2.6.9-34, 2.6.9-42, 2.6.9-67, 2.6.9-78, 2.6.9-89, 2.6.10, 2.6.14, 2.6.16, 2.6.18, 2.6.25, 2.6.26, 2.6.27, 2.6.29.4, 3.0.13 and 3.0.101.
- Checkpoint files should be created and restarted using the same machine.

  The tool issues warnings when you try to restart a checkpoint file from a different machine. The warning message can be ignored if the restart is always successful. For example, in a homogenous virtual machine computation environment, the host name might be different, but the OS version and configuration are the same. Thus, the warning message can be ignored.

  If you are creating a checkpoint file that you plan to restart using a different license server, add the restart license server to the LM_LICENSE_FILE variable before invoking Conformal and before creating the checkpoint file; otherwise, you will not be able to restart the checkpoint file with the new server. For example:

  ```
  setenv LM_LICENSE_FILE "$LM_LICENSE_FILE":5280@mylic01
  ```
- Do not enter the GUI mode if you plan to create a checkpoint file that you will want to run later in the GUI mode.

  If a checkpoint file is created after having entered GUI mode, when the checkpoint file is restarted, it will restart and run in non-GUI mode and the GUI mode is disabled. If a checkpoint file is created before entering the GUI mode, the checkpoint file can enter the GUI mode when it is restarted.

  The Conformal `-restart_checkpoint` option tries to restore the process to the same status as when it was checkpointed. If a file cannot be re-opened during restart (permission problems, for instance), Conformal issues an error and opens it at `/dev/null` with only read permissions. You can restart again in the correct directory, or ask the file owner to change the permissions and try restarting the checkpoint file.

  ```
  <Conformal_tool> -restart_checkpoint [-protect <password>]
  ```
- You cannot specify the stack limit in a restarted tool process. You can, however, specify the stack limit when you issue the `CHECKPOINT` command:

```
CHECKPOINT -stack <multiplier>
```

Default multiplier is 1 (in other words, 64MB).

Related Commands

INFO CHECKPOINT

CLOSE SCHEMATICS

```
CLOse SChematics
```
    (*Setup / LEC Mode*)

Closes all schematic viewer windows.

Note: You cannot use this command in non-GUI mode.

Related Command

OPEN SCHEMATICS

COMMIT POWER INTENT

```
COMmit POwer Intent
     [-INSERT_ISOLATION]
     [-GOLden | -REVised | -BOTH]
     (Setup Mode)
```

Note: This is a Conformal Low Power command and is for the CPF flow only.

Applies power intent.

Tcl Command

```
commit_power_intent
```

Parameters

`-INSERT_ISOLATION`	Inserts isolation cells for all isolation rules that are inserted for equivalence checking. Level shifters, switch cells, and retention are not inserted. **Note:** This is for the CPF flow only.
`-GOLden`	Inserts low power cells in the Golden design. *This is the default.*
`-REVised`	Inserts low power cells in the Revised design.
`-BOTH`	Inserts low power cells in both the Golden and Revised designs.

Related Command

COMMIT POWER INTENT

READ POWER INTENT

COMPARE

```
COMpare
     [-EFFORT < Low | Medium | High | Auto | LIght | COMPlete>]
     [-ABORT_Print]
     [-ABORT_Stop <integer>]
     [-GATE_TO_GATE]
     [-NONEQ_Print]
     [-NONEQ_Stop <integer>]
     [-Random [<integer>]]
     [-SIngle]
     [-THREADS <integer>[,<integer>]]
     [-TIMEstamp]
     [-TURBO | -NOTURBO]
     [-Verbose]
     (LEC Mode)
```

Starts the equivalency checking comparison between the Golden and Revised designs on the added compared points. During the comparison, the following information is displayed:

- Progress percentile number, which displays the completion rate
- Running count, which displays the number of key points that have been compared along with the total number of non-equivalent key points

Each compared point results in a status drawn from the following five possibilities:

- Equivalent
- Inverted equivalent
- Nonequivalent
- Abort
- Not compared

When Conformal completes the comparison, it displays a summary table of the number of equivalent and nonequivalent compared points.

Notes:

- If you must interrupt the comparison, the `Control-C` keys stop the process.
- By default, when the clock port is disabled, the data cone will not be compared. The software might perform modeling that disables the clock port, so the disabled clock might not necessarily mean the clock in original netlist is disabled.

- For datapath intensive designs, run time can take longer when the `ANALYZE DATAPATH` command is not used before the comparison.

Tcl Command

`compare`

Parameters

`-EFFORT`	Specifies the comparison for this compare command. This supersedes the `SET COMPARE EFFORT` setting. `Low`: Applies low effort to equivalency checking for each gate. *This is the default.* `Medium`: Applies greater effort to equivalency checking for each gate. `High`: Applies the maximum effort to equivalency checking for each gate. `Auto`: Starts with low effort and automatically increases the compare effort if abort points are present in the design. `LIght`: Applies minimal effort to equivalency checking for each gate. `COMPlete`: Performs equivalency checking for each gate until the comparison results in an EQ or NONEQ result. With this option, the tool never returns an abort (in other words, if EQ or noneq is not returned, the compare will go on indefinitely).
`-ABORT_Print`	Displays the abort points as they are found.
`-ABORT_Stop <integer>`	Stops the comparison after finding the specified number of abort points.
`-GATE_TO_GATE`	Enables an algorithm that might improve the run time of large gate-to-gate netlist comparisons.
`-NONEQ_Print`	Displays the nonequivalent points as they are found.

`-NONEQ_Stop <integer>`	Stops the comparison after finding the specified number of nonequivalent points.
`-Random <integer>`	Performs random-based simulation on the compare points, where *`integer`* specifies the number of random patterns to use. The default is 1600.
`-SIngle`	Compares each key point as a single point. By default, the `COMPARE` command compares by key point groups.
`-THREADS <integer>[,<integer>]`	Specifies the minimum and maximum number of compare threads. If only one number entered, this specifies both the minimum and maximum number of threads. For example, '`-threads 2`' specifies two threads; '`-thread 2,4`' specifies a minimum of two threads, and a maximum of four threads. This supersedes the `SET PARALLEL OPTION -threads` setting.
`-TIMEstamp`	Displays the system time (HH:MM format) of the last compare status update.
`-TURBO`	Specifies using parallel turbo threads.
`-NOTURBO`	Specifies without using parallel turbo threads. *This is the default*.
-Verbose	Displays detailed comparison information, such as the circuit duplication in multithreaded comparison.

Examples

The following is a set of sample commands that shows this and related commands in context. The following set of commands assumes that you have read in your library, design, and have switched to the LEC mode.

1. Use the `ADD COMPARED POINTS` command to add mapped points to the compare list.

```
LEC> add compare point -all
//2 compared points added to compare list
```

2. Use the `COMPARE` command to start the equivalency comparison between the Golden and Revised designs.

```
LEC> compare
========================================================================
Compared points      PO        Total
```

```
----------------------------------------------------------------------------
Equivalent          1         1
----------------------------------------------------------------------------
Nonequivalent       1         1
============================================================================
```

3. Use the `REPORT COMPARE DATA` command to report the nonequivalent points.

```
LEC> rep comp data -noneq

Compared points are: Nonequivalent
+ 11  PO   /x                     + 11  PO   /x

1 compared point(s) reported
============================================================================
Compared points      PO        Total
----------------------------------------------------------------------------
Equivalent          1         1
----------------------------------------------------------------------------
Nonequivalent       1         1
============================================================================
```

4. Use the `ADD DYNAMIC CONSTRAINT` command to add a dynamic constraint to the design.

```
LEC> add dyn con 0 /c -gold

LEC> add dyn con 0 /c -rev
```

5. Use the `REPORT DYNAMIC CONSTRAINTS` to report the dynamic constraints in the design.

```
LEC> report dynamic constraints

============================================================================
Design     ID     Type     Value    Name
----------------------------------------------------------------------------
Golden     3      PI       0        /c
Revised    3      PI       0        /c
============================================================================
```

6. Use the `PROVE` command to show whether the specified gates are equivalent or not equivalent.

```
LEC> prove /x /x

//Compared points are: Nonequivalent
//(G) + 11  PO   /x
//(R) + 11  PO   /x
```

7. Use the `DIAGNOSE` command to diagnose the nonequivalent points.

```
LEC> diag /x

//Diagnosis for Nonequivalent key points:
//(G) + 11  PO   /x
//(R) + 11  PO   /x


The diagnosis point can be corrected by changing the following gates:
============================================================================
Correction      ID (R)  Type          Name
----------------------------------------------------------------------------
DEL_INVERTER    24      INV           /gextra
```

8. Use the `REPORT MESSAGE` command to show the message related to the comparison.

```
LEC> report message -compare -verb

// Warning: 1 DFFs/DLATs have 1 disabled clock port: skipped data cone comparison

  (G) + 4   DLAT /z_reg - skipped data cone z_reg with corresponding clock cone z_reg

  (R) + 4   DLAT /z_reg - skipped data cone z_reg with corresponding clock cone z_reg
```

Related Commands

ADD COMPARED POINTS

ANALYZE DATAPATH

DELETE COMPARED POINTS

DIAGNOSE

PROVE

REPORT COMPARE DATA

REPORT COMPARED POINTS

RUN PARALLEL COMPARE

SET COMPARE EFFORT

SET COMPARE OPTIONS

USAGE

COMPARE LIBERTY

```
COMpare LIberty
     (Setup/LEC Mode)
```

Description

Note: This is a Conformal Low Power command.

Compares the Golden Liberty against the revised Liberty.

The following lists some of the over 70 attributes compared by this command:

- is_macro_cell
- pg_type
- switch_function
- pg_function
- voltage_name
- std_cell_main_rail
- related_power_pin
- related_ground_pin
- related_bias_pin
- is_isolated
- power_down_function
- switch_pin
- is_analog
- isolation_enable_condition
- antenna_diode_type
- antenna_related_power_pin
- antenna_related_ground_pin

Use this command after you have read in the Liberty library. No design needs to be read to use this command.

Tcl Command

`compare_liberty`

Example

The following is a sample dofile that reads in two Liberty libraries and compares them:

```
read library -liberty gold_sample.lib -golden -lp
read library -liberty revs_sample.lib -revised -lp
compare liberty
report compared liberty -verbose
```

After you read in the libraries, use the `COMPARE LIBERTY` command to compare the two libraries, and the `REPORT COMPARED LIBERTY` to view the details of the comparison.

Related Commands

DELETE LIBERTY_COMPARE FILTER

REPORT COMPARED LIBERTY

REPORT LIBERTY_COMPARE FILTER

COMPARE POWER CONSISTENCY

```
COMpare POwer Consistency
     <The option is 1801 flow only)
     [-EXCLUDE_ICG_mapped_pair_retention_strategy_compare]
     [-EXCLUDE_DIODE_CLAMP_CELL]
     (LEC Mode)
```

Note: This is a Conformal Low Power command and is for the 1801 flow. For CPF flow, this command is the same as "check lowpower cells".

Checks the consistency of the supply set and applied retention strategy of mapped state elements (DFF and DLAT) between Golden and Revised. Specifically, power domains are compared; if they are not equivalent, the design may not be functionally equivalent. State retention strategy is also compared for the part of retention power and ground supplies which determine the power domain assignment. Different retention strategies can be applied when the Golden and Revised power intent are different.

Use this command after keypoint mapping.

Use the `REPORT POWER CONSISTENCY` -verbose command to diagnose failures.

Tcl Command

```
compare_power_consistency
```

Parameters

`-EXCLUDE_ICG_mapped_pair_retention_strategy_compare`	
	Exclude the mapping state element pair at the retention strategy comparison and the retention supply set comparison when either one side element has the integrated clock gating attribute.
`-EXCLUDE_DIODE_CLAMP_CELL`	
	Exclude the mapping black box pair at the supply set comparison when the black box instances in both side contain pins with `antenna_diode` attribute.

Related Commands

READ POWER INTENT

REPORT POWER CONSISTENCY

COMPARE POWER GRID

```
COMpare POwer Grid
     [-INCLUDE <HIER_IN_MAPPED_BBOX_MACRO>]
     [-EXCLUDE <[UNMAPPED_SIMPLE_DOMAIN_ELEMENT | [UNMAPPED_AON_BUF_INV_SUPPLY] |
     [UNMAPPED_LEVEL_SHIFTER_SUPPLY] | [NONEQ_DIODE_CLAMP_CELL_SUPPLY] |
     [NONEQ_PHY_CELL_SUPPLY]>]
     (Setup Mode)
```

Note: This is a Conformal Low Power command and an 1801 feature.

Compares the supply network connectivity between the golden design and the revised design. The comparison starts from each supply source and ensures that a supply network is implemented consistently with the corresponding supply network on the other side, where two supply networks are considered corresponded if they are driven or partially driven by the supply sources that are mapped to each other by name. The supply network comparison results in a status with the following three possibilities:

- Equivalent: The two supply networks are mapped and are driven by the same supply sources
- Nonequivalent: There is at least one supply point of the supply network whose driving supply source is different than that of the mapped supply point.
- Inconclusive: There is at least one supply point of the supply network that does not have a corresponding mapped supply point in the other design, while all of the mapped supply points are driven by the same supply source.

Tcl Command

```
compare_power_grid
```

Parameters

`-INCLUDE`	Specifies types of supply points that need to be compared and reported. `HIER_IN_MAPPED_BBOX_MACRO`: Supply points under a hierarchy mapped to a black-box. If not specified, they will be ignored by default.

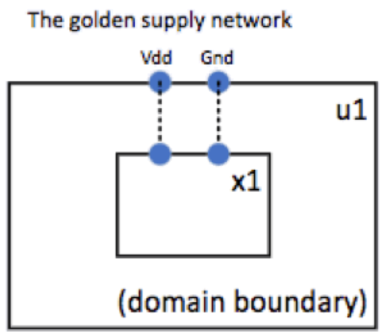
`-EXCLUDE`	Specifies types of supply points to be ignored in compare power grid and its report. `UNMAPPED_SIMPLE_DOMAIN_ELEMENT`: Unmapped supply ports of a simple domain element. A simple domain element is an effective element (an instance in the netlist) of a power domain, which has only supply pins physically or virtually connected to its power domain's primary supply set. This option filter the supply points if such domain while the driving supply points at the domain boundary are still compared. `UNMAPPED_AON_BUF_INV_SUPPLY`: Unmapped real hierarchical supply ports which supply only always on buffer and inverter cells. `UNMAPPED_LEVEL_SHIFTER_SUPPLY`: Unmapped real hierarchical supply ports which supply only level shifter cells. `NONEQ_DIODE_CLAMP_CELL_SUPPLY`: Diode clamp cell supply ports whose driving supplies are different to the driving supplies of the mapped diode clamp cell ports. `NONEQ_PHY_CELL_SUPPLY`: Physical cell supply ports whose driving supplies are different to the driving supplies of the mapped physical cell ports.

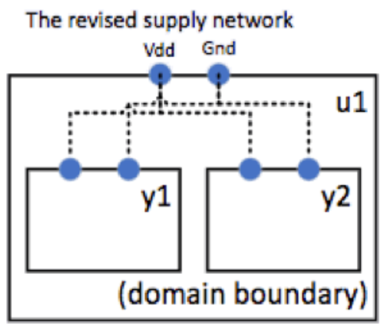
Examples

To filter the supply points that are implemented and only used for always-on buffers/inverters insertions.

```
TCL_SETUP > compare_power_grid -exclude unmapped_aon_buf_inv_supply
```

In the below example, u1 is the domain boundary instance. u1/y1 and u1/y2 of the revised supply network and u1/x1 of the golden supply network are simple domain elements because their supply sources are implied or implemented by the automatic connections from the domain, which can be filtered with `-exclude_unmapped_simple_domain_element` option.





Related Commands

ADD MAPPED INSTANCE

READ KEYPOINT MAPPING

SET EQUIVALENT SUPPLY_SOURCES

REPORT COMPARED POWER_GRID

ADD IGNORED GRID

DELETE IGNORED GRID

REPORT IGNORED GRID

COMPARE POWER INTENT

```
COMPare POwer Intent
     [-COMPARE_DESIGN_OBJECTS [ALL | <command_name*>]]
     [-ALL | -DESign | -LIBrary | -MACRO]
     (Setup/LEC Mode)
```

This is a Conformal Low Power command. Compares the Golden power intent specification against the revised power intent specification.

Design objects that are specified in the power intent can be renamed during synthesis and place-and route causing subsequent attempts to verify the design data using the original power specification to fail. Implementation tools are then used to write out revised power intent using the new design object names. Power intent comparison is necessary to ensure that the revised power intent has not changed the intention of the original power intent specification.

Note: The two power intent files must be of the same format type (either CPF or UPF).

The following lists some of the power intent aspects compared by this command:

- Nominal conditions
- Library sets and Liberty file names
- Power modes
- Analysis views and SDC file names

Note: This command does not compare design objects such as instance names and pin names, because these can be changed by the implementation tools.

Use this command after you have read in the library, design data, library power intent, and Golden and Revised design power intent. You can use this command before or after you commit the power intent. See example section for a sample dofile.

Tcl Command

```
compare_power_intent
```

Parameters

`-COMPARE_DESIGN_OBJECTS`	Force the comparison of design objects. By default, only top-level ports are compared. `ALL`: Compare the design objects of all commands. *This is the default*. `<command_name*>`: Compare design objects for the specified command.
`-ALL`	Compares library power intent and design power intent between the Golden and revised files. Library power intent is read in using the `READ LIBRARY` command; design power intent is read in using the `READ POWER INTENT` command. *This is the default*.
`-DESign`	Compare only the design power intent.
`-LIBrary`	Compare only the library power intent.
`-MACRO`	Compare only the macro power intent.

Example

For example, to compare design objects of only connect commands:

```
compare power intent -compare_design_objects connect_commands
```

For example, to compare design objects of all commands:

```
compare power intent -compare_design_objects
```

For an additional example, see REPORT COMPARED INTENT.

Related Commands

READ DESIGN

READ LIBRARY

READ POWER INTENT

COMPARE POWER INTENT

REPORT COMPARED INTENT

CONTINUE

`CONTinue`

(Setup / LEC Mode)

Used in conjunction with the `BREAK` command in a dofile, when a dofile executes the `BREAK` command, Conformal issues a warning and prompts you to use the `CONTINUE` command. The `CONTINUE` command has no effect if you type it without being prompted by Conformal.

The `CONTINUE` command supports mixed GUI and non-GUI mode. For example, you can run a dofile in non-GUI mode, encounter a break in the dofile, issue `set gui on`, and run `continue` from the GUI. The same applies when you break in the GUI mode: switch to command mode and enter `continue`.

Conformal also supports nested breaks inside dofiles, working in a stack fashion. For example, when you type in `continue` from a lower-level dofile, Conformal proceeds until it encounters the next `BREAK` command.

Example

```
//Warning: Break dofile 'my_dofile' at line 32. Use 'continue' command to continue.
LEC> continue
```

Related Commands

BREAK

DOFILE

SET DOFILE ABORT

COPY MODULE

```
COPy MOdule
     <[-Golden | -Revised] <source_module_name>
        [-Revised | -Golden] <target_module_name>>
     [-LOGIC | -PINDIR]
     [-USE_RENAME_RULE | -NOUSE_RENAME_RULE]
     (Setup Mode)
```

Copies the logic or pin direction from a source module in one design to a target module in the other design. If you must copy both the logic *and* the pin direction, use two separate commands.

Tcl Command

`copy_module`

Parameters

`-Golden`	Specifies that the source module is located in the Golden design. *This is the default.*
`-Revised`	Specifies that the source module is located in the Revised design.
`<source_module_name>`	Specifies the name of the source module to be copied.
`-Revised`	Specifies that the target module is located in the Revised design. *This is the default.*
`-Golden`	Specifies that the target module is located in the Golden design.
`<target_module_name>`	Specifies the name of the target module for the copy operation.
`-LOGIC`	Copies the logic from the source module into the target module. *This is the default.* If you must copy both logic and pin direction, use two separate `COPY MODULE` commands.
`-PINDIR`	Copies the pin direction from the source module into the target module. If you must copy both logic and pin direction, use two separate `Copy Module` commands.

`-USE_RENAME_RULE`	Uses renaming rules for matching pin and module names. *This is the default.*
`-NOUSE_RENAME_RULE`	Do not use renaming rules for matching pin and module names.

Related Command

ASSIGN PIN DIRECTION

DELETE ALIAS

```
DELete ALias
     <aliasname* ...>
     (Setup / LEC Mode)
```

Deletes aliases created with the `ADD ALIAS` command. Use the `REPORT ALIAS` command to display a list of all aliases.

Wildcard: The wildcard (*) represents any zero or more characters in alias names.

Tcl Command

`delete_alias`

Parameters

`<aliasname* ...>`	Deletes the specified aliases.

Related Commands

ADD ALIAS

REPORT ALIAS

DELETE BLACK BOX

```
DELete BLack Box
     <<blackbox_name*> ... [-Module] | <blackbox_name> ... -Instance | -All>
     [-HIER]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes specified blackboxes from the design. These blackboxes were either created with the `ADD BLACK BOX` command or were a part of the original design.

Use the `REPORT BLACK BOX` command to display a list of all blackboxes.

Wildcard: The wildcard (*) represents any zero or more characters in module names.

Tcl Command

```
delete_black_box
```

Parameters

`<blackbox_name*> ... -Module`	Deletes blackboxes specified by this list. `-module` *is the default.*
`<blackbox_name> ... -Instance`	Specifies that the blackbox names are instance names.
`-All`	Deletes "all" defined blackboxes. `-All` applies within the given defaults.
`-HIER`	When you generate a hierarchical dofile using the `WRITE HIER DOFILE` command, this option is automatically added by the tool. It deletes the modules that were blackboxed by the tool during hierarchical compare.
`-Golden`	Deletes blackboxes from the Golden design. *This is the default.*
`-Revised`	Deletes blackboxes from the Revised design.
`-Both`	Deletes blackboxes from both the Golden and Revised designs.

Related Commands

ADD BLACK BOX

REPORT BLACK BOX

DELETE CLOCK

```
DELete CLock
     <-ALL | net_name...> [-Module <module_name>]
     [-Golden | -Revised]
     (Setup Mode)
```

Deletes clocks added with the `ADD CLOCK` command.

Use the `REPORT CLOCK` command to display a list of all aliases.

Tcl Command

```
delete_clock
```

Parameters

`-ALL`	Deletes all the defined clocks within the given defaults.
`<net_name ...>`	Deletes the net(s) that were defined as clock(s) and specified in this list.
`-Module <module_name>`	Specifies that the defined clock pin is located in this module.
`-Golden`	Deletes the clock(s) from the Golden design. *This is the default.*
`-Revised`	Deletes the clock(s) from the Revised design.

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE MOS DIRECTION

DELETE COMPARED POINTS

```
DELete COmpared Points
     < -All |<<gate_id> | <instance_pathname*> | <pin_pathname*> ...
        [-Golden |-Revised]>
     < -CLASS <EQuivalent | INVequivalent | NONequivalent | ABort | NOTcompared>>
     >
     (LEC Mode)
```

Deletes compared points originally added with the `ADD COMPARED POINTS` command. If the compared point is deleted from the Golden design, Conformal also deletes its mapped compared point from the Revised design. Alternately, if the compared point is deleted from the Revised design, Conformal also deletes its mapped compared point from the Golden design.

Use the `REPORT COMPARED POINTS` command to display a list of all added compared points.

Wildcard: The wildcard (*) represents any zero or more characters in instance and pin paths.

Tcl Command

```
delete_compared_points
```

Parameters

`-All`	Deletes all compare points within the given defaults.
`<gate_id>`	Deletes the specified gates as compare points. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`<instance_pathname*>`	Deletes the specified instance paths as compare points.
`<pin_pathname*>`	Deletes the specified pin paths as compare points.
`-Golden`	Deletes the compare points from the Golden design. *This is the default.*
`-Revised`	Deletes the compare points from the Revised design.

`-CLASS`	Deletes the specified class of compared points Equivalent: Equivalent points INVequivalent: Inverted-equivalent points NONequivalent: Nonequivalent points ABort: Aborted points NOTcompared: All points not compared

Related Commands

ADD COMPARED POINTS

COMPARE

REPORT COMPARED POINTS

DELETE CUT POINT

```
DELete CUt Point
     <-All | <pathname ...>>
     [-Net | -Pin]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes cut points originally added with the `ADD CUT POINT` command.

Use the `REPORT CUT POINT` command to display a list of all added cut points.

Tcl Command

```
delete_cut_point
```

Parameters

`-All`	Deletes all cut points within the given defaults.
`<pathname ...>`	Deletes cut points from the specified paths.
`-Net`	Specifies that the named path is a net. *This is the default.*
`-Pin`	Specifies that the named path is a pin.
`-Golden`	Deletes the cut points from the Golden design. *This is the default.*
`-Revised`	Deletes the cut points from the Revised design.
`-Both`	Deletes the cut points from both the Golden and Revised designs.

Related Commands

ADD CUT POINT

REPORT CUT POINT

REPORT PATH

DELETE DONTTOUCH REGISTERS

```
DELete DOnttouch Registers
     <register_pathname* ...>
     [-SEQ_Constant | -SEQ_Merge]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes sequential constant and/or sequential merge don't-touch registers to Golden and/or Revised designs which were specified with the `add_donttouch_registers` command.

Tcl Command

```
delete_donttouch_registers
```

Parameters

`<register_pathname* ...>`	
`-seq_constant`	Deletes sequential constant don't-touch registers.
`-seq_merge`	Deletes sequential merge don't-touch registers.
`-Golden`	Deletes don't-touch registers to the Golden design.
`-Revised`	Deletes don't-touch registers to the Revised design.
`-Both`	Deletes don't-touch registers to both the Golden and Revised designs.

Related Commands

ADD DONTTOUCH REGISTERS

REPORT DONTTOUCH REGISTERS

DELETE DYNAMIC CONSTRAINTS

```
DELete DYnamic Constraints
     <-All | <identifier ...>>
     [-INStance | -Pin | -Net | -ID]
     [-Golden | -Revised | -Both]
     (LEC Mode)
```

Deletes dynamic constraints originally added with the `ADD DYNAMIC CONSTRAINTS` command.

Use the `REPORT DYNAMIC CONSTRAINTS` command to display a list of all added dynamic constraints.

Tcl Command

`delete_dynamic_constraints`

Parameters

`-All`	Deletes all dynamic constraints within the given defaults.
`<identifier ...>`	Deletes dynamic constraints from the specified identifier. If you do not specify one of the following options, Conformal automatically determines if the identifier is a number or a path. In the case of a number, Conformal uses the `-id` option; otherwise, Conformal searches for the gate with the `-instance`, `-pin`, or `-net` option; in this respective order.
`-INStance`	Specifies the hierarchical instance path. *This is the default.*
`-Pin`	Specifies the pin path, which is the module instance name concatenated with the pin name.
`-Net`	Specifies the net path, which is the instance name concatenated with the net name.
`-ID`	Specifies the identification number (ID) of a gate. The identification number is an integer assigned automatically by Conformal. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.

`-Golden`	Deletes dynamic constraints from the Golden design. *This is the default.*
`-Revised`	Deletes dynamic constraints from the Revised design.
`-Both`	Deletes dynamic constraints from both the Golden and Revised designs.

Related Commands

ADD DYNAMIC CONSTRAINTS

PROVE

REPORT DYNAMIC CONSTRAINTS

DELETE IGNORED GRID

```
DELete IGnored Grid
     <-Golden | -Revised | -Both>
     <-SUPply <supply_point_name> | -MODule <module_name> | -INStance
     <instance_name> | -ALL>
     (Setup Mode)
```

Note: This is a Conformal Low Power command and an 1801 feature.

Delete ignored port, instance, or module added by command `ADD IGnored Grid`.

Tcl Command

`delete_ignored_grid`

Parameters

`-Golden`	Deletes the ignored supply point of Golden netlist.
`-Revised`	Deletes the ignored supply point of Revised netlist.
`-Both`	Deletes the ignored supply points of both Golden netlist and Revised netlist.
`-SUPply <supply_point_name>`	Specifies an ignored supply object, which is a supply pin in the design netlist or a virtual supply port created in the 1801 power intent.
`-MODule <module_name>`	Specifies an ignored module.
`-INStance <instance_name>`	Specifies an ignored design instance.
-ALL	Deletes all ignored supply points

Related Commands

COMPARE POWER GRID

REPORT COMPARED POWER_GRID

ADD IGNORED GRID

DELETE IGNORED INPUTS

```
DELete IGnored Inputs
     <-ALL_Pin | <primary_pin* ...>>
     [-ROot |-Module <module_name> | -ALL_Module]
     [-Golden | -REvised | -Both]
     (Setup Mode)
```

Deletes input pins originally added as ignored inputs in the Golden or Revised design with the `ADD IGNORED INPUTS` command.

Use the `REPORT IGNORED INPUTS` command to display a list of all added ignored input pins.

Wildcard: The wildcard (*) represents any zero or more characters in ignored input names.

Tcl Command

```
delete_ignored_inputs
```

Parameters

`-ALL_Pin`	Deletes all previously added ignored inputs within the given defaults.
`<primary_pin* ...>`	Deletes the specified pins as ignored inputs. This accepts wildcards.
`-ROot`	Deletes the ignored inputs in the root module. *This is the default.*
`-Module <module_name>`	Deletes the ignored inputs from the specified module.
`-ALL_Module`	Deletes the ignored inputs from all of the modules, including the root module.
`-Golden`	Deletes the specified ignored inputs from the Golden design. *This is the default.*
`-REvised`	Deletes the specified ignored inputs from the Revised design.
`-Both`	Deletes the specified ignored inputs from both the Golden and Revised designs.

Related Commands

ADD IGNORED INPUTS

REPORT IGNORED INPUTS

DELETE IGNORED OUTPUTS

```
DELete IGnored Outputs
     <-ALL_Pin | <primary_pin* ...>>
     [-ROot | -Module <module_name> | -ALL_Module]
     [-Golden | -REvised | -Both]
     (Setup Mode)
```

Deletes output or I/O pins originally added as ignored outputs with the `ADD IGNORED OUTPUTS` command.

Use the `REPORT IGNORED OUTPUTS` command to display a list of all added ignored output or I/O pins.

Wildcard: The wildcard (*) represents any zero or more characters in ignored output names.

Tcl Command

```
delete_ignored_outputs
```

Parameters

`-ALL_Pin`	Deletes all added ignored outputs within the given defaults.
`<primary_pin* ...>`	Deletes the specified pins as ignored outputs. This accepts wildcards.
`-ROot`	Deletes the ignored outputs in the root module. *This is the default.*
`-Module <module_name>`	Deletes the ignored outputs from the specified module.
`-ALL_Module`	Deletes the ignored outputs from all modules, including the root module.
`-Golden`	Deletes the specified ignored outputs from the Golden design. *This is the default.*
`-REvised`	Deletes the specified ignored outputs from the Revised design.
`-Both`	Deletes the specified ignored outputs from both the Golden and Revised designs.

Related Commands

ADD IGNORED OUTPUTS

REPORT IGNORED OUTPUTS

DELETE INSTANCE ATTRIBUTE

```
DELete INstance Attribute
     <module_name> <instance_name> <WEAK>
     [-Golden | -Revised]
     (Setup Mode)
```

Deletes instance attributes originally added with the `ADD INSTANCE ATTRIBUTE` command.

Use the `REPORT INSTANCE ATTRIBUTE` command to display a list of all added instance attributes.

Tcl Command

```
delete_instance_attribute
```

Parameters

`<module_name>`	Deletes the instance attribute from the specified module.
`<instance_name>`	Deletes the instance attribute from the specified instance.
`<WEAK>`	Specifies drive strength.
	Note: This option applies to Conformal GXL.
`-Golden`	Deletes the instance attribute from the Golden design. *This is the default.*
`-Revised`	Deletes the instance attribute from the Revised design.

Related Commands

ADD INSTANCE ATTRIBUTE

REPORT INSTANCE ATTRIBUTE

DELETE INSTANCE CONSTRAINTS

```
DELete INstance Constraints
     <<name ...> [-Module <module_name* ...> ] | -All>
     [-Golden | -Revised | -BOTH]
     (Setup Mode)
```

Deletes instance constraints originally added with the `ADD INSTANCE CONSTRAINTS` command.

Use the `REPORT INSTANCE CONSTRAINTS` command to display a list of all added instance constraints.

Tcl Command

`delete_instance_constraints`

Parameters

`<name ...>`	Deletes constraints on the specified instance paths. **Note:** The names are either DFFs or D-latches.
`-Module <module_name* ...>`	Deletes the constraints from the specified module(s). This accepts wildcards.
`-All`	Deletes all instance constraints. `-All` applies within the given defaults.
`-Golden`	Deletes instance constraints in the Golden design. *This is the default.*
`-Revised`	Deletes instance constraints in the Revised design.
`-BOTH`	Deletes instance constraints in both the Golden and Revised designs.

Related Commands

ADD INSTANCE CONSTRAINTS

REPORT INSTANCE CONSTRAINTS

DELETE INSTANCE EQUIVALENCES

```
DELete INstance Equivalences
     < <instance_pathname* ...> | -All>
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes instance equivalences originally added with the `ADD INSTANCE EQUIVALENCES` command.

Use the `REPORT INSTANCE EQUIVALENCES` command to display a list of all added instance equivalences.

Tcl Command

```
delete_instance_equivalances
```

Parameters

`<instance_pathname* ...>`	Deletes equivalences on the specified instance path(s). This accepts wildcards.
`-All`	Deletes all instance equivalences within the given defaults.
`-Golden`	Deletes instance equivalences in the Golden design. *This is the default.*
`-Revised`	Deletes instance equivalences in the Revised design.
`-Both`	Deletes instance equivalences in both the Golden and Revised designs.

Related Commands

ADD INSTANCE EQUIVALENCES

REPORT INSTANCE EQUIVALENCES

DELETE INSTANCE RENAMING

```
DELete INstance Renaming
     <rule_name*>
     ...
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Deletes the specified instance renaming rule (added through the `ADD INSTANCE RENAMING COMMAND`).

Tcl Command

```
delete_instance_renaming
```

Parameters

<rule_name>*	Deletes the specified instance renaming rule. Simple wildcard matching is supported.

Related Commands

ADD INSTANCE RENAMING

ADD MAPPING MODEL

DELETE MAPPING MODEL

MOS2BUFIF

REPORT INSTANCE RENAMING

REPORT MESSAGES

DELETE KEYPOINT INFO

```
DELete KEypoint Info
     <keyword>
     <gate_id ...> | <instance_pathname* ...> | <pin_pathname* ...>
     [-Golden | -Revised]
     (Lec Mode)
```

This command deletes information added to key points using the with the `ADD KEYPOINT INFO` command.

Tcl Command

```
delete_keypoint_info
```

Parameters

`<keyword>`	Specifies the name of the property to delete.
`-Golden`	Specifies that the key points are in the Golden design.
`-Revised`	Specifies that the key points are in the Revised design.

Related Commands

ADD KEYPOINT INFO

REPORT KEY POINT

SET PROJECT NAME

DELETE LIBERTY_COMPARE FILTER

```
DELete LIberty_compare Filter
     <filter_name*> ...
     (Setup/LEC Mode)
```

Description

Note: This is a Conformal Low Power command.

Deletes Liberty compare filters that were added by the `ADD LIBERTY_COMPARE FILTER` command.

Tcl Command

```
delete_liberty_compare_filter
```

Paramters

`<filter_name*> ...`	Specifies the name of the filter(s) to delete. Wildcards are supported.

Example

- The following command deletes Liberty compare filter 'ign_related_power'

  ```
  delete liberty_compare filter ign_related_power
  ```

- The following command deletes all Liberty compare filters:

  ```
  delete liberty_compare filter *
  ```

Related Commands

ADD LIBERTY_COMPARE FILTER

COMPARE LIBERTY

REPORT COMPARED LIBERTY

REPORT LIBERTY_COMPARE FILTER

DELETE LOWPOWER CELLS

```
DELete LOwpower Cells
     <<module_name*> | -All>
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Note: This is a Conformal Low Power command.

Deletes low power cells that were originally defined for modules with the `ADD LOWPOWER CELLS` command.

Use the `REPORT LOWPOWER CELLS` command to display a list of the low power cells used in the design.

Tcl Command

```
delete_lowpower_cells
```

Parameters

`<module_name*>`	Deletes previously added low power cells from the specified modules. This accepts wildcards.
`-ALL`	Deletes previously added low power cells from all modules. All applies within the given defaults.
`-Both`	Deletes the low power cells in the Golden and Revised designs. *This is the default.*
`-Golden`	Deletes the low power cells in the Golden design.
`-Revised`	Deletes the low power cells in the Revised design.

Related Commands

ADD LOWPOWER CELLS

CHECK LOWPOWER CELLS

REPORT LOWPOWER CELLS

REPORT LOWPOWER DATA

SET LOWPOWER OPTION

DELETE MAPPED POINTS

```
DELete MApped Points
     <-All [-Class <Full | User | System>]
       |<<gate_id*> | <instance_pathname*> | <pin_pathname* ...>
       [-Golden | -Revised]>
       |[-NONEQ]
       |[-FUNCTIONAL_mapped]
       |[-Quick]
       |[-UNREACH]
       |[-MERGED_COMPARE_NONEQ]
     >
     (LEC Mode)
```

Deletes mapped points that were one of the following:

- Automatically identified
- Added with the `ADD MAPPED POINTS` command.

Additionally, Conformal deletes all compared points associated with the deleted mapped points.

Use the `REPORT MAPPED POINTS` command to display a list of all mapped points in the *User* and *System* classes of the Golden and Revised designs.

Wildcard: The wildcard (*) represents any zero or more characters in instance or pin paths of mapped points.

Tcl Command

`delete_mapped_points`

Parameters

`-All`	Deletes all mapped points within the given defaults.	
`-Class`	Deletes the specified class of mapped points.	
	`Full`	The *Full* class includes mapped points from both the *User* and *System* classes. *This is the default.*

	`User` — The User class includes mapped points that were previously added with the `ADD MAPPED POINTS` command.
	`System` — The *System* class includes mapped points that were automatically identified when Conformal exited the Setup system mode or were mapped with the `MAP KEY POINTS` command.
`<gate_id*> ...`	Deletes mapped points with these gate ID numbers. This accepts wildcards. ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`<instance_pathname*> ...`	Deletes mapped points from the specified instance paths. This accepts wildcards.
`<pin_pathname*> ...`	Deletes mapped points from the specified pin paths.
`-Golden`	Delete the mapped points from the Golden design. *This is the default.*
`-Revised`	Deletes the mapped points from the Revised design.
`-NONEQ`	Deletes all nonequivalent mapped points.
`-FUNCTIONAL_mapped`	Deletes all functional mapped points.
`-Quick`	Deletes mapped points in quick mode. All compare data will be clear.
`-UNREACH`	Deletes all unreachable mapped points. An unreachable mapped point is one where both the Golden and Revised key points are unreachable. If the key point is the representative of the equivalence group or sequential merge group, it is considered unreachable only if all the member key points in the group are unreachable.
`-MERGED_COMPARE_NONEQ`	Deletes all nonequivalent mapped points in a merged compare.

Related Commands

ADD MAPPED POINTS

DELETE MAPPING MODEL

```
DELete MApping Model
     <module_name | -All> ...
     [-Golden | -Revised]
     (Setup Mode)
```

The `DELETE MAPPING MODEL` command deletes the phase information that was added for the given modules using the `ADD MAPPING MODEL` command.

Tcl Command

`delete_mapping_model`

Parameters

`<module_name> ...`	Specifies the name(s) of the module(s). This accepts wildcards.
`-All`	Specifies that this command applies to all modules.
`-Golden`	Specifies that the modules are in the Golden design. *This is the default*.
`-Revised`	Specifies that the modules are in the Revised design.

Examples

The following command deletes the phase information that is attached to module "`mymod`" in the Revised design.

```
MODE> delete mapping model mymod -revised
```

Related Commands

DELETE MODULE ATTRIBUTE

```
DELete MOdule Attribute
     <<module_name ...> | -All>
     <-PIPELINE_Retime | -COMPARE_Effort | -CPU_Limit | -NOBBOXEMpty | -NOFLatten|
     -NODYNAMIC_RESOLUTION>>
     | -ISOlate_module <g_instance_name> <r_instance_name>>
     [-FLAT_ECO_module]
     [-Golden | -Revised]
     (Setup Mode)
```

Deletes attributes originally assigned to modules with the `ADD MODULE ATTRIBUTE` command.

Use the `REPORT MODULE ATTRIBUTE` command to display a list of all added module attributes.

Tcl Command

`delete_module_attribute`

Parameters

`<module_name ...>`	Deletes previously added attributes from the specified modules.
`-ALL`	Deletes previously added attributes from all modules within the given defaults.
`-PIPELINE_Retime`	Deletes attributes previously added for pipeline-retiming.
`-COMPARE_Effort`	Deletes compare effort levels previously added to modules.
`-CPU_Limit`	Deletes the CPU time limit imposed with the `ADD MODULE ATTRIBUTE` command.
`-NOBBOXEMpty`	Deletes attributes previously added by `ADD MODULE ATTRIBUTE -NOBBOXEMpty` attribute.
`-NOFLatten`	Deletes attributes previously added by `ADD MODULE ATTRIBUTE -NOFlatten`.
`-NODYNAMIC_RESOLUTION`	Deletes attributes previously added by `ADD MODULE ATTRIBUTE -NODYNAMIC_RESOLUTION`.

`-ISOlate_module <g_instance_name><r_instance_name>`	
	Deletes the module attribute required for module isolation (added by `ADD MODULE ATTRIBUTE -isolate_module` option).
`-FLAT_ECO_module`	Deletes attribute previously added and specified by the `ADD MODULE ATTRIBUTE -FLAT_ECO_module` attribute.
`-Golden`	Deletes the specified module attributes in the Golden design. *This is the default.*
`-Revised`	Deletes the specified module attributes in the Revised design.

Related Commands

ADD MODULE ATTRIBUTE

REPORT MODULE ATTRIBUTE

WRITE HIER_COMPARE DOFILE

DELETE NAME ALIAS

```
DELete NAme Alias
     [<filename ...> | -ALL]
     [-GOLden|-REVised]
     (Setup / LEC Mode)
```

Deletes name alias as specified in a JSON data file, or deletes all existing name aliases.

The JSON data file must use following format for the array of name aliases to delete:

```
[{"m":"<module_name>","t":"ins","n":"<name>",...},...]
```

The tool will delete the name alias of keypoint "name" in "module_name".

Tcl Command

```
delete_name_alias
```

Parameters

`<filename>`	Specifies the JSON file that contains the name aliases to delete.
`-ALL`	Delete name aliases for the specified design (Golden or Revised).
`-GOLden`	Delete name alias data for the Golden design.
`-REVised`	Delete name alias data for the Revised design.

Related Commands

ADD NAME ALIAS

REPORT NAME ALIAS

SET MAPPING METHOD

REPORT GATE

WRITE MAPPED POINTS

DELETE MOS DIRECTION

```
DELete MOs Direction
     <module_name> <instance_name>
     [-Golden | -Revised]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Deletes the unidirection that was placed on transistor-MOS instances with the `ADD MOS DIRECTION` command.

Use the `REPORT MOS DIRECTION` command to display a list of all transistor-MOS direction instances.

Tcl Command

`delete_mos_direction`

Parameters

`<module_name>`	Deletes MOS direction for the specified module.
`<instance_name>`	Deletes MOS direction for the specified instance.
`-Golden`	Deletes MOS direction from the Golden design. *This is the default.*
`-Revised`	Deletes MOS direction from the Revised design.

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE NET ATTRIBUTE

```
DELete NEt Attribute
     <-ALL_Net | <net_name>>
     [-ROot | -Module <module_name> | -ALL_Module]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Deletes attributes that were placed on transistor-MOS nets with the `ADD NET ATTRIBUTE` command.

Use the `REPORT NET ATTRIBUTE` command to display a list of all attributes placed on transistor-MOS nets.

Tcl Command

`delete_net_attribute`

Parameters

`-ALL_Net`	Deletes all net attributes within the given defaults.
`<net_name>`	Deletes the specified transistor-MOS net attributes.
`-ROot`	Deletes net attributes associated with the root module, which contains the transistor-MOS. *This is the default.*
`-Module <module_name>`	Deletes net attributes associated with the specified module, which contains the transistor-MOS.
`-ALL_Module`	Deletes a specified net attribute for all modules, or delete all net attributes for all modules. (Refer to `-all_net` and `-net_name` to understand the two choices.)
`-Golden`	Deletes net attributes from the Golden design. *This is the default.*
`-Revised`	Deletes net attributes from the Revised design.
`-Both`	Deletes net attributes from both the Golden and Revised designs.

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

READ PATTERN

REPORT CLOCK

REPORT MOS DIRECTION

REPORT NET ATTRIBUTE

REPORT PIN DIRECTION

RESOLVE

DELETE NET CONSTRAINTS

```
DELete NEt Constraints
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes either the Golden or Revised design net constraints originally added with the `ADD NET CONSTRAINTS` command.

Use the `REPORT NET CONSTRAINTS` command to display a list of all added net constraints.

Tcl Command

```
delete_net_constraints
```

Parameters

`-Golden`	Deletes net constraints from the Golden design. *This is the default.*
`-Revised`	Deletes net constraints from the Revised design.
`-Both`	Deletes net constraints from both the Golden and Revised designs.

Related Commands

ADD NET CONSTRAINTS

REPORT NET CONSTRAINTS

DELETE NOBLACK BOX

```
DELete NOblack Box
     <<module_name* ...> | -All>
     [| -INCLUDE_SUBmodules | -SUBmodules_only]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes the specified module names originally added with the `ADD NOBLACK BOX` command.

Use the `REPORT NOBLACK BOX` command to display a list of all of the modules that will be resolved (flattened) to their parents' modules during hierarchical dofile script generation.

Wildcard: The wildcard (*) represents any zero or more characters in module names.

Tcl Command

`delete_noblack_box`

Parameters

`<module_name* ...>`	Deletes the previously added noblackbox modules.
`-All`	Deletes all previously added noblackboxes. `-All` applies within the given defaults.
`-INCLUDE_SUBmodules`	Deletes the noblack box attribute on the specified module(s) and all its submodules that were added using a previous `ADD NOBLACK BOX -include_submodules` command.
`-SUBmodules_only`	Deletes the noblack box attribute on all submodules of the specified module(s) that were added using a previous `ADD NOBLACK BOX -submodules_only` command.
`-Golden`	Deletes the specified Golden module names. *This is the default.*
`-Revised`	Deletes the specified Revised module names.
`-Both`	Deletes all of the specified modules from both the Golden and Revised designs.

Related Commands

ADD NOBLACK BOX

REPORT NOBLACK BOX

WRITE HIER_COMPARE DOFILE

DELETE NOTRANSLATE FILEPATHNAMES

```
DELete NOtranslate Filepathnames
     <<filepath_names* ...> | -All>
     [ | -Library | -Design]
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Deletes the specified file pathnames originally added with the `ADD NOTRANSLATE FILEPATHNAMES` command.

Use the `REPORT NOTRANSLATE FILEPATHNAMES` command to display a list of all of the library and design file pathnames.

Wildcard: The wildcard (*) represents any zero or more characters in module names.

Tcl Command

```
delete_notranslate_filepathnames
```

Parameters

`<filepath_name* ...>`	Deletes the listed notranslate file pathnames.
`-All`	Deletes all previously added notranslate file pathnames within the given defaults.
`-Library`	Deletes the specified library file pathnames. *This is the default.*
`-Design`	Deletes the specified design file pathnames.
`-Both`	Deletes the specified file pathnames from both the Golden and Revised designs. *This is the default.*
`-Golden`	Deletes the specified Golden file pathnames.
`-Revised`	Deletes the specified Revised file pathnames.

Related Commands

ADD NOTRANSLATE FILEPATHNAMES

DELETE NOTRANSLATE MODULES

REPORT NOTRANSLATE FILEPATHNAMES

REPORT NOTRANSLATE MODULES

DELETE NOTRANSLATE MODULES

```
DELete NOtranslate Modules
    <<module_name* ...> | -All>
    [-Library | -Design]
    [-Both | -Golden | -Revised]
    (Setup Mode)
```

Deletes the specified module names originally added with the `ADD NOTRANSLATE MODULES` command.

Use the `REPORT NOTRANSLATE MODULES` command to display a list of all of the library and design module names that will not be compiled.

Wildcard: The wildcard (*) represents any zero or more characters in module names.

Tcl Command

```
delete_notranslate_modules
```

Parameters

`<module_name* ...>`	Deletes the listed modules.
`-All`	Deletes all previously added notranslate module names within the given defaults.
`-Library`	Deletes the specified library module names. *This is the default.*
`-Design`	Deletes the specified design module names.
`-Both`	Deletes the specified modules from both the Golden and Revised designs. *This is the default.*
`-Golden`	Deletes the specified Golden module names.
`-Revised`	Deletes the specified Revised module names.

Related Commands

ADD NOTRANSLATE MODULES

READ DESIGN

READ LIBRARY

REPORT NOTRANSLATE MODULES

DELETE NOTRANSLATED LINES

```
DELete NOtranslated Lines
     <filename | partial_filename> <[lib libname]>
     [-Golden | -Revised | -Both]
     (Setup / LEC Mode)
```

Deletes the specified file pathname originally added with the `ADD NOTRANSLATED LINES` command.

Use the `REPORT NOTRANSLATED LINES` command to display a list of all of the library and design file pathnames.

Wildcard: The wildcard (*) represents any zero or more characters in partial filename.

Tcl Command

```
delete_notranslated_lines
```

Parameters

`<filename>`	Deletes the listed notranslate file pathname.
`<partial_filename>`	Deletes the listed notranslate file partial pathname.
`<[lib libname]>`	Delete the file only apply when reading in files for the specified library by `-library` option of `READ DESIGN` command.

Related Commands

ADD NOTRANSLATED LINES

REPORT NOTRANSLATED LINES

DELETE OUTPUT EQUIVALENCES

```
DELete OUtput Equivalences
     <-ALL_Pin | <primary_pin* ...>>
     [-ROot | -Module <module_name> | -ALL_Module]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes the output pin equivalences placed on output boundary module pins with the `ADD OUTPUT EQUIVALENCES` command.

Use the `REPORT OUTPUT EQUIVALENCES` command to display a list of all added output pin equivalences.

Wildcard: The wildcard (*) represents any zero or more characters in output boundary module pin names.

Tcl Command

```
delete_output_equivalences
```

Parameters

`-ALL_Pin`	Deletes all output pin equivalences within the given defaults.
`<primary_pin* ...>`	Deletes output pin equivalences from the listed output boundary module pins.
`-ROot`	Deletes the output pin equivalences from the root module.
`-Module <module_name>`	Deletes the output pin equivalences from the specified module.
`-ALL_Module`	Deletes the output pin equivalences from all modules.
`-Golden`	Deletes the specified output pin equivalences in the Golden design. *This is the default.*
`-Revised`	Deletes the specified output pin equivalences in the Revised design.
`-Both`	Deletes the specified output pin equivalences in both the Golden and Revised designs.

Related Commands

ADD OUTPUT EQUIVALENCES

REPORT OUTPUT EQUIVALENCES

DELETE OUTPUT STUCK_AT

```
DELete OUtput Stuck_at
     <-ALL_Pin | <primary_pin* ...>>
     [-ROot | -Module <module_name*> | -All]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes the output `stuck_at` values placed on output boundary module pins with the `ADD OUTPUT STUCK_AT` command.

Use the `REPORT OUTPUT STUCK_AT` command to display a list of all added output `stuck_at` values and their pin names.

Tcl Command

```
delete_output_stuck_at
```

Parameters

`-ALL_Pin`	Deletes all output `stuck_at` values within the given defaults.
`<primary_pin* ...>`	Deletes the output `stuck_at` values associated with the listed output boundary module pins. This accepts wildcards.
`-ROot`	Deletes the output `stuck_at` values in the root module boundary pin. *This is the default*.
`-Module <module_name*>`	Deletes the output `stuck_at` values in the specified module. This accepts wildcards.
`-All`	Deletes the output `stuck_at` values in all output boundary module pins.
`-Golden`	Deletes the output `stuck_at` values in the Golden design. *This is the default.*
`-Revised`	Deletes the output `stuck_at` values in the Revised design.
`-Both`	Deletes the output `stuck_at` values in both the Golden and Revised designs.

Related Commands

ADD OUTPUT STUCK_AT

REPORT OUTPUT STUCK_AT

DELETE PARTITION KEY_POINT

```
DELete PArtition Key_point
     (Setup Mode)
```

Deletes all of the specified partition key points originally added with the `ADD PARTITION KEY_POINT` command.

Use the `REPORT PARTITION KEY_POINT` command to display a list of all added partition key points.

Tcl Command

```
delete_partition_key_point
```

Related Commands

ADD PARTITION KEY_POINT

REPORT PARTITION KEY_POINT

WRITE PARTITION DOFILE

DELETE PARTITION POINTS

```
DELete PARtition Points
     <<pathname> | -All | -BAD_cuts [-EFFORT <Low | Medium | High>]>
     [-NONequivalent]
     [-Golden | -Revised]
     [-NOVerbose | -Verbose]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Deletes the partition points that were created with the `ADD PARTITION POINTS` command.

Note: Partition points are always deleted in pairs.

Tip

You can get the pathname of the partition point with the `REPORT PARTITION POINT` command.

Caution

Deleting partition (CUT) points in LEC mode causes flattened netlists to change. As a result, all the gate-ids are subjected to change. Deleting cut points does not affect the existing compare points list; however, all the compare data is invalidated after deleting cut points.

Tcl Command

```
delete_partition_points
```

Parameters

`<pathname>`	Specifies the name of the path for the partition points to be deleted.
`-All`	Specifies that all the existing partition points will be deleted.
`-BAD_cuts`	Automatically deletes the set of partition points that cause false nonequivalence.

`-EFFORT <Low | Medium | High>`	
	Specifies the effort level for deleting the bad set of partition points.
	Increasing the effort level from `Low` to `High` will intelligently delete more partition points decreasing the probability of false nonequivalence. However, increasing the effort will result in increased time as well.
	The default effort level is `Low`.
`-NONequivalent`	Deletes partition cut points that are nonequivalent.
`-Golden`	Specifies that the partition point is in the Golden design. *This is the default.*
`-Revised`	Specifies that the partition point is in the Revised design.
`-NOVerbose`	Do not provide additional information. *This is the default.*
`-Verbose`	Provides additional information.

Related Commands

ADD PARTITION POINTS

REPORT PARTITION POINTS

DELETE PHYSICAL CELLS

```
DELete PHysical Cells
     <-ALL | <module* ...> | -REMove_in_netlist>
     [-BOTh | -GOLden | -REVised ]
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Deletes physical cells from the `physical_cell_list`, which is created by the `EXTRACT PHYSICAL CELL` command.

Tcl Command

```
delete_physical_cells
```

Parameters

`-ALL`	Removes all cells from the `physical_cell_list`, which is created by the `EXTRACT PHYSICAL CELL` command.
`<module_name* ...>`	Removes the specified module(s) from the `physical_cell_list`, which is created by the `EXTRACT PHYSICAL CELL` command. Simple wildcard matching is supported.
`-REMove_in_netlist`	Removes all instances listed in the `physical_cell_list` from the netlist and from the `physical_cell_list`.
`-BOTh`	Delete physical cell in both the Golden and Revised designs. *This is the default.*
`-GOLden`	Delete physical cell in the Golden design.
`-REVised`	Delete physical cell in the Revised design.

Related Commands

ADD PHYSICAL CELLS

EXTRACT PHYSICAL CELLS

REPORT PHYSICAL CELLS

DELETE PIN BINDING

```
DELete PIn Binding
     <golden_pin> <revised_pin>
     [-MOdule <golden_module_name> <revised_module_name>]
     (Setup Mode)
```

Deletes pin binding pairs originally added with the ADD RENAMING RULE command. Use REPORT PIN BINDING command to display a list of all of the pairs of pin binding.

Tcl Command

`delete_pin_binding`

Parameters

`-MOdule`	Specifies the module pair that the pin binding rules are applied in. Default module pair is the golden and revised top modules.

Related Commands

ADD RENAMING RULE

ADD PIN BINDING

REPORT PIN BINDING

DELETE PIN CONSTRAINTS

```
DELete PIn Constraints
     <-ALL_Pin | <primary_pin* ...>>
     [-Module <module_name>]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes constraints originally placed on named primary input pins with the `ADD PIN CONSTRAINTS` command.

Use the `REPORT PIN CONSTRAINTS` command to display a list of all added pin constraints.

Wildcard: The wildcard (*) represents any zero or more characters in primary input names.

Tcl Command

```
delete_pin_constraints
```

Parameters

`-ALL_Pin`	Deletes all constraints placed on primary input pins within the given defaults.
`<primary_pin* ...>`	Deletes constraints from the listed primary inputs.
`-Module <module_name>`	Deletes pin constraints from the specified module.
`-Golden`	Deletes the specified pin constraints from the Golden design. *This is the default.*
`-Revised`	Deletes the specified pin constraints from the Revised design.
`-Both`	Deletes the specified pin constraints from both the Golden and Revised designs.

Related Commands

ADD PIN CONSTRAINTS

REPORT PIN CONSTRAINTS

DELETE PIN EQUIVALENCES

```
DELete PIn Equivalences
     <-ALL_Pin | <primary_pin* ...>>
     [-ROot | -Module <module_name> | -ALL_Module]
     [-Golden | -REvised | -Both]
     (Setup Mode)
```

Deletes the added pin equivalences from the specified primary input pins. These equivalences were placed on primary input pins with the `ADD PIN EQUIVALENCE` command.

Use the `REPORT PIN EQUIVALENCES` command to display a list of all of the added pin equivalences.

Wildcard: The wildcard (*) represents any zero or more characters in primary input names.

Tcl Command

```
delete_pin_equivalences
```

Parameters

`-ALL_Pin`	Deletes all added pin equivalences within the given defaults.
`<primary_pin* ...>`	Deletes pin equivalences from the listed primary input pins. (Pin equivalence was originally added with the `ADD PIN EQUIVALENCES` command.)
`-ROot`	Deletes pin equivalences from the root module.
`-Module <module_name>`	Deletes pin equivalences from the specified module.
`-ALL_Module`	Deletes pin equivalences from all modules.
`-Golden`	Deletes the specified pin equivalences from the Golden design. *This is the default.*
`-REvised`	Deletes the specified pin equivalences from the Revised design.
`-Both`	Deletes the specified pin equivalences from both the Golden and Revised designs.

Related Commands

ADD PIN EQUIVALENCES

REPORT PIN EQUIVALENCES

DELETE POWER_INTENT_COMPARE FILTER

```
DELete POwer_intent_compare Filter
     <filter_name*> ...
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Deletes the specified power intent compare filters. Power intent compare filters are added using the `ADD POWER_INTENT_COMPARE FILTER` command.

Tcl Command

`delete_power_intent_compare_filter`

Parameters

`<filter_name*> ...`	Specifies the name of the filter(s) to delete. Wildcards are supported.

Example

- The following command deletes power intent compare filters '`iso_loc`'

```
delete power_intent_compare filter iso_loc
```

- The following command deletes all power intent filters:

```
delete power_intent_compare filter *
```

Related Commands

COMPARE POWER INTENT

REPORT COMPARED INTENT

REPORT POWER INTENT

DELETE PRIMARY INPUTS

```
DELete PRimary Inputs
     <-All | <pathname* ...>>
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes specified primary inputs that were originally added with the `ADD PRIMARY INPUT` command. After you delete the primary input pins from either the Golden or Revised design, the associated nets become floating nets, unless there are other net drivers.

Use the `REPORT PRIMARY INPUTS` command to display a list of all primary inputs.

Wildcard: The wildcard (*) represents any zero or more characters in paths of primary inputs.

Tcl Command

```
delete_primary_inputs
```

Parameters

`-All`	Deletes all primary inputs within the given defaults.
`<pathname* ...>`	Deletes the specified primary inputs.
`-Golden`	Deletes the specified primary inputs from the Golden design. *This is the default.*
`-Revised`	Deletes the specified primary inputs from the Revised design.
`-Both`	Deletes the specified primary inputs from both the Golden and Revised designs.

Related Commands

ADD PRIMARY INPUT

REPORT PRIMARY INPUTS

DELETE PRIMARY OUTPUTS

```
DELete PRimary Outputs
     <-All | pathname* ...>
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Deletes primary outputs that were originally added with the `ADD PRIMARY OUTPUT` command. When you delete the primary output pins from the Golden or Revised design, the nets become floating nets, unless there are other net drivers.

Use the `REPORT PRIMARY OUTPUTS` command to display a list of all primary outputs.

Wildcard: The wildcard (*) represents any zero or more characters in paths of primary outputs.

Tcl Command

```
delete_primary_outputs
```

Parameters

`-All`	Deletes all primary outputs within the given defaults.
`<pathname* ...>`	Deletes the specified primary outputs.
`-Golden`	Deletes primary outputs from the Golden design. *This is the default.*
`-Revised`	Deletes primary outputs from the Revised design.
`-Both`	Deletes primary outputs from both the Golden and Revised designs.

Related Commands

ADD PRIMARY OUTPUT

REPORT PRIMARY OUTPUTS

DELETE PROJECT

```
DELete PRoject
     <-Run <integer ...> >
     (Setup/LEC Mode)
```

This command deletes the LEC run information (specified by the `Run` number) from the current `LEC` project. Use the `SET PROJECT NAME` command to set the current LEC project.

Tcl Command

```
delete_project
```

Parameters

`<-Run <integer ...> >`	Deletes one or more LEC runs. Note: You cannot delete an LEC run until it has completed.

Related Commands

ANALYZE PROJECT

SET PROJECT NAME

SET PROJECT OPTIONS

SET PROJECT PROPERTY

DELETE RENAMING RULE

```
DELete REnaming Rule
     <-All | <rule_name>>
     [-MAp | -MOdule | -PIn | -INSTance]
     (Setup / LEC Mode)
```

Deletes renaming rules originally added with the `ADD RENAMING RULE` command.

Use the `REPORT RENAMING RULE` command to display a list of all of the renaming rules and their rule numbers.

Tcl Command

`delete_renaming_rule`

Parameters

`-All`	Deletes all previously added renaming rules from the Map, Module, Pin, and Instance categories. If you do not specify a category, Conformal deletes all previously added renaming rules from the Map category.
`<rule_name>`	Deletes the specified renaming rule.
`-MAp`	Deletes only map renaming rules. *This is the default.*
`-MOdule`	Deletes only module renaming rules.
`-PIn`	Deletes only pin renaming rules.
`-INSTance`	Deletes only instance renaming rules.

Related Commands

ADD RENAMING RULE

CHANGE NAME

MAP KEY POINTS

READ DESIGN

READ LIBRARY

REPORT RENAMING RULE

SET MAPPING METHOD

SET NAMING RULE

TEST RENAMING RULE

DELETE RETENTION MAPPING

```
DELete REtention_register Mapping
     <-All | <rule_name>>
     (Setup / LEC Mode)
```

Note: This is a Conformal Low Power command.

Deletes the state retention mapping rules added using the `ADD REtention_register Mapping` command.

Note: Use the `REPORT RETENTION MAPPING` command to display a list of all the state retention mapping rules. Note that the default rule added by the system can never be deleted.

For a description of the default rules that are added by the system, see CHECK LOWPOWER CELLS.

Tcl Command

```
delete_retention_register_mapping
```

Parameters

`-All`	Deletes all the state retention register mapping rules added using the `ADD RETENTION MAPPING Mapping` command. This option does not delete the default rule added by the system.
`<rule_name>`	Deletes the specified state retention mapping rule. **Note:** The default rule added by the system cannot be deleted.

Related Commands

ADD RETENTION MAPPING

REPORT RETENTION MAPPING

DELETE RULE FILTER

```
DELete RUle Filter
     <filtername*> ...
     (Setup Mode)
```

Deletes the specified rule filters.

Tcl Command

```
delete_rule_filter
```

Parameters

`<filtername*> ...`	Specifies the name(s) of the filter(s) to delete.

Related Commands

ADD RULE FILTER

REPORT RULE FILTER

DELETE RUNTIME LIMIT

```
DELete RUntime Limit
     [-MODule [module_name]]
     [-HIER_compare]
     [-COMmand [command_name]]
     (Setup Mode)
```

Deletes the runtime limit for the specified module, `hierarchical_compare` and commands. The unit of time is in seconds..

Tcl Command

```
delete_runtime_limit
```

Parameters

`-MODule`	Deletes the runtime limit of LEC in the specified module. If the module name is not specified, the runtime limit of the root module will be deleted.
`-HIER_compare`	Deletes the runtime limit of the hierarchical comparison.
`-COMmand`	Deletes the runtime limit of the specified command. If the command name is not specified, the runtime limit of all the commands would be deleted.

Related Commands

DELETE RUNTIME LIMIT

SET RUNTIME LIMIT

DELETE SEARCH PATH

```
DELete SEarch Path
     <-All | <pathname...>>
     [-Design | -Library | -POWER_intent]
     [-Golden | -Revised]
     [-RECursive]
     (Setup Mode)
```

Deletes search paths Conformal uses for the `READ DESIGN` and `READ LIBRARY` commands. Use the `REPORT SEARCH PATH` command to display a list of all search paths.

Tcl Command

`delete_search_path`

Parameters

`-All`	Deletes all previously added search paths within the given defaults.
`<pathname ...>`	Deletes the specified search paths.
`-Design`	Deletes search paths used by the `READ DESIGN` command. *This is the default.*
`-Library`	Deletes search paths used by the `READ LIBRARY` command.
`-POWER_intent`	This is a low power command option. Deletes the search path(s) for power intent files.
`-Golden`	Deletes search paths used by the Golden design or library. *This is the default.*
`-Revised`	Deletes search paths used by the Revised design or library.
`-RECursive`	Deletes the subdirectories of the specified directory from the search paths.

Related Commands

ADD SEARCH PATH

READ DESIGN

READ LIBRARY

REPORT SEARCH PATH

DELETE SEQ_CORRESPONDENCE

```
DELete SEQ_CORRespondence
     <-ALL | <golden_identifier> <revised_identifier> >
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Deletes the sequential corresponding points that were added with the `ADD SEQ_CORR` or `ANALYZE RETIMING -general` command.

Tcl Command

`delete_seq_correspondence`

Parameters

`-ALL`	Deletes all the sequential corresponding points.
`<golden_identifier>`	Specifies the sequential corresponding gate ID or instance pathname for the Golden register to delete.
`<revised_identifier>`	Specifies the sequential corresponding gate ID or instance pathname for the Revised state point to delete.

Example

The following command deletes all the sequential corresponding points:

```
delete seq_corr -all
```

Related Commands

ANALYZE RETIMING

ADD SEQ_CORRESPONDENCE

REPORT SEQ_CORRESPONDENCE

SET RETIMING OPTION

DELETE SUPPLY

```
DELete SUpply
     <object_list>
     [-ROOT | -Module <name_list*> | -ALL]
     [-PORT | -GLOBAL ]
     [-Golden| -Revised | -Both]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Deletes power and ground pins in the design that were defined with the `ADD SUPPLY` command.

Tcl Command

`delete_supply`

Parameters

`<object_list>`	Specifies that the list of net or port names that will have their attribute settings deleted.
`-ROOT`	Delete this supply attribute to the specified objects in the current scope and all hierarchy of this scope. *This is the default.*
`-Module <name_list*>`	Delete the attribute setting to the specified module. This accepts wildcards.
`-ALL`	Deletes the attribute settings to the objects for all modules.
`-PORT`	The defined object(s) must be the port(s) at the root or the specified module level. *This is the default.*
`-GLOBAL`	The defined object(s) could be the port(s) and wire(s) in the hierarchy of the root or the specified module.
`-Golden`	Specifies that the listed names are from the Golden design. *This is the default.*
`-Revised`	Specifies that the listed names are from the Revised design.

`-Both`	Specifies that the listed names are from both the Golden and Revised designs.

Related Commands

ADD SUPPLY

REPORT SUPPLY

DELETE TIED SIGNALS

```
DELete TIed Signals
     <-All | <signal_name ...>>
     [-Class <Full | User | System>]
     [-Module <module_name>]
     [-Net |-Pin]
     [-Golden |-Revised]
     (Setup Mode)
```

Deletes specified tied signals from the Golden or Revised design.

Use the `REPORT TIED SIGNALS` command to display a list of all of the tied signals.

Tcl Command

```
delete_tied_signals
```

Parameters

`-All`	Deletes all tied signals within the given defaults.	
`<signal_name ...>`	Deletes the specified tied signals.	
`-Class`	Deletes tied signals of this class.	
	`Full`	Tied signals from both the User and System classes. *This is the default.*
	`User`	Tied signals the user previously added with the `ADD TIED SIGNALS` command.
	`System`	Tied signals from the original design.
`-Module <module_name>`		
	Specifies the name of the module where the floating net or pin resides.	
`-Net`	Specifies that the deleted tied signal is a net. *This is the default.*	
`-Pin`	Specifies that the deleted tied signal is a pin.	
`-Golden`	Deletes tied signals from the Golden design. *This is the default.*	

`-Revised`	Deletes tied signals from the Revised design.

Related Commands

ADD TIED SIGNALS

REPORT TIED SIGNALS

DIAGNOSE

```
DIAgnose
     <<gate_id> | <instance_pathname> | <pin_pathname>
       [-Golden | -Revised]
       [-SUPport]
       [-MERge]
       [-NUm <integer>]
      | -SUMmary [integer][-SOrt <SUpport | SIze>]
      |[-NOneq]>
      [-GROup]
      [-VERBose]
     (LEC Mode)
```

Runs diagnosis on a specified compared point. Specify the compared point by its gate identification number, instance path, or a pin path. Use this command to determine why the software identified nonequivalence between compared points.

The diagnosis displays all of the non-corresponding support key points with a list of all likely error candidates from the Revised design. The list organizes likelihood in descending order with 1.00 being the greatest possible error candidate.

Use the `REPORT ENVIRONMENT` command to display the maximum diagnosis candidates setting.

Note: The syntax above assumes you are diagnosing mapped compare points (where you only need to specify one compare point). When you are diagnosing instance/sequential merge nonequivalence, you must specify two compare points.

Tcl Command

```
diagnose
```

Parameters

`<gate_id>`	Diagnoses the specified gate. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`<instance_pathname>`	Diagnoses the specified instance path.
`<pin_pathname>`	Diagnoses the specified pin path.

`-Golden`	Diagnoses the Golden design. *This is the default.*
`-Revised`	Diagnoses the Revised design.
`-MERge`	Diagnoses merged compare points. **Note:** The following options are not supported when diagnosing instance/sequential merge nonequivalence: ■ `-SUMmary integer` ■ `-SOrt` ■ `-SUPport`
`-SUPport`	Displays the list of corresponding and non-corresponding support points.
`-NUm integer`	Lists the specified number of error candidates. *By default, Conformal lists all error candidates.*
`-SUMmary integer`	Displays a table of the nonequivalent points with their corresponding support size, non-corresponding support size, and cone size. The integer represents the number of nonequivalent points you wish to display in the table. *By default, Conformal displays all nonequivalent points.*
`-SOrt`	Sorts the summary table results by one of the following.: `SUpport` Sorts by corresponding support size. *This is the default.* `SIze` Sorts by cone size
`-NOneq`	Diagnoses every nonequivalent point.
`-GROup`	Performs multipoint diagnosis.
`-VERbose`	Prints out additional information about multipoint diagnosis. This option is only effective with option `-GROup`.

Examples

Note: For a set of sample commands that shows this and related commands in context, see the example for the COMPARE command.

To diagnose nonequivalence shown in the "Compare results of merged compare points" section, use the `diagnose` command with the `-merge` option and specify two compare points. For example:

```
diagnose -merge 6 10
```

The `diagnose` command specifies two compare points, the first in Golden design and the second in the Revised design.

To diagnose nonequivalence shown in the "Compare results of instance/output/pin equivalences and/or sequential merge" section, use the `diagnose` command with either the `-golden` or `-revised` option and specify two compare points that are in the same design:

```
diagnose -revised 9 10
```

The `diagnose` command specifies two compare points in the Revised design.

Related Commands

PROVE

REPORT COMPARE DATA

REPORT ENVIRONMENT

REPORT TEST VECTOR

DIAGNOSE RULE CHECK

```
DIAgnose RUle Check
     <occr_fullname>
     (Setup Mode)
```

Note: This is a Conformal Low Power command.

Diagnoses rule occurrences.

Tcl Command

`diagnose_rule_check`

Parameters

`<occr_fullname>`	Species the fully qualified name of the rule occurrence to diagnose. The name of a rule occurrence is `<rule_inst_name>/<occr_id>`.

Example

The following command runs diagnosis on occurrence 1 in CROSSING2/1:

```
diagnose rule check CROSSING2/1
```

DOFILE

```
DOFile
     <<filename> [-FORCE] | -SHOW_STACK>
     [-NOECHO]
     (Setup / LEC Mode)
```

Executes a set of commands contained in a specified file. If there is an error while the Conformal software is executing the dofile script, it terminates the dofile execution and returns to the system mode prompt.

Use the `SET DOFILE ABORT` command to specify how you want the Conformal software to respond when an error message occurs. You can choose to terminate, continue, or exit the session.

Use the `BREAK` command in a dofile script to terminate the dofile script and return to the system mode prompt.

Note: The `DOFILE` command is recommended (over the Tcl `source` command, for example) when executing a CFM command dofile.

Tcl Command

```
dofile
```

Parameters

`<filename>`	Specifies a file containing a set of commands the Conformal software executes one at a time.
`-FORCE`	Allows a dofile to be executed multiple times, up to a limit of 16. Without this option, you will get an error if you attempt to run a dofile multiple times.
`-SHOW_STACK`	Specifies that if the current execution is stopped because of a break in one or more dofiles, it will display the current dofile execution stack. For example:

```
1:  dofile_2
   break (line:2)
2:   dofile_1
   dofile dofile_2 (line:5)
```

`-NOECHO`	Do not echo the commands while executing the commands in the file.

Related Commands

BREAK

CONTINUE

SET DOFILE ABORT

WRITE COMPARED POINTS

ELABORATE DESIGN

```
ELAborate DEsign
     [-PARAmeter [-INT | -STR | -ENUM | -TYPE] <parameter_name> <value>]
     [-RAngeconstraint]
     [-ROot <module_name>]
     [-ROOTConfig <configuration_name>]
     [-NOCONFiguration]
     [-ROOTONLY]
     [-GOlden | -REvised]
     (Setup Mode)
```

Completes the `READ DESIGN` command specified with the `-noelaborate` option. During this step, modules are synthesized and the complete design hierarchy is created.

This command is typically used for mixed design flows where the Verilog modules and VHDL entity or architectures are read in separately. Then they can be elaborated using this command.

Tcl Command

`elaborate_design`

Parameters

`-PARAmeter [-INT | -STR | -ENUM] <parameter_name> <value>`

Assigns design parameters or replace existing design parameters. To specify multiple parameters, use the `-parameter` option for each parameter you want to set.

When using the `-parameter -int <parameter_name> <value>` command, the `<value>` will be converted to integer value, which can be a positive integer (`1`), negative integer (`-1`), an integer value recognized as a string (`"1"/"-1"`), or a Verilog style integer (`"16'h0001"`). When using a Verilog style integer, the value must be specified between double-quotes (" ").

When using the `-parameter -str <parameter_name> <value>` command, the `<value>` will be saved as a string.

When using the `-parameter -enum <parameter_name> <value>` command, the `<value>` will be converted to a VHDL enumeration literal.

	When using the `-parameter -type <parameter_name> <value>` command, `<value>` is a Verilog/SystemVerilog built-in type or user-defined type compiled in the compilation unit scope `$unit`: types defined in a module or package are not supported.
	Notes: Any value that is not recognized as an unsigned decimal integer value is interpreted as string value.
	If `-int` or `-str` is not specified, then the parameter value will be interpreted as an integer if it is not between double-quotes (" "), and as a string if it is between double-quotes. Therefore, if you want to specify a Verilog format value, it must be between double-quotes and used with the `-int` option.
	See `READ DESIGN -parameter` for examples.
`-RAngeconstraint`	Applies range constraints during verification. If this option is not specified, all range constraints are ignored.
`-ROot <module_name>`	Specifies the root module to be elaborated. If this option is not specified, the Conformal software automatically selects a root module.
`-ROOTConfig <configuration_name>`	
	Specifies that the design includes the specified configuration for the top-level module.
	Use this option when the design includes multiple configurations for the top-level module. When you use the `-rootconfig` option, you must also use the `-root module_name` option (above).
`-NOCONFiguration`	Do not interpret V2K, SystemVerilog, and VHDL configuration constructs
`-ROOTONLY`	Elaborates the root module only and skips elaboration of the other modules that are not instantiated from the root module.
	Because the elaboration stage is skipped for the uninstantiated modules, this option reduces memory usage and omits the elaboration time error checking.
`-Golden`	Specifies to elaborate the Golden design. *This is the default.*
`-Revised`	Specifies to elaborate the Revised design.

Related Commands

READ DESIGN

READ LIBRARY

EXIT

```
EXIt
     [-Force]
     (Setup / LEC Mode)
```

Ends the existing Conformal session and returns you to the operating system.

Exit Status Codes

On exiting, Conformal returns a status code. A nonzero status code means there is a potential error; that is, either no comparison was done or unmapped, abort, or nonequivalent points exist. The exit status code consists of flags that represent different conditions.

Saving GUI Settings

By default, Conformal *does not* automatically save GUI settings for future sessions. To save your preferred settings, use the GUI exit window and click the *Save GUI settings* check box.

Refer to the *Conformal Equivalence Checking User Guide* for additional information about exit status codes and the procedure to save GUI settings.

Tcl Command

```
exit
```

Parameters

`-Force`	Exits without confirmation.

EXTRACT PHYSICAL CELLS

```
EXTract PHysical Cells
     [-GOLden | -REVised | -BOTh ]
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Physical cells do not affect the behavior of the circuit and therefore they are not necessary for structural and functional analysis. However, a netlist containing physical cells may increase run time and memory usage. To mitigate the run time and memory usage, you can remove the physical cells in the design before performing any analysis or rule checks.

A cell is included in this list if it has satisfied all the following conditions:

- Has no output pins
- Is blackboxed
- Not defined as macro model
- Not defined as a diode clamp cell or with diode clamp pins
- Only has supply pins

Once the list is created, you can report the list of physical cells (`REPORT PHYSICAL CELLS`), add to the list (`ADD PHYSICAL CELLS`), or delete from the list and netlist (`DELETE PHYSICAL CELLS`).

Tcl Command

```
extract_physical_cells
```

Parameters

`-GOLden`	Extract the cell(s) from the Golden design. *This is the default.*
`-REVised`	Extract the cell(s) from the Revised design.
`-BOTh`	Looks for the cell(s) in both the Golden and Revised designs.

Related Commands

ADD PHYSICAL CELLS

DELETE PHYSICAL CELLS

FLATTEN

```
FLAtten
     <-MODule <module_name> | -ALL>
     [-Force | -NOForce]
     [-Library | -NOLibrary]
     [-MATCHHierarchy [-INSTancename | -NOINSTancename][-USE_RENaming_rules]]
     [-Golden | -Revised]
     (Setup Mode)
```

Tip

When using this command with Conformal ECO, you should use the `-nolibrary` option to prevent flattening to the primitive level.

Removes all hierarchy on a specified module or for all modules in the database. If you do not specify one or all modules, Conformal flattens the root module by default. Thus, this command expands all of the gate primitive or transistor primitive devices into the cell that is being flattened.

The following example illustrates the effects of this command:

A cell that is to be flattened contains three cells. One cell has 25 gates and the other two are the same, each with 33 gates. After flattening, the cell now contains 0 cells and 91 gates (1 * 25 + 2 * 33 = 91).

To ensure that the hierarchy matches between the designs, it is recommended that you match the hierarchy between the Golden and Revised designs (`FLATTEN -MATCHHierarchy`) before you use `UNIQUIFY`. Note that in the ECO flow, the module and instance names should not be changed in the golden G1 netlist. Therefore, the `FLATTEN` and `UNIQUIFY` commands should only be applied to the revised G2 netlist in that particular use case.

Tcl Command

```
flatten
```

Parameters

`-MODule <module_name>`

	Flattens the specified module. *The default is to flatten the root module.*
`-Force`	Forces flattening of specified modules in the Golden or Revised design even if those modules have not been flattened in the complementing design. (For example, force flattening for Golden modules when the Revised have not been flattened.) *This is the default.*
`-NOForce`	Do not force flattening when those modules have not been flattened in the complementing design.
`-ALL`	Flattens all modules within the given defaults.
`-Library`	Flattens all modules in the designs and libraries. This is the `-ALL` option default.
`-NOLibrary`	Flattens all modules in the designs.
`-MATCHHierarchy`	To make the hierarchy match between the designs, this option flattens hierarchical modules in the Golden or Revised design. `-INSTancename`: Flattens a module only when there is no matching module and no matching instance of the same name in the complementing design. *This is the default*. `-NOINSTancename`: Flattens a module only if there is no matching module in the complementing design.
`-USE_RENaming_rules`	With this option is specified, the keypoint renaming rules are applied for Golden/Revised instance name matching and module renaming rules are applied for Golden/Revised module name matching.
`-Golden`	Flattens the specified module(s) from the Golden design. *This is the default.*
`-Revised`	Flattens the specified module(s) from the Revised design.

Related Command

RESOLVE

UNIQUIFY

FORWARD

```
FORward
     [<integer>]
     (LEC Mode)
```

Reports fan-out gate information from the currently displayed flatten gate information. The fan-out gate you choose with this command becomes the current flattened gate. Use this command to trace gates in place of repeatedly using the `REPORT GATE` command.

Note: This command does not report gates at the design level.

Tcl Command

```
forward
```

Parameters

`<integer>`	Reports the specified fan-out gate. The value 1 denotes the first fan-out. *The default is 1.*

Related Commands

BACKWARD

REPORT GATE

GENERATE DOFILE

```
GENerate DOfile
     <verification_dir_name>
     <-INPUT_DOFILE <dofile> |
       [-GOLden <netlist_name>]
       -REVised <netlist_name>>
     -IMPLementation_information_directory <dir_name>
```

(*Setup/LEC Mode*)

Generates a refined version of a verification dofile that leverages implementation information and is tailored to the current version of the tool; both of which can help ease verification setup. The generated dofile performs the same verification as the input dofile or between the Golden and Revised netlists, depending on what is specified.

Tcl Command

```
generate_dofile
```

Parameters

`<verification_directory_name>`	
	Name of the directory that will contain the generated dofile. The directory will be named as `<verification_directory_name>.cfm` and will be created in the current directory.
`-INPUT_DOFILE <dofile>`	
	Specifies an input dofile.
	Currently, only dofiles generated by the RTL Compiler `write_do_lec` command are supported.
`-Golden <netlist_name>`	Specifies the Golden netlist to verify. The default is RTL.
`-REVised <netlist_name>`	Specifies the Revised netlist to verify.
`-IMPLementation_information_directory <dir_name>`	
	Specifies the directory that contains the implementation information. Currently, only `fv/*` directories generated by RTL Compiler are supported.

Related Commands

READ IMPLEMENTATION INFORMATION

SET VERIFICATION INFORMATION

WRITE VERIFICATION INFORMATION

GENERATE ROM PRIMITIVE

```
GENerate ROm Primitive
     <-SIM <outfile>>
     <-CODE_FILE <codefilename>>
     [-CODE_FILE_FORMAT [BIN | HEX]]
     [-CODE_SIGNATURE]
     [-MOD <module_name>]
     [-NO_ACCESS_OUT_LOW | -NO_ACCESS_OUT_HIGH]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Generates a ROM primitive model that you can use to verify against a valid ROM circuit. Conformal generates a ROM primitive that has the following interface:

- `Addr`: Address bus for accessing ROM data.
- `Dout`: Output data from ROM.
- `RE`: Control clock for the output latch or flip-flop, when you set the `-outstate` option to `dlat` or `dff`. When RE is high, ROM data is sampled.
- `CK`: Address decode clock for ROM read operations. ROM is read when the clock is high.

This command reads in a code file that initializes the ROM. This code file should contain one number per line, in binary format. The number of entries in the code file should match the number of words in the memory. As Conformal reads the code file, it assigns each entry to a successive word element in the memory.

The following illustrates sample contents for a code file called `rom.code`, which initializes a 4 X 4 ROM:

```
0000
1111
1010
0111
```

Note: To perform simulation, you must define the macro `SIM`.

Tcl Command

```
generate_rom_primitive
```

Parameters

`-SIM <outfile>`	Specifies the output file for the ROM primitive.
`-CODE_FILE <codefilename>`	Specifies the code file that will initialize the ROM. This code file should contain initialization data in binary format.
`-CODE_FILE_FORMAT [BIN | HEX]`	Specifies the output file format for the ROM primitive.
`-CODE_SIGNATURE`	Includes the signature generated in the ROM primitive in the code file content.
`-MOD <module_name>`	Specifies the module name of the ROM primitive that is created.
`-NO_ACCESS_OUT_LOW`	Fills the memory address with '0' when it is not initialized. *This is the default.*
`-NO_ACCESS_OUT_HIGH`	Fills the memory address with '1' when it is not initialized.

Related Command

READ ROM PRIMITIVE

Examples

The following command generates a ROM model with a code file called `rom.code`.

```
generate rom primitive -sim VROM.v -code_file rom.code -mod VROM /
-code_file_format bin -no_access_out_low
```

GROUP

```
GROUP
     <-Module <module_name>>
     <-Instance <instance_name* ...>>
     <-NEWModule <module_name> >
     <-NEWInstance <instance_name>>
     [-net_to_pin_name]
     [-Golden | -Revised]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Groups defined instances together so that they become a new submodule. This command is the opposite of the `RESOLVE` command; it applies to submodules, latches, registers, gates, and transistors. By default, this command assigns unique and arbitrary submodule pin names.

Wildcard: The wildcard (*) represents any zero or more characters in existing instance names.

Tcl Command

```
group
```

Parameters

`-Module <module_name>`	
	Specifies a module for which to apply the grouping.
`-Instance <instance_name* ...>`	
	Specifies the instances to group. This accepts wildcards.
`-NEWModule <module_name>`	
	Specifies the name of the new module.
`-NEWInstance <instance_name>`	
	Specifies the instance name for the new module.
`-net_to_pin_name`	Specifies that the pin names of the new modules will be the same as the nets connected to them, and not unique and arbitrary.

`-Golden`	Applies this command to modules and instances in the Golden design. *This is the default.*
`-Revised`	Applies this command to modules and instances in the Revised design.

Related Command

RESOLVE

GO HIER COMPARE

```
GO HIer_compare
     <dofile_name>
     [-CHECK_NONEQ]
     [-COMPARE_String <string>]
     [-DYNamic_hierarchy | -NODYNamic_hierarchy]
     [-HIER_STATUS <file>]
     [-NOANALYZE_abort | -ANALYZE_abort]
     [-NOREStart | -REStart]
     [-RETIMED_modules [-TOP | -NOTOP]]
     [-ROOT_module <golden_module> <revised_module>]
     [-VERBOSE]
     (Setup Mode)
```

Note: This requires a Smart LEC license.

Smart LEC offers distributed hierarchical comparison, where modules can be compared in parallel and over multiple computation resources. For a design with an evenly distributed module complexity, performing the hierarchical comparison this way can significantly reduce the turnaround time.

Distributed hierarchical comparison is most effective when comparing designs:

- That are large and have many submodules
- Have balanced hierarchies
- Whose datapath and/or complex modules are isolated to leaf modules
- Where no sub-module comparison dominates run time
- Where the top module comparison has few or no dynamic flattening iterations and there are sufficient CPU and memory resources

To achieve parallel efficiency, LEC uses workers to execute the module compare. A worker is an LEC process used for module comparisons. Workers can be invoked on the localhost or on remote machines.

In a run, workers collaborate with each other to compare the modules in parallel. When switching between the different modules, workers quickly set the target module and perform verification without reloading the design data. Workers can use all LEC multi-threading features, including compare and module datapath analysis (MDP), to achieve massive parallelization.

Tcl Command

`go_hier_compare`

Parameters

`<dofile_name>`	Specifies the name of the hierarchical dofile that was generated with the `write_hier_compare_dofile` command
`-CHECK_NONEQ`	Identify NEQ modules before running the `analyze_datapath` command and skip datapath analysis for these modules.
`-COMPARE_String <string>`	Replaces the compare command in the hierarchical dofile with a string of compare commands while running hierarchical comparison. If the option `-compare_string <string>` is also specified with the `write_hier_compare_dofile` command, LEC still replaces the commands while running hierarchical comparison. Use the semi-colon character (`;`) to separate commands. Use double quotes to surround each compare command (see "Examples" section below).
`-DYNamic_hierarchy`	Auto-flattens the submodules to propagate any design errors to the top level. The flattened modules are merged to the next level in the hierarchy and compared at that level. *This is the default.* When using this option, `-noneq_stop 1` is automatically appended to the compare command during each module comparison.
`-HIER_STATUS <file>`	Track the hierarchical progress in file.
`-NODYNamic_hierarchy`	Runs static hierarchical comparison without auto-flattening the submodules.
`-NOANALYZE_abort`	Appends `-abort_stop 1` to the compare command during each module comparison. *This is the default.* However, if you use the `set_analyze_option -auto` command, it will override this option.

`-ANALYZE_abort`	Inserts the `analyze_abort -compare` command into each uncompared and aborted module's compare script. If this option is used, the compare command is not appended with `-abort_stop 1` during each module comparison.
`-NORESTart`	Preserves the previous compare results. *This is the default*.
`-REStart`	Deletes the previous comparison results.
`-RETIMED_modules [-TOP | -NOTOP]`	Compares and blackboxes the submodules with the `pipeline_retime` attribute. The `pipeline_retime` attribute can be attached to a module using the `add_module_attribute` command. For this option to work correctly, modules with `pipeline_retime` attribute should exist in the hierarchical dofile script. `-top` runs the comparison of the top module such that submodules without the `pipeline_retime` attribute are fully flattened. *This is the default* for `-retimed_module`. `-notop` specifies that comparison stops after the modules with the `pipeline_retime` attribute have been compared and blackboxed. The hierarchical result is reported as 'Inconclusive' because the entire design is not compared.
`-ROOT_module <golden_module> <revised_module>`	Uses the specified modules as the root modules. This is similar to the `-module` option with the `write_hier_compare_dofile` command without having to regenerate the dofile.
`-VERBOSE`	Lists all the hierarchical constraints and additional information.

Examples

The following is the distributed hierarchical comparison flow; it is very similar to a dynamic hierarchical comparison, but steps 2 and 5 apply only to the hierarchical comparison flow:

1. Read in libraries and designs

2. Specify renaming rules and user constraints, if needed

3. Configure resource settings using `set_parallel_option` command options

4. Generate a hierarchical compare dofile using the `write_hier_compare_dofile` command

5. Execute the hierarchical compare dofile using the `go_hier_compare` command

For example, a dofile for a distributed hierarchical comparison flow would look like:

```
read_library lib.v -both
read_design Golden.v -verilog -golden
read_design revised.v -verilog -revised
add_pin_constraint 0 scan_en -revised
add_renaming_rule hier_rule1 "%s_%d$" "@1" -module -revised
set_parallel_option -workers localhost localhost localhost localhost
write_hier_compare_dofile hier_compare.do -constraint ... -replace
go_hier_compare hier_compare.do
```

- In the following command, the compare command in the hierarchical dofile is replaced with two commands during each module comparison, `set compare effort low` and `compare -abort_stop 1 -noneq_stop 1`:

  ```
  go_hier_compare hier.do -compare_string \
  "set compare effort low; compare -abort_stop 1 -noneq_stop 1"
  ```

Related Commands

ANALYZE ABORT

COMPARE

REPORT HIER_COMPARE RESULT

RESET HIER_COMPARE RESULT

SAVE HIER_COMPARE RESULT

WRITE HIER_COMPARE DOFILE

HELP

```
HELp
     [<command_name> | <message_name> | -message] [-Verbose [-VERSION]]
     [-COLOR | -NOCOLOR]
     [-NOSHOW_ERROR_ID | -SHOW_ERROR_ID]
     [-NOSHOW_EXTENDED_HELP | -SHOW_EXTENDED_HELP]
     [-PAGE | -NOPAGE]
     [<error_id>]
     (Setup / LEC Mode)
```

Note: Although the `HELP` command is still available, it is recommended that you use the `MAN` command.

Displays the Conformal commands and their command syntax. To display a group or set of commands, use a keyword such as `ADD`, `DELETE`, `REPORT`, or `SET`.

While in the Tcl mode, the `HELP` command displays a list of all available Conformal Tcl mode commands.

Tcl Command

```
help
```

Parameters

`<command_name>`	Displays the command syntax for a given command name. If you do not specify a command name, the Conformal Equivalence Checker displays all of the commands.
`-COLOR`	Displays the help text with color highlights. *This is the default.* **Note:** This option has no effect if the terminal is not an 'xterm', as determined by the environmental variable `TERM`, or when running `HELP` in the GUI window.
`-NOCOLOR`	Disables the help text with color highlight display. Use this if the text terminal does not support color.
`<message_name>`	Displays help for the corresponding rule check message.
`-message`	Displays all rule check messages.
`-Verbose`	Expands information about the command, including descriptions of the parameters and related commands.

`-VERSION`	Displays the software version at the end of the help output. This can only be used with the `-Verbose` option.
`-NOSHOW_ERROR_ID`	Do not display the error ID. *This is the default.*
`-SHOW_ERROR_ID`	Displays the error ID.
`-NOSHOW_EXTENDED_HELP`	Do not display the extended help. *This is the default.*
`-SHOW_EXTENDED_HELP`	Displays the extended help only. This does not include error IDs.
`-PAGE`	Displays the help text one screenful at a time. *This is the default*. The output is paused for input after one screenful of text is displayed, where you can continue by pressing the following: space bar: displays the next page `h` key: displays a complete list of options `q` key: quits from the pager **Note:** Output displayed with the pager is not saved to the log file specified by `SET LOG FILE` command. **Note:** The pager is not enabled if the help text is less than a screenful, when output is redirected to a file, or when running `HELP` in the GUI window.
`-NOPAGE`	Disables the help text paging display feature.
`<error_id>`	Displays the error message of the specified message ID.

Example

The following is an example of the Tcl mode system prompt and the `HELP` command:

```
TCL_SETUP> help set_current_module
```

Related Command

SEARCH

HISTORY

```
HISTory
     [int]
     [-ALL]
```
(Setup/LEC mode)

Lists the commands entered in the current session.

Tcl Command

```
history
```

Parameters

`[int]`	Limits the number of commands to return. By default, this command returns the last 20 commands entered.
`[-ALL]`	List all commands entered in the current session.

INFO CHECKPOINT

```
INFo CHeckpoint
    <checkpoint_file_name>
    (Setup / LEC Mode)
```

Provides more information about the specified checkpoint file.

Tcl Command

```
info_checkpoint
```

Parameters

checkpoint_file_name	Specifies the name of the checkpoint file that you want to query.

Related Command

CHECKPOINT

INFO SESSION

```
INFo SEssion
     <session_file>
     [-VERsion | -PLATform]
     (Setup / LEC Mode)
```

Displays information about a specified session file. By default (without any options specified), the tool displays the version and platform for the session file.

Note: Restoring a session file from a version of the tool or for a platform that is different from the one you are currently using can cause unexpected results.

Tcl Command

`info_session`

Parameters

`session_file`	Specifies the session file.
`-VERsion`	Display only the Conformal version where the session was created.
`-PLATform`	Display only the platform where the session was created.

Example

```
info session -version session_file
```

Related Command

RESET

RESTORE SESSION

SAVE SESSION

SEARCH

INVERT MAPPED POINTS

```
INVert MApped Points
     <<gate_id> | <instance_pathname> | <pin_pathname> ...>
     [-MODULE <module_name>]
     [-GOLden |-REVised]
     (LEC Mode)
```

When switching the system from Setup mode to LEC mode, Conformal automatically maps key points and places them in the *System* class of mapped points. Use this command to invert the mapping phase for any mapped points. This command also places the points in the *User* class of mapped points.

In the GUI Mapping and Diagnosis Managers, a (-) sign represents an inverted-mapped point. A (+) sign represents a non-inverted mapped point.

Tcl Command

```
invert_mapped_points
```

Parameters

`<gate_id>`	Inverts the mapping phase for the specified gates (identified by number). **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`<instance_pathname>`	Inverts the mapping phase for the specified instances.
`<pin_pathname>`	Inverts the mapping phase for the specified pins.
`-MODULE <module_name>`	Specifies the root module for the specified instance pathname.
`-GOLden`	Specifies that the point identifier refers to the Golden design. *This is the default.*
`-REVised`	Specifies that the point identifier refers to the Revised design.

Related Commands

ADD MAPPED POINTS

DELETE MAPPED POINTS

MAP KEY POINTS

REPORT MAPPED POINTS

REPORT UNMAPPED POINTS

SET MAPPING METHOD

SET NAMING RULE

LICENSE

```
LICense
     [<license_string>]
     (Setup / LEC Mode)
```

Displays information about the Conformal licenses that you currently have checked out.

Tcl Command

```
license
```

Parameters

`<license_string>`	Displays information for a specific license.

MAN

```
MAN
     [<expression>] [-Keyword] [-Verbose]
     [-COLOR | -NOCOLOR] [-PAGE | -NOPAGE]
     (All Modes)
```

This displays the manual pages for a given expression. By default, the MAN command searches through command names, TCL function names, and rule or modeling message IDs for a best match. You can also use the `-Keyword` option to search through descriptions. If an exact match is not found, this command returns a list of all matches. Running this command without any arguments returns a list of all available pages.

Note: In TCL mode, this command is `VPXMODE MAN`.

Tcl Command

`man`

Parameters

`<expression>`	Specifies the search expression.
`-Verbose`	By default, the MAN command display only the first section of a manual page. Use this option to display the entire manual page.
`-Keyword`	Search for all commands, rules, or messages whose description contains the expression given by `name`. Using this option, you can also search based through specific sections. For example, the following command searches through all Example sections for the word `gated_clock`: `MODE> man -k example gated_clock` Keyword searching is done line by line. Therefore, the search expression cannot span over multiple lines.
`-COLOR`	Displays manual page with color highlighting. *This is the default.*
`-NOCOLOR`	Disables color highlighting. You can also use the `c` keyboard shortcut to toggle color highlighting.

`-PAGE`	Displays manual pages one screen at a time. You can use the following keyboard shortcuts while in MAN: ■ Spacebar: Forward one screen. ■ Return Key: Forward one line. ■ b Key: Back one screen. ■ d Key: Forward 1/2 screen. ■ u Key: Back 1/2 screen. ■ a Key: Display all the remaining text. ■ q: Quit. Note: The displayed output is not saved in the logfile specified by the SET LOG FILE command. The pager is not enabled if the help text is less than a screen, when output is redirected to a file, or when MAN is run in the GUI window.
`-NOPAGE`	Displays the entire man page at once.

MAN Page Categories

With the `man` command, results are categorized according to the type of information. For example:

Category	**Description**
LEC-CMD	LEC Commands
CFM-RULE	HDL Rules
LEC-MODEL	LEC Modeling Rules
LEC-TCL	LEC TCL Commands
ECO-CMD	Conformal ECO Commands
CCD-CMD	Conformal CD Commands
CCD-LINT	Conformal CD Lint Rules
CCD-MODEL	Conformal CDCCD Modeling Rules

Category	Description
CCD-TCL	Conformal CD TCL Commands
SDC-RULE	SDC Rules
CLP-CMD	Conformal Low Power Commands
CPF-RULE	Common Power Format Rules
CLP-RULE	Conformal Low Power Rules
CDC-CMD	Conformal Extended Checks Commands
CDC-MODEL	Conformal Extended Checks Modeling Rules
CDC-TCL	Conformal Extended Checks TCL Commands

If there are more than one page page for a command. For example:

```
LEC> man exit
```

has the following results:

```
EXIt (LEC-CMD)
SET EXit Code (LEC-CMD)
exit (LEC-TCL)
get_exit_code (LEC-TCL)
```

To obtain the man page for the exit Tcl command, use the following command:

```
LEC> man exit ccd-tcl
```

Example

For example, the following command displays the manual page for the REPORT DESIGN DATA command:

```
MODE> man rep de d
```

For example, the following command retrieves all entries with the word `datapath`.

```
MODE> man datapath
ANALYZE DATAPATH (LEC)
REPORT DATAPATH OPTION (LEC)
```

The following lists all pages whose Syntax section contains the word `thread`.

```
MODE> man -k syntax thread
```

The following displays all RTL rules whose default severity is error:

```
MODE> man -k rule default severity error
```

There are times when there are multiple uses for a command. For example, the `USAGE` command also has a Tcl version. In those cases, the `MAN` command will list all the possible uses with descriptors. For example:

```
TCL_SETUP> man usage
USAge (LEC-CMD)
usage (LEC-TCL)
```

To retrieve a specific version of a command, append its descriptor. For example:

```
TCL_SETUP> man usage lec-tcl
usage [-CPU | -MEM]
TCL_SETUP> man usage lec-cmd
USAge
[ | -Elapse | -Delta]
[-MIN_COMMAND_SECONDS <double>]
[-NOAuto | -Auto]
(Setup / LEC Mode)
```

Related Commands

HELP

MAP KEY POINTS

```
MAP KEy Points
      (LEC Mode)
```

Automatically maps all key points, then displays a summary of the mapped points in the Golden and Revised designs. In addition, if there are any unmapped points, Conformal displays a summary of the unmapped points in the Golden and Revised designs. Conformal automatically executes this command the first time you exit the Setup system mode and when the flattened gate model changes.

Tcl Command

```
map_key_points
```

Related Commands

ADD MAPPED POINTS

ADD RENAMING RULE

DELETE MAPPED POINTS

DELETE RENAMING RULE

REPORT MAPPED POINTS

REPORT RENAMING RULE

REPORT UNMAPPED POINTS

SET MAPPING METHOD

SET NAMING RULE

TEST RENAMING RULE

MOS2BUFIF

```
MOS2BUFIF
     [-MODule <mod1> <mod2> ... <modn> | -ALL]
     [-FORce | -DRIVENmos | -INStance <ins1> <ins2> ... <insn>]
     [-Golden | -Revised]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

After abstraction for emulation and test support, this transforms all of the unidirectional NMOS devices to `BUFIF1` and unidirectional PMOS devices to `BUFIF0`.

Tcl Command

```
mos2bufif
```

Parameters

`-MODule <mod1> <mod2> ... <modn>`	Transforms the specified list of modules. **Note:** If you do not specify modules, Conformal transforms the current root module.
`-ALL`	Transforms all modules.
`-FORce`	Converts all [r]nmos to bufif1, [r]pmos to bufif0, [r]cmos to a bufif0 – bufif1 pair. *This is the default.*
`-DRIVENmos`	Converts only those MOS devices that are driven by a logic gate or a primary input.
`-INStance <ins1> <ins2> ... <insn>`	Transforms the specified instances. You can only use this option when you have specified a single module (see the `-module` definition).
`-Golden`	Applies transformation in the Golden design. *This is the default.*
`-Revised`	Applies transformation in the Revised design.

Example

Sample Netlist Transformation:

The following is the original netlist:

```
nmos (out, in, ctl) ;
pmos (out, in, ctl) ;
```

The `MOS2BUFIF` command transforms the netlist to the following:

```
bufif1 (out, in, ctl) ;
bufif0 (out, in, ctl) ;
```

Related Command

ABSTRACT LOGIC

MOVE INSTANCE DOWN

```
MOVe INstance Down
     -MODule <module_name>
     -From <from_instance>
     -TO <to_instance_list>
     [-Golden | -Revised]
     (Setup Mode)
```

Moves instances in the same parent module.

Tcl Command

```
move_instance_down
```

Parameters

`-MODule <module_name>`	Specifies the parent module
`-From <from_instance>`	Specifies the instance that is to be moved.
`-TO <to_instance>`	Specifies the destination instance or list of instances.
`-Golden`	Applies in the Golden design. *This is the default.*
`-Revised`	Applies in the Revised design.

NCENCRYPT

```
NCENCRYPT
     <<input_file>
      <output_file>
      [-VErilog | -VHdl | -TCL]
      [-PRAgma]
      [-REPlace]
     | -VERSion >
```

(*Setup / LEC Mode*)

Use the NC Protect protection scheme to encrypt the specified files. This command also works in Tcl mode.

Tcl Command

`ncencrypt`

Parameters

`<input_file>`	File to be encrypted.
`<output_file>`	Name of the encrypted file.
`[-VErilog | -VHdl | -TCL]`	Specifies the format of the input file. The default is Verilog.
`-PRAGMA`	Encrypt only the text between the `protect begin` and `protect end` NC Protect pragmas.
`-REPLACE`	Replaces existing file.
`-VERSion`	Reports the NC library version.

Related Commands

ncdecrypt (Tcl command)

OPEN SCHEMATICS

```
OPEn SChematics
     [-Golden | -Revised]
     [<full_pathname>]
     (Setup / LEC Mode)
```

Opens the schematic viewer and displays the root module schematics. This command cannot be used in the non-graphic mode.

Tcl Command

```
open_schematics
```

Parameters

`-Golden`	Specifies that the path is in the Golden design.
`-Revised`	Specifies that the path is in the Revised design.
`<full_pathname>`	Opens a schematic for the specified path.

Related Commands

CLOSE SCHEMATICS

DIAGNOSE

REPORT GATE

PIN GROUP

```
PIN GRoup
     [-Golden | -Revised]
     [-DEScend | -ASCend]
     [-ADDEXPression <string> <string>]
     [-ADDList <name>[#:#] "<net> <net> ..."]
     [-ALL | -Module "<module_name> <module_name> ..."]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Combines a group of single nets or pins into a bus. The Conformal software uses the following two default patterns to group pins or nets into buses:

- `Name[#]`
- `Name<#>`

For example, nets `blb[3], blb[4], and blb[5]` will be grouped into bus `blb[5:3]`, and pins `wladd[1], wladd[2], and wladd[3]` will be grouped into bus `wladd[3:1]`.

Tcl Command

```
pin_group
```

Parameters

`-Golden`	Performs pin grouping in the Golden design. *This is the default*.
`-Revised`	Performs pin grouping in the Revised design.
`-DEScend`	Defines the bus in descending numerical order. *This is the default*.
`-ASCend`	Defines the bus in ascending numerical order.

`-ADDEXPression <string> <string>`

Specifies expression(s) for rules on signals to bus. You can specify your own renaming mapping of specific names to two default patterns, so that it recognizes those names as buses also.

For example, `-ADDexpression "mybus_%d_bar" "mybus_bar[@1]"`

This maps the following names into the first default bus name:

```
mybus_12_bar mybus_13_bar mybus_14_bar =>
mybus_bar[12] mybus_bar[13] mybus_bar[14]
```

Then the renamed names will be further grouped into bus `mybus_bar[14:12]`

The renaming mapping syntax: `mybus_%d_bar" "mybus_bar[@1]"` is defined in the `ADD RENAMING RULE` command

`-AddList <name>[#:#] "<net> <net>"`

Allows you to bus random signals. These nets can be single nets or mixed single nets and complete busses. The number of nets defined must be equal to the bus range. If a bus range with more than the number of nets is defined, a "-" character is used as a placeholder for that bit position.

`-ALL` Specifies that when pins of a module are converted to a bus, all instantiations of that module need to be updated. *This is the default*.

`-Module "<module_name> <module_name>"`

Specifies the module that needs to be updated pins of that module are converted to a bus.

Related Command

GROUP

PRINTENV

```
PRINTENV
     [<variable>]
     (Setup / LEC Mode)
```

Displays environment variable values.

Tcl Command

```
printenv
```

Parameters

`<variable>`	Prints the value of the specified variable. If you do not specify a variable, this command displays the value of every environment variable.

Related Command

SETENV

PROGRAM MONITOR

```
PROgram MOnitor
     [-INTERVAL <number>]
     [-FILE <name>]
     [-PROFILE <ON|OFF>]
     [-MAX_FILE_SIZE <number>]
     (Setup / LEC Mode)
```

The program monitor is a run time engine that transparently profiles and checks the execution of the program. This information can help troubleshoot performance issues. The profiling file records traces of the program execution and does not contain any design specific information such as netlist and names.

Tcl Command

`program_monitor`

Parameters

`-INTERVAL <`*`variable`*`>`	Sets the monitoring interval in microseconds. The default value is 100000. This setting affects the size of the profile file. You need approximately 360k bytes for one-hour of profiling with an interval setting of one second (for example, `-interval 1000000`). The smallest effective interval is system dependent and is usually greater than 1mS.
`-FILE <`*`filename`*`>`	Specifies the name of the profile file that will store the profiling information. The default filename is "`conformal.prof`". This will overwrite any existing files.
`-PROFILE <ON|OFF>`	Enables or disables the profiling. When profiling is ON, program execution traces are stored in the profile-file.

`-MAX_FILE_SIZE <`*`filesize`*`>`	Specifies the maximum file size of the profile-file in M-bytes. The default value is 128. Profiling will stop when the size of the file reaches approximately this limit. Notice that the final file size may be slightly larger or smaller than this setting.

Example

The following command enables profiling at a monitoring interval of 100000 microseconds and saves the information in a file called `prof.txt`:

```
SETUP> program monitor -profile on -file prof.txt -interval 100000
```

PROVE

```
PROve
     <identifier1>
     <-ONe | -ZEro | <identifier2>>
     [-NOInvert | -Invert | -Both]
     [-Golden | -Revised]
     (LEC Mode)
```

Starts a process that shows whether the specified gates are equivalent or nonequivalent. The proof process checks equivalency for one of the following pairs:

- One gate in each of the Golden and Revised designs
- Two gates in the Golden design
- Two gates in the Revised design.

Use the `ADD DYNAMIC CONSTRAINTS` command to specify constraints you want to use during this proof process.

Tcl Command

```
prove
```

Parameters

`<identifier1>` `<identifier2>`	Specifies the gates from the given design as the first and second prove points. The `identifier` will be either a gate identification number, instance path, or pin path. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`-ONe`	Prove whether the gate specified by `<identifier1>` is equal to a one value.
`-ZEro`	Prove whether the gate specified by `<identifier1>` is equal to a zero value.
`-NOInvert`	Proves for equivalence. *This is the default.*
`-Invert`	Proves for inverted equivalence.

`-Both`	Proves for *either* non inverted or inverted equivalence.
`-Golden`	When two prove points are specified, indicates that both are from the Golden design. **Note:** By default (when `-Golden` or `-Revised` are not specified) the tool assumes the first prove point is from the Golden design, and the second is from the Revised design.
`-Revised`	When two prove points are specified, indicates that both are from the Revised design.

Examples

For a set of sample commands that shows this and related commands in context, see the example for the COMPARE command.

Related Commands

ADD DYNAMIC CONSTRAINTS

DELETE DYNAMIC CONSTRAINTS

DIAGNOSE

REPORT COMPARE DATA

REPORT DYNAMIC CONSTRAINTS

READ DESIGN

```
READ DEsign
     <filename* ...>
     [-ARchitecture <architecture_name>]
     [ | -BBOXUNResolve | -NOBBOXEMpty]
     [-BLAST_inst_port]
     [-CONFiguration | -NOCONFiguration]
     [-CONTINUOUSASSIGNment <BIdirectional | UNIdirectional>]
     [-Define <variable_name>]
     [-ENUMConstraint]
     [-EXClude  <exclude_file_name*>]
     [-File <command_filename>]
     [-FUnctiondefault [0 | 1 | x | z]]
     [-INITial_value]
     [-INSERT_FEEDTHROUGH_buffer]
     [-KEEP_ESCAPED_ID]
     [-KEEP_Float_instance | -KEEP_All_instance | -REMOVE_Float_instance]
     [ | -LAstmod | -OVERWrite_mod]
     [-LIBRary <library_name> <library_path>]
         (this option is the same as -Map)
     [-LOCalref]
     [-LOGIC_ENCODING_OFF]
     [-Map <library_name> <library_path>]
     [-MAPFile <library_name> [=<library_name2>] <filename ...>[-ENDMAPFile]]
     [-MAPRecursive <library_name> <library_path>]
     [-MErge BBox]
     [-MIXvlog]
     [-NETLIST]
     [-NOELaborate]
     [-NOKeep_unreach | -Keep_unreach]
     [-NORENName]
     [-NOSTRength]
     [-NOZPUSHing | -ZPUSHing]
     [-OPTimize | -NOOPTimize]
     [-PARAmeter [-INT | -STR | -ENUM | -TYPE] <parameter_name> <value>]
         (combined with -ROot option)
     [-RAngeconstraint | -NORAngeconstraint]
     [ | -REPlace | -APPend]
     [-ROot <module_name>]
     [-ROOTConfig <configuration_name>]
     [-ROOTONLY]
     [-SEnsitive | -NOSEnsitive]
     [-STATEtable | -NOSTATEtable]
     [-SUMmary_report | -NOSUMmary_report]
     [-SUPPLY | -NOSUPPLY]
     [-UNCompress <zip_file_name>]
     [-UNZip <zip_file_name ...>]
     [-VErilog | -VERILOG2K | -V1995 | -SYStemverilog | -SVA | -SV09
       | -VHdl [93 | 87 | 2008] | -SPice [-NO_RESistor]| -Ndl | -EDIF | -LIBErty]
```

```
[-VHDLESCaped_to_verilog]
[-VMEM_LIB]
[-VMEM_ULTRA]
[-VERBose]
[-Golden | -REVised]
(Setup Mode)
```

Reads in the Golden and Revised designs.

Important

Review these important reminders before using the `READ DESIGN` command:

- Use the `SET NAMING RULE` command first if you intend to read in an RTL design that requires specific naming conventions.
- Use the `SET UNDEFINED CELL` command before the `READ DESIGN` command if your design includes undefined cells that should be treated as blackboxes.

Note:

- If your design includes duplicate modules, Conformal uses the first module and ignores later ones. However, you can use the `-lastmod` option to specify that Conformal use the last module and ignore the earlier ones.
- Use the tilde character (~) to shorten the path of the file.
- Use the backslash character (\) at the end of a line to show that the command you are entering continues on the next line.

Supported Options

The following Verilog and VHDL considerations are offered:

- Use the `-file` option with a Verilog Command file list. However, only the `-v`, `-y`, `+incdir`, `+libext`, and `+define` options are supported. Additionally, use the `-yd` option to treat library modules as design modules.
- The VHDL option supports all VHDL constructs and all standard and IEEE packages, including synthesis packages. It has an elaboration engine and RTL logic generation that support most RTL VHDL synthesis subset constructs (see below for details). For most non-synthesizable VHDL constructs, Conformal displays warning messages.

VHDL and Verilog 2001 Library Mapping

You can specify how VHDL and Verilog 2001 libraries are mapped using the `READ DESIGN` command's `-map`, `-mapfile`, or `-library` options.

The `-map` and `-library` options work the same in that they map logical library names to physical directories. You can use multiple `-map` commands to map multiple physical directories to one logical library. Use the `-mapfile` option for more specific library mapping, such as specifying that a list of files must be compiled into a specified library. If you read in a file without specifying its library mapping, that file is stored in a default library called `work`.

Note: You can map a file into more than one library. In this case, the file is stored in each library for which it is mapped.

See the "VHDL Support" and "Verilog Support" appendices in the *Conformal Equivalence Checking User Guide* for additional information, including examples on library mapping.

IP Protection

Conformal can read in files that are encrypted using the `ncprotect` utility (available with the Cadence® NC-Verilog Simulator and Cadence® NC-VHDL Simulator) or Cadence encryption tools. Designs that are encrypted can be read into Conformal and compared with other non-encrypted designs.

When there are protected modules, the WRITE DESIGN command will write out the full netlist of the protected modules in either clear text or encrypted text, depending on the level of protection specified when the file was encrypted (for example, the `output_netlist` attribute of NC-Protect specifies the encryption level of the output netlist).

Note: Conformal does not support Cadence Verilog-XL protected files (they are only for simulations), and any other proprietary encryption files.

For more information on protection and synthesis rights, refer to the *Protecting IP Source Files* document of the Cadence simulation tool.

Tcl Command

```
read_design
```

Parameters

`<filename>`	Reads in the specified file. (Required.)
`-ARchitecture <architecture_name>`	Reads in files with the specified architecture. Use this option when multiple architectures of the same root design are compiled. This option is associated with the `-configuration` option, otherwise only one architecture is kept for each entity.
`-BBOXUNResolve`	Specifies that unresolved semantics as unsupported constructs (in VHDL) will be blackboxed instead of erroring out.
`-NOBBOXEMpty`	Specifies that empty modules will be retained instead of being blackboxed.
`-BLAST_inst_port`	Allows the cell model to have a bus pin while the instantiation is bit-blasted. By default, instantiations that contain bit-blasted connections will error out. The Verilog standard does not allow these connections.
`-CONFiguration`	Supports V2K, SystemVerilog, and VHDL configuration constructs. *This is the default.*
`-NOCONFiguration`	Do not interpret V2K, SystemVerilog, or VHDL configuration constructs.
`-CONTINUOUSASSIGNment <BIdirectional | UNIdirectional>`	Specifies that continuous assignment in the design should be interpreted as uni-directional or bi-directional assignment. *The default is bi-directional.*
`-Define <variable_name>`	

Defines `` `ifdef `` variable names in Verilog. To specify multiple definitions, use this option for each definition you want to set. For example:

```
read design filename -define definition1=value1 \
                      -define definition2=value2
```

The macro redefinition in Verilog source code is ignored if the text macro is specified at the Conformal command line or Verilog command file.

`-ENUMConstraint` When a variable is of type of enumeration, a value check is made against the enumeration range. If a value is out of range, it will be interpreted as don't care. For example:

```
variable v : day;
```

Where `day` is an enumeration of `{sun, mon, tue}`

With `-ENumconstraint`, `v` will be interpreted as:

```
((v>=sun && v<=tue)? v : 2'bx)
```

Note: This option is valid only for VHDL and System Verilog.

`-EXClude <exclude_file_name*>`

Specifies files to exclude when reading in the design. This accepts wildcards.

Note: You cannot use multiple wildcards with this option.

`-File <command_filename>`

Reads in the specified command file as a design.

Note: This option is for Verilog, VHDL, or Spice command file lists.

The options for `<command_filename>` are described in this document below in Table 2-1 on page 393 for Verilog and Table 2-2 on page 396 for VHDL.

`-FUnctiondefault` Specifies the default return value for unspecified or incompletely specified functions.

`0` returns zero. *This is the default*.

`1` returns one.

`x` returns x.

`z` returns z.

`-INITial_value`	Specifies that the variable's initial value will not be ignored. **Note:** This option applies only to VHDL designs.
`-INSERT_FEEDTHROUGH_buffer`	When a continuous assignment has a signal that inputs into a submodule and outputs directly, the tool will interpret this as a `through wire` (no direction). LEC will insert a buffer to indicate the wire direction.
`-KEEP_ESCAPED_ID`	When used, this option keeps escaped identifiers, as in Verilog 2001. See "Support for Macro Expansions" in the *Conformal Equivalence Checking User Guide* for more information.
`-KEEP_Float_instance`	Keeps Verilog/VHDL floating module instances that do not have instance pin connections. Note that this option does not keep unreachable Verilog primitives or unreachable logic inferred from the Verilog/VHDL RTL statements. *This is the default.*
`-KEEP_All_instance`	Keeps all Verilog/VHDL floating module instances, the unreachable Verilog primitives, and the unreachable logics inferred from the Verilog/VHDL RTL statements.
`-REMOVE_Float_instance`	Removes Verilog/VHDL floating instance and its corresponding modules in the design database.
`-LAstmod`	Specifies that if duplicate modules exist, the Conformal software uses the last module and ignores the earlier ones. By default, Conformal uses the first module and ignores later ones.
`-OVERWrite_mod`	All module names must be unique within a given library, and same module name can exist in the different libraries. Use this option to keep the current module and delete all existing modules with the same name from all libraries. Without this option, only the module with the same name within the target library is affected.
`-LIBRary <library_name> <library_path>`	Reads in the specified file in the given library and path for user-defined VHDL libraries. (This option is the same as `-map`)

`-LOCalref`	Keep all Verilog modules as local private modules if they are referenced by any module within the same Verilog file. Local private modules cannot be referenced from other Verilog files.
`-LOGIC_ENCODING_OFF`	Do not interpret the user-defined `enum` types with literal value `0`, `1`, `X`, `Z`, and so on as one-bit logic values.

For example, `MY_MVL` is the user-defined `enum` type:

```
type MY_MVL is ('X', '0', '1', 'Z');
```

The type of signal `val` is `MY_MVL`. For the statement:

```
val <= 'Z';
```

With this option, `val` is `2'd3`; without this option, val is `1'bx`.

`-Map <library_name> <library_path>`

Reads in files for the specified `<library_name>` from `<library_path>`.

Use this option to read in all of the files in the specified library path for the given library name. You can also map multiple directories to a single library. Supported for only VHDL files.

For example:

```
read design -vhdl top.vhd -map mylib /design/path1 \
  -map mylib /design/path2
```

```
-MAPFile <library_name> [=<lib_name2>] <filename ...>
```

Reads in the specified `<filename>` and includes them in the `<library_name>`.

Use this option to specify the files that belong to a given library. The file list terminates with the next option or the end of the `READ DESIGN` command.

Note: If you are specifying a list of files with different VHDL standards (`-vhdl 87` and `-vhdl 98`), you must use the `-ENDMAPFile` option to specify the end of the `-MAPFile` file list; otherwise, the file list terminates prematurely. Or, you can use multiple `-mapfile` options to specify multiple files in a library.

For example, the following command includes VHDL-87

files f1.vhd and f2.vhd in lib1, and includes VHDL-93

files f3.vhd and f4.vhd in the default library work:

```
read design -vhdl 87 -mapfile lib1 \

f1.vhd f2.vhd -vhdl 93 f3.vhd f4.vhd
```

However, with the -ENDMAPFile option, the following

command will include all VHDL files in lib1:

```
read design -vhdl 87 -mapfile lib1 \

f1.vhd f2.vhd -vhdl 93 f3.vhd \

f4.vhd -ENDMAPFile
```

You can specify multiple aliases (or names) for a

library using the equal sign (=). The real library

should be specified last. For example, the following

command reads in test.vhd and includes it in lib1; lib2

and lib3 are aliases of lib1:

```
read design -vhdl -mapfile \

lib3=lib2=lib1 test.vhd
```

In this example, lib1 is the real library; lib2 and

lib3 are merely aliases

`-ENDMAPFile`	Specifies the end of the `-MAPFile` file list.
`-MAPRecursive <library_name> <library_path>`	This option has the same function as `-Map`, but it searches for all VHDL or Verilog files recursively down to the subdirectories of the `<library_path>`. `-Map` searches VHDL or Verilog files under the `<library_path>` and will not search any VHDL files under the subdirectories of `<library_path>`.
`-MErge BBox`	Replaces all blackboxed modules in the design space with modules in the library space.
`-MIXvlog`	Automatically reads Verilog source file in Verilog-1995, Verilog2K, or SystemVerilog standard according to the file extension. Any other extensions not specified by SET RTL TYPE command will be read according to the language option specified in `READ DESIGN` command.
`-NETLIST`	Specifies that HDL files conform to structural Verilog-1995. Using this switch will force `READ DESIGN` command to enable -v1995.
`-NOELaborate`	Reads in multiple files of different languages.
`-NOKeep_unreach`	Remove any unreachable DFF or D-Latch in a module during RTL synthesis. *This is the default.*
`-Keep_unreach`	Keeps any unreachable DFF or D-Latch in a module during RTL synthesis.
`-NOREName`	Do not rename duplicate pin/port names. For example, without this option, the duplicate port `dout` will be renamed to `Port4`:

```
module test ( dout, clk, din, dout );
  input clk, din;
  output dout;
endmodule
```

`-NOSTRength`	Specifies that drive strengths are automatically ignored without being removed from design.
`-NOZPUSHing`	Do not push tristate toward module output. *This is the default.*
`-ZPUSHing`	Pushes tristate toward module output.

`-OPTimize`

Optimizes redundant logic (in library cells) that can affect the way Conformal interprets the design. *This is the default.*

Note: Using this option does not *always* optimize *all* redundant logic.

See the following example:

```
read design
file1 -nooptimize
file2 -optimize
file3 -nooptimize \
-replace
```

`-NOOPTimize`

Preserves redundant logic in Library cells.

See the example listed above (`-optimize`)

`-PARAmeter [-INT | -STR | -ENUM | -TYPE] <parameter_name> <value>`

Assigns design parameters or replace existing design parameters. To specify multiple parameters, use the `-parameter` option for each parameter you want to set. For example:

```
read design filename -parameter parm1 value1 \
-parameter -int parm2 value2
```

This option applies to both Verilog and VHDL files. (Combine with `-root`)

When using the `-parameter -int <parameter_name> <value>` command, the `<value>` will be converted to integer value, which can be a positive integer (`1`), negative integer (`-1`), an integer value recognized as a string (`"1"/"-1"`), or a Verilog style integer (`"16'h0001"`). When using a Verilog style integer, the value must be specified between double-quotes (" ").

When using the `-parameter -str <parameter_name> <value>` command, the `<value>` will be saved as a string.

When using the `-parameter -enum <parameter_name> <value>` command, the `<value>` will be converted to a VHDL enumeration literal. For example, the following command sets the parameter P4 to VHDL enumeration literal GREEN:

```
read design -root mod1 filename \
-parameter -enum P4 GREEN
```

	When using the `-parameter -type <parameter_name> <value>` command, `<value>` is a Verilog/SystemVerilog built-in type or user-defined type compiled in the compilation unit scope `$unit`: types defined in a module or package are not supported. For example, the following command sets the parameter P2 to SystemVerilog `shortint` type:
	`read design -root mod1 filename -parameter \`
	`-type P2 shortint`
	The format "`$unit::xxx`" is not supported for `<value>`.
	Notes:
	Any value that is not recognized as an unsigned decimal integer value is interpreted as string value.
	If `-int` or `-str` is not specified, then the parameter value will be interpreted as an integer if it is not between double-quotes (" "), and as a string if it is between double-quotes. Therefore, if you want to specify a Verilog format value, it must be between double-quotes and used with the `-int` option.

`-RAngeconstraint`	When a variable is of type integer with a value range, a value check is made against the range. If a value is out of range, it will be interpreted as don't care. *This is the default.*
	For example:
	`variable v : integer range 3 to 5;`
	With `-RAngeconstraint`, `v` will be interpreted as:
	`((v>=3 && v<=5)? v : 3'bx)`
	Note: This option applies only to VHDL designs.

`-NORAngeconstraint`	Specifies that no value check is made against the integer value range. **Note:** This option applies only to VHDL designs.
`-REPlace`	Removes all designs that were previously read in, and replaces them with the specified design.

`-APPend`	Appends the design to the one that was previously read. For example, you can use this option to fix a top module and then read it in again without parsing the entire design file again: `read design top.v -append -lastmod` **Note:** The top module cannot pass parameters to modules that are read in previously.
`-ROot <module_name>`	The specified module is the top root module.
`-ROOTConfig <configuration_name>`	The design includes the specified configuration for the top-level module. Use this option when the design includes multiple configurations for the top-level module. When you use the `-rootconfig` option, you must also use the `-root module_name` option (above).
`-ROOTONLY`	Elaborates the root module only and skips elaboration of the other modules that are not instantiated from the root module. Because the elaboration stage is skipped for the uninstantiated modules, this option reduces memory usage and omits the elaboration time error checking.
`-SEnsitive`	Specifies that the design is case sensitive. The default case sensitivity is set by the input language. For example, the default case sensitivity for VHDL input is `-NOSEnsitive`, and the default case sensitivity for Verilog input is `-SEnsitive`.
`-NOSEnsitive`	Specifies that the design is not case sensitive.
`-STATEtable`	Enables support for Synopsys Liberty state tables. *This is the default.* **Note:** This option supersedes the SET STATETABLE command.
`-NOSTATEtable`	Disables support for Synopsys Liberty state tables.
`-SUMmary_report`	Print out HDL rule check messages while reading the library or design file(s). *This is the default*.

`-NOSUMmary_report`	Do not print out HDL rule check messages while reading the library or design file(s).
`-SUPPLY`	Keeps all Verilog `supply0` and `supply1` type nets unchanged. *This is the default for all Conformal tools.*
`-NOSUPPLY`	Converts the Verilog `supply0` and `supply1` type nets to Verilog wire type nets.
`-UNCompress <zip_file_name>`	Reads in the specified compressed file. By default, the Conformal software uses the `gunzip` tool to uncompress the file into the `/tmp` directory. Files created with `gzip` or `compress` are supported. If the compressed file cannot be uncompressed using the `gunzip` tool, you specify another tool by setting UNIX variable `CONFORMAL_UNCOMPRESS` You can also set the UNIX variable `CONFORMAL_TMP` to a path other than the default `/tmp` directory.
`-UNZip <zip_file_name ...>`	This option has the same function as `-UNCompress` except that it can include a list of filenames. For example: `read design fileABC -UNZ zip1 zip2 zip3` The list of filenames end when a subsequent option is specified, or if it is at the end of the command line. **Note:** Specifying `-uncompress` or `-unzip` is optional for gzipped files because the Conformal software can automatically recognize the file type created by the `gzip` tool. This includes command file lists specified with `-file`
`-VErilog`	Specifies that this design is a Verilog design. (Use this option for Verilog designs that comply with IEEE 1364-2001.) *This is the default.*
`-VERILOG2K`	Specifies that this design is a Verilog2K design (Use this option for Verilog designs that comply with IEEE 1364-2001).
`-V1995`	Specifies that this design is a Verilog-1995 design (Use this option for Verilog designs that comply with IEEE 1364-1995).
`-SYStemverilog`	Specifies that this design is a SystemVerilog design.
`-SVA`	Enables SystemVerilog Assertion (SVA) support.

`-SV09`	Specifies that this design is a SystemVerilog 1800-2009 design.
`-VHdl`	Specifies that this design is written in VHDL with the specified standard:

`93`	VHDL-93 (Use this option for VHDL designs that comply with IEEE Std 1076-1993.) *This is the default.*
`87`	VHDL-87 (Use this option for VHDL designs that comply with IEEE Std 1076-1987.)
`2008`	VHDL-2008 (Use this option for VHDL designs that comply with IEEE Std 1076-2008.) VHDL-2008 cannot be used with other versions of VHDL as some standard libraries are defined differently.

Note: The Conformal software supports multiple uses the `-vhdl 93` and `-vhdl 87` options. See the example.

`-SPice`	Specifies that the design is a SPICE netlist design. **Note:** This option requires a GXL license.
`-NO_RESistor`	Removes resistor instances specified in the SPICE netlist.
`-Ndl`	Specifies that the design is an NDL design.
`-EDIF`	Specifies that the design is an EDIF design.
`-LIBErty`	Specifies that the design has a Liberty library format type. Use this option to qualify the library as Liberty.
`-VHDLESCaped_to_verilog`	Specifies that if the content between '\' pairs does not contain any white space, the new name is the escaped content. For example, `\A_B_C_\` changes to `\A_B_C_`. In all other cases, the VHDL escaped name is unchanged. For example, `\1 2\` is unchanged.
`-VMEM_LIB`	Reads in the RTL model containing the Conformal Memory Primitive. This option is a Conformal GXL option.

	Without the `-vmem_lib` option, Conformal does not recognize the Conformal Memory Primitive.
`-VMEM_ULTRA`	Reads in the RTL model containing the Conformal Memory Primitive for checking by Conformal XL. This option requires a Conformal XL license. When you use this option, no debugging is allowed. Use the `-vmem_lib` option for memory verification.
`-VERBose`	Displays the verbose messages of parsing and translating each design module.
`-Golden`	Designates this design Golden. *This is the default.*
`-REVised`	Designates this design Revised.

Table 2-1 Supported Verilog Command-File Options

The following table lists the Verilog command options that Conformal supports.

`<file>`	(Design data file)	A list of the design files.
`-v <file>`	(Library file)	A list of library files. Refer to "Search Path Order when using Verilog Command Options," following this table for more information.
`-y <directory>`	(Library directory)	A list of library directories. Conformal searches for modules not defined in the design files and reads in these modules as library modules. Refer to "Search Path Order when using Verilog Command Options," following this table for more information.

`+incdir+<dirname>...`	(Include directories)	A list of directories used to resolve the Verilog files that have `` `include`` directives. Refer to "Search Path Order when using the `include Directive," following this table for more information.
`+libext+<extension>...`	(Library extensions)	A list of library extensions you can include when using the `-y` option. *The default is* `.v`.
`+define+<macro_name> ...`	(Define macros)	A list of macro names you can include in the `` `define`` statement
`-yd <directory>`	(Design directory)	A list of directories. Conformal searches for modules that are not defined in the design files and reads these modules in as design modules. Refer to "Search Path Order when using Verilog Command Options," following this table for more information.
`+search_yd_path`		Enable Conformal to search for files in the directories specified by the `-yd` option.
`-f <file>`	(another command file)	

Search Path Order when using Verilog Command Options

Verilog command options used to resolve Verilog module references (such as `-y`/`-yd`, `-v`/`-vd`, and `+libext+<ext_suffix>`) can also affect the search path order. When a Verilog module instance cannot be resolved to any existing modules in memory, the Verilog parser searches for the module from the directories (`-y`/`-yd`) or files (`-v`/`-vd`) specified by these options. The order in which module references are resolved is the order specified in the Verilog command file.

To search for files in the directories specified by the `-yd` option in the Verilog command file, use the `+search_yd_path`. The additional search paths specified by `-yd` are effective only in the current `READ DESIGN` command.

For example, a Verilog command file named `gol.f` contains:

```
+search_yd_path
+incdir+include
-yd hdl
-yd test
mux.vs
```

In LEC, `READ DESIGN -f gol.f` will search for `mux.vs` in the following directories (in the order listed):

- Current working directory "."
- `-yd hdl`
- `-yd test`
- Any other paths specified by ADD SEARCH PATH

Search Path Order when using the `include Directive

When a Verilog include file is specified using `include directive, the default file search order is as follows:

1. Current working directory
2. Directories specified by `+incdir` option in the Verilog command file
3. Directory that contains the file that specifies the `include directive
4. Search paths added by the ADD SEARCH PATH command
5. Directories specified by `-yd` in the Verilog command file
6. Directories specified by `-y` in the Verilog command file

When using the command SET HDL OPTIONS `-INCLUDE_SRC_DIR OFF` (note that the default is `ON`), which disables and prevents LEC from reading include files in the source's relative path, the search order becomes as follows.

1. Current working directory
2. Directories specified by `+incdir` option in the Verilog command file
3. Search paths added by the ADD SEARCH PATH command
4. Directories specified by `-yd` in the Verilog command file
5. Directories specified by `-y` in the Verilog command file

To customize the search order, use the SET HDL OPTIONS `-verilog_include_dir` command option.

Table 2-2 Supported VHDL Command-File Options

The following table lists the VHDL command options that Conformal supports.

`<file>`	Individual design file
`-map <libname> <dirname>`	Map logical library name to a directory
`-mapfile <libname> <filename ...>`	Map logical library name to a list of files
`-library <libname> <dirname>`	Map logical library name to a directory

The VHDL command file options behave the same way as if they are specified as command line parameters. Refer to the **Parameters** section for details of each of the option.

Examples

The following example demonstrates how to use the backslash character to show that your command continues on the next line.

```
read design tran1.spi tran2.spi tran3.spi \
-spice -revised
```

In the following example, only `ent2.vhdl` and `arch2.vhdl` will be parsed according to VHDL 87 syntax rules. All the other files will be parsed according to VHDL 93 syntax rules.

```
read design ent0.vhdl arch0.vhdl \
-vhdl 93 ent1.vhdl arch1.vhdl \
-vhdl 87 ent2.vhdl arch2.vhdl \
-vhdl ent3.vhdl arch3.vhdl
```

In the following example, you can use `read design -localref` command to use local module `sub1` (see line10 of each file).

For each file, the instantiation of `sub1` in line10 will use the `sub1` defined in line `1` (`module sub1(aa, oo)`).

File mod1.v

```
1 module sub1(aa, oo);
2 input aa;
3 output oo;
4   assign oo = aa;
5 endmodule
6
```

```
module mod1(aa, oo);
input aa;
output oo;
  sub1 u0 (aa, oo);
endmodule
```

File mod2.v

```
module sub1(aa, oo);
input aa;
output oo;
  assign oo = !aa;
endmodule

module mod2(aa, oo);
input aa;
output oo;
  sub1 u0 (aa, oo);
endmodule
```

Related Commands

ADD NOTRANSLATE MODULES

ADD RENAMING RULE

ADD SEARCH PATH

DELETE NOTRANSLATE MODULES

DELETE RENAMING RULE

DELETE SEARCH PATH

READ LIBRARY

REPORT DESIGN DATA

REPORT MESSAGES

REPORT MODULES

REPORT NOTRANSLATE MODULES

REPORT RENAMING RULE

REPORT RULE CHECK

REPORT SEARCH PATH

READ EXTENDED MAPPING

```
READ EXtended Mapping
     <filename>
     [-REPlace] [-USE_PINname <Golden|Revised|Both>]
     [-GOLDen_prefix <string>]
     [-REVIsed_prefix <string>]
     [-CHECK_names]
```

This command reads in the extended mapping information for the specified file to help in LEC mapping and verification setup. LEC does not check if the information in the file has conflicts. If you are reading a file which is not generated by `WRITE EXTENDED MAPPING`, use the read command with caution.

This command should be executed after both Golden and Revised designs and libraries are read in.

Tcl Command

```
read_extended_mapping
```

Parameters

`<filename>`	Specifies extended mapping file location. GZIP is supported.
`-REPlace`	Replaces the whole existing extended mapping information in LEC. If `read_extended_mapping` is executed multiple times without `-REPlace` option, LEC will keep appending information into the database. If there is a conflict, LEC overwrites the old information.
`-USE_PINname`	Specifies if instance names in Golden, Revised, or Both are pin names. This option allows LEC to use the pin names to distinguish different registers in a multi-bit cell. Use this option when sequential elements are represented by pin names.

`-GOLDen_prefix <string>`

	Appends this Golden prefix string to the instance names in the extended mapping file. This option lets Conformal read the extended mapping file for a higher-level module containing hierarchy.
`-REVIsed_prefix <string>`	
	Appends this revised prefix string to the instance names in the extended mapping file. This option lets Conformal read the mapping file for a higher-level module containing hierarchy.
`-CHECK_names`	Performs a check on the extended mapping file. A warning will be generated if LEC cannot find the pin or instance specified in the file.

Related Commands

ANALYZE EXTENDED MAPPING

WRITE EXTENDED MAPPING

READ FSM ENCODING

```
READ FSm Encoding
     <filename>
     [-Golden | -Revised]
     (Setup Mode)
```

Directs Conformal to read in a file that defines new Finite State Machine (FSM) encoding.

By default, Conformal reads binary encoding when building an FSM. Therefore, if your gate netlist uses different encoding (for example, one-hot), you must use the `READ FSM ENCODING` command to specify the correct encoding. See the example of an FSM encoding file below.

Tcl Command

```
read_fsm_encoding
```

Parameters

`<filename>`	Reads in the specified file. This option is a required filename that contains the encoding differences.
`-Golden`	Uses this library with the Golden design. *This is the default.*
`-Revised`	Uses this library with the Revised design.

Examples

The following example shows the need for the `READ FSM ENCODING` command. In this case, the user was alerted to encoding differences during mapping. The Golden design showed two registers, while the Revised showed four registers. The following is an example of an FSM encoding file that replaces the binary encoding with one-hot encoding:

```
.fromstates current_state_reg[1] current_state_reg[0]
.tostates current_state_reg[3] current_state_reg[2] current_state_reg[1]
current_state_reg[0]
.begin
00 0001
01 0010
10 0100
11 1000
.end
```

In this example, between `.begin` and `.end:`

- `.fromstates` are the left-hand side states
- `.tostates` are the right-hand side states

READ GUIDE FILE

```
READ GUide File
     <file ...>
     (Setup Mode)
```

Reads in the OVF guide file. Synthesis tools generate a guide file to record information, such as instance or module naming, sequential element merging or duplication, module ungrouping or unification, and datapath logic transformation. LEC uses this information for setup, hierarchical comparison, abort resolving, or comparison performance.

Read in the guide file before you read in the Golden RTL design; information in the guide file might be referenced during the LEC-RTL interpretation and hierarchy build-up stage.

The `READ GUIDE FILE` command takes a list of guide files (reading them in the order specified), as guide information is sometimes recorded into separate files. Take note of any dependencies in the content of the guide files, as you might need to specify the files in a specific order.

Tcl Command

```
read_guide_file
```

Parameters

`file ...`	Specifies the guide file, or list of guide files.

Related Commands

APPLY GUIDED TRANSFORMATIONS

REPORT GUIDE INFORMATION

READ GUIDANCE INFORMATION

```
READ GUidance Information
     <file>
     [-APPLY_PIN]
     [-NOExact_pin_mapping]
     [-COMMIT|-NOCOMMIT]
     (Setup / LEC Mode)
```

The automatic flow to apply the `SETUP_NAME` command to enable the mapping and modeling guidance.

The `<file>` specified will be used in `SETUP_NAME -source <file>`.

Tcl Command

```
read_guidance_information
```

Parameters

`-APPLY_PIN`	With the option `-apply_pin` it needs more memory for computation. The pin-mapping by using the `SETUP_NAME` is enabled when `-APPLY_PIN` is used.
`-NOExact_pin_mapping`	Apply the pin mapping information for mismatched pin-mapped modules. The default pin mapping information is applied if the module has the matched pin or extra in one side only. If the module has the mismatched pin, we would not apply it in default.
`-COMMIT`	Apply `SETUP_NAME -commit` at the end of the process. It releases the naming DB.
`-NOCOMMIT`	Do not apply `SETUP_NAME -commit` in the end of command. The Naming DB keeps and the user can issue the `SETUP_NAME`. This is used often when users want to debug the mapping quality.

Related Command

SETUP NAME

READ IMPLEMENTATION INFORMATION

```
READ IMplementation Information
     <<implementation_information_directory>
     [-Golden <label>] -REVised <label>>
     | <[-Golden <json filename>] -REVised <json filename>]
     [-IGNORE <ALL | PHASE| SEQ_Constant | SEQ_Merge
     | SEQ_Unreachable | SEQ_Duplication | BOUNDary_optimization
     | SYNTHesis_equation | LOOP_Breaker> ...]
     [-NOUSE_RTL_NAMEs | -USE_RTL_NAMEs]
     [-SCOPE]
     (Setup/LEC Mode)
```

This command reads in verification-related synthesis information that can help improve verification setup. This command can only be used after the `SET VERIFICATION INFORMATION` command. Currently, this command can only read from the Genus generated `fv` directory.

For information on how to use this command when verifying Genus synthesized netlists with sequential phase inversion optimization, go to the web interface by using the `set web_interface ON` command.

Tcl Command

`read_implementation_information`

Parameters

`<implementation_information_directory>`	
	Name of the directory from which to read the implementation information. Currently, this command can read from only the Genus generated `fv/*` directory.
`-Golden <label>`	Specifies the label of the set of information for the golden netlist.
`-REvised <label>`	Specifies the label of the set of information label for the revised netlist.
`-Golden <json filename> | -REVised <json filename>`	

	Directly specifies the locations of the json files after `-revised` or `-golden`. The filename must include the full pathname to each file. Gzipped files are supported. The implementation information directory argument must be omitted.
`-IGNORE`	Ignores the specified implementation information. When this option is specified, the specified types information will be skipped during the reading of information and the responding information total counts in the summary table will be 0. `ALL`: All types of information `PHASE`: Sequential phase for mapping `SEQ_Constant`: Sequential constant `SEQ_Merge`: Sequential merge `SEQ_Unreachable`: Sequential unreachable `SEQ_Duplication`: Sequential duplication `BOUNDary_optimization`: Boundary optimization `SYNTHesis_equation`: Datapath synthesis equation `LOOP_Breaker`: Combinational loop breaker
`-NOUSE_RTL_NAMEs`	When specified, LEC uses netlist names to reference the optimized registers in the netlist, instead of the RTL names.
`-USE_RTL_NAMEs`	When specified, LEC uses RTL names to reference the optimized registers in the netlist, instead of the netlist names.

`-SCOPE`	Leverages the verification information of sub-modules with the following:
	`SETUP> REAd IMplementation Information -scope [top instance path] ...`
	`SETUP> REAd VErification Information -scope [top instance path] ...`
	The [path] is the top instance path that would be used to modify the verification information. If there are more files that need to be read or more instances to be applied, use this command multiples times.
	`REAd IMplementation Information -scope [path 1] <dir 1>`
	`REAd IMplementation Information -scope [path 2] <dir 2>`

Related Commands

SET VERIFICATION INFORMATION

WRITE VERIFICATION INFORMATION

READ KEYPOINT MAPPING

```
READ KEypoint Mapping
     <-GOLDEN_REVISED|-REVISED_GOLDEN> <key_point_mapping_file1>
     [<key_point_mapping_file2> ...]
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

This command reads in the key point file(s) generated by WRITE MAPPED POINT command to assist supply mapping in power grid comparison.

Tcl Command

`read_keypoint_mapping`

Parameters

`<key_point_mapping_file>`	Specifies the key point mapping file.
`-GOLDEN_REVISED`	Specifies the key points specified in the file are mappings from the golden design to the revised design.
`-REVISED_GOLDEN`	Specifies the key points specified in the file are mappings from the revised design to the golden design.

Examples

```
read keypoint mapping -golden_revised mapfile
```

READ LEF FILE

```
READ LEF File
     <filename>
     [-MACRO_PINS_ONLY]
     [-SUMmary_report | -NOSUMmary_report]
     [-GOLden | -REVised | -BOTh]
     (Setup Mode)
```

Reads in the Library Exchange Format (LEF) file, which contains the design's library information.

During design elaboration, Conformal checks whether each cell used in the specified design (Golden, Revised, or both) has a corresponding cell in the library. This command reads in the LEF file and checks that the port declarations are in both the LEF and library. If a power/ ground port declaration exists only in the LEF file, Conformal will add the port to the library cell.

Tcl Command

`read_lef_file`

Parameters

`<filename>`	Specifies the name of the LEF file.
`-MACRO_PINS_ONLY`	Reads in the LEF file, but uses a line-by-line parser to report information only on macros and pins.
`-SUMmary_report`	Print out HDL rule check messages while reading the library or design file(s). *This is the default*.
`-NOSUMmary_report`	Do not print out HDL rule check messages while reading the library or design file(s).
`-Golden`	Reads in the LEF file for the Golden design. *This is the default.*
`-Revised`	Reads in the LEF file for the Revised design.
`-BOTh`	Reads in the LEF file for both the Golden and Revised designs.

READ LIBRARY

```
READ LIbrary
     <filename* ...>
     [-VErilog | -VERILOG2K | -V1995 | -SYStemverilog | -SVA | -SV09
       | -VHdl [93 | 87 | 2008] | -Liberty [-LP [ALL | STD] [1.1 | 2.0]][-PG_PIN]
       | -CPF [-LP [STD | ALL][1.1 | 2.0]]]
     [-BBOXSolver]
     [-CONFiguration | -NOCONFiguration]
     [-Define <variable_name>]
     [-EXClude  <exclude_file_name*>]
     [-EXtract]
     [-File <command_filename>]
     [-KEEP_ESCAPED_ID]
     [-LAstmod]
     [-LOGIC_ENCODING_OFF]
     [-Map <library_name> <library_path>]
     [-MAPFile <library_name> [=<library_name2>] <filename ...>]
     [-MAPRecursive <library_name> <library_path>]
     [-MULTIPLE_LIBraries]
     [-NOELaborate]
     [-NOKeep_unreach | -Keep_unreach]
     [-NOShare]
     [-NOSTRength]
     [-OPTimize | -NOOPTimize]
     [ | -REPlace | -APPend]
     [-RESPECT_timing_type_for_flop]
     [-SEnsitive | -NOSEnsitive]
     [-STATEtable | -NOSTATEtable]
     [-SUMmary_report | -NOSUMmary_report]
     [-SUPPLY | -NOSUPPLY]
     [-UNCompress <zip_file_name>]
     [-UNZip <zip_file_name ...>]
     [-Both | -Golden | -Revised]
     [-VERBose]
     (Setup Mode)
```

Reads in the library model descriptions for Verilog, VHDL, or Liberty designs. The library is either a Verilog simulation library or a Synopsys Liberty library. It is read for the Golden, Revised, or both designs.

Note: For RTL to gate formal equivalence checking, use simulation libraries instead of synthesis libraries because design verification signoff happens for simulation libraries: not for synthesis libraries.

The `READ LIBRARY` command must be used *before* the READ DESIGN command if the design is Verilog, VHDL, or Liberty.

Note: Library in this context refers to the technology library, such as ASIC cell and memory definitions. See READ DESIGN for information on reading VHDL libraries and packages.

Important

- If there are duplicate modules, the Conformal software uses the first module and ignores later ones. However, you can use the `-lastmod` option to specify that Conformal use the last module and ignore earlier ones.
- Use the backslash character (\) at the end of a line to show that the command you are entering continues on the next line.
- Use the tilde character (~) to shorten the path of the file.

VHDL and Verilog File Encryption

Conformal tools support Verilog and VHDL files which are encrypted by the NC-Protect and Cadence encryption Cadence tools. NC-Protect is available from Cadence NC-VHDL and Verilog simulators. Cadence encryption is Cadence proprietary tool.

The Cadence Verilog-XL protected files are for simulations only and are unsupported by Conformal tools. Other proprietary encryption files are unsupported by Conformal tools.

Tcl Command

```
read_library
```

Parameters

`<filename* ...>`	Reads in the specified file(s). This option is a required filename that contains the Verilog simulation library or the Synopsys Liberty library. This accepts wildcards.
`-VErilog`	Contains the Verilog library model descriptions. *This is the default.* **Note:** This supports NC-Protect and Cadence encryption.
`-VERILOG2K`	Contains Verilog2k library model descriptions.
`-V1995`	Contains Verilog-95 library model descriptions.

`-SYStemverilog`	Contains SystemVerilog library model descriptions.
`-SVA`	Enables SystemVerilog Assertion (SVA) support.
`-SV09`	Contains SystemVerilog 1800-2009 library model descriptions.
`-VHdl`	Specifies that the library is written in VHDL-2008. The VHDL file is of the specified standard:

`93`	VHDL-93 (Use this option for VHDL designs that comply with IEEE Std 1076-1993.) *This is the default.*
`87`	VHDL-87 (Use this option for VHDL designs that comply with IEEE Std 1076-1987.)
`2008`	VHDL-2008 (Use this option for VHDL designs that comply with IEEE Std 1076-2008.) VHDL 2008 cannot be used with other versions of VHDL as some standard libraries are defined differently.

Note: The VITAL format is unsupported.

Note: This supports NC-Protect and Cadence encryption.

`-Liberty` Specifies that the library filename is in the Synopsys Liberty format. This file can contain functional and low power attributes.

Notes:

Supports Cadence encryption.

Supports LDB library files. LDB files contain binary database information and compressed Liberty source text. Conformal will independently parse the compressed Liberty source text portion of the LDB file.

The `-LP` option is for Conformal Low Power. This option extracts low power attributes into library power intent database.

`-LP ALL`	Use Liberty attributes to extract the standard cells (isolation, level shifter, always on, retention, power switch, ground switch) and the IO Pad, and macro model cells into the Conformal Low Power library and macro model power intent database. *This is the default*. For example, when you specify only `read library -liberty -lp`, the tool uses `ALL` by default.
`-LP STD`	Use Liberty attributes to extract standard low power cells (isolation, level shifter, always on, retention, power switch, and ground switch cells) into the Conformal LP library power intent database.
`1.1`	Map the extracted cells to CPF 1.1 features. *This is the default*.
`2.0`	Map the extracted cells to CPF 2.0 features.
`-PG_PIN`	Extracts the Liberty `pg_pin` group as module ports. This is recommended when you have instances with power and ground ports. Note: This option is not required when using low power equivalency checking or low power design checks as these features decide the power/ground pin extraction automatically based on the low power option setup.

`-CPF`

Specifies that filename is a CPF file which contain references to Liberty file paths defined by CPF library set objects (`library_set`).

The `-LP` option is for Conformal Low Power. This option allows extraction of low power cells defined by Liberty attributes.

`-LP STD`	Extracts standard low power cells (isolation, level shifter, always on, retention, power switch, and ground switch cells) into the Conformal LP library power intent database. *This is the default*.

`-LP ALL`	Extracts the standard cells (isolation, level shifter, always on, retention, power switch, ground switch) and the IO Pad, and macro model cells into the Conformal Low Power library and macro model power intent database.
`1.1`	Map the extracted cells to CPF 1.1 features. *This is the default*.
`2.0`	Map the extracted cells to CPF 2.0 features.

`-BBOXSolver` When there are multiple modules with the same name, skip the blackbox modules and only select non-blackboxed modules.

`-CONFiguration` Supports VHDL configuration constructs. *This is the default.*

`-NOCONFiguration` Do not interpret VHDL configuration.

`-Define <variable_name>`

Defines `` `ifdef `` variable names in Verilog.

The macro redefinition in Verilog source code is ignored if the text macro is specified at the Conformal command line or Verilog command file.

`-EXClude <exclude_file_name*>`

Specifies files to exclude when reading in the library. This accepts wildcards.

Note: You cannot use multiple wildcards with this option.

`-EXtract` Abstracts the gate information from any transistor library models.

Note: This option requires a Conformal GXL license.

`-File <command_filename>`

Reads in the specified command file as a library.

Note: This option is for Verilog or VHDL command file lists. It also supports Liberty command files (-v/-y commands are not supported in Liberty command files).

`-KEEP_ESCAPED_ID` When used, this option keeps escaped identifiers, as in Verilog-2001. See "Support for Macro Expansions" in the *Conformal Equivalence Checking User Guide* for more information.

`-Keep_unreach`	Keeps any unreachable DFF or D-Latch in a module during RTL synthesis.
`-LAstmod`	If duplicate modules exist, Conformal uses the first module and ignores the later ones by default. Use this option to specify that Conformal use the last module and ignore earlier ones.

`-LOGIC_ENCODING_OFF`	Do not interpret the user-defined `enum` types with literal value `0`, `1`, `X`, `Z`, and so on as one-bit logic values.
	For example, `MY_MVL` is the user-defined `enum` type:
	`type MY_MVL is ('X', '0', '1', 'Z');`
	The type of signal `val` is `MY_MVL`. For the statement:
	`val <= 'Z';`
	With this option, `val` is `2'd3`; without this option, val is `1'bx`.

`-Map <library_name>`	`<library_path>` Reads in files for the specified `library_name` from `library_path`. Use this option to read in all of the VHDL or Verilog files in the specified library path for the given library name. You can also map multiple directories to a single library. For example:

```
read library -vhdl top.vhd -map mylib /design/path1 \
  -map mylib /design/path2
```

`-MAPFile <library_name>[=<library_name2>] <file_name ...>`

Reads in the specified files and include them in the library.

Use this option to specify the files that belong to a given library. The file list terminates with the next option or the end of the `READ LIBRARY` command.

You can also use multiple `-mapfile` options to specify multiple files in a library.

For example, the following two commands are the same:

```
read library -vhdl top.vhd -mapfile mylib x1.vhd \
-mapfile mylib x2.vhd
```

```
read library -vhdl top.vhd -mapfile mylib x1.vhd x2.vhd
```

You can specify multiple aliases (or names) for a library using the equal sign (=). The real library should be specified last. For example, the following command reads in `test.vhd` and includes it in `lib1`; `lib2` and `lib3` are aliases of `lib1`:

```
read library -vhdl -mapfile lib3=lib2=lib1 test.vhd
```

In this example, `lib1` is the real library; `lib2` and `lib3` are merely aliases.

`-MAPRecursive <library_name> <library_path>`

This option has same function as `-Map`, but it searches for all VHDL or Verilog files recursively down to the subdirectories of the `<library_path>`.

`-Map` searches VHDL or Verilog files under the `<library_path>` and will not search any VHDL or Verilog files under the subdirectories of `<library_path>`.

`-MULTIPLE_LIBraries` If duplicate modules exist, Conformal uses the first modules and ignores the later ones by default. Use this option to specify that Conformal store duplicated library modules if they are in a different library.

`-NOELaborate` Reads in multiple files of different languages. With this option, you can defer the binding of entity or module instantiations.

This is for cases when you have mixed library files in VHDL and Verilog languages, where a VHDL entity in one library file instantiates a Verilog module, and a Verilog module in another library file instantiates a VHDL entity.

`-NOKeep_unreach`	Remove any unreachable DFF or D-Latch in a module during RTL synthesis. *This is the default.*
`-NOShare`	Do not share the library files for both the Golden and Revised designs, and appends the library files to both designs.
	When this option is used together with the default `-Both` option, this is the equivalent of running the following two commands:

```
read library -golden <filenames>
read library -revised <filenames>
```

This is the default.

`-NOSTRength`	Specifies that drive strengths are automatically ignored without being removed from library.
`-OPTimize`	Removes redundant buffers in library cells. *This is the default.*
	Note: Using this option does not *always* optimize *all* redundant logic.
	See the following example:

```
read library
file1 -nooptimize
file2 -optimize
file3 -nooptimize \
-replace
```

`-NOOPTimize`	Preserves redundant buffers in library cells.
	See the example listed above (`-optimize`).
`-REPlace`	Replaces the existing library. The designs are also deleted.
`-APPend`	Appends this library to the one that was previously read.
	For example, you can use this option to fix a top module and then read it in again without parsing the entire library file again:

```
read library top.v -append -lastmod
```

Note: The top module cannot pass parameters to modules that are read in previously.

`-RESPECT_timing_type_for_flop`

If the timing type and the clock phase conflict with the master-slave flip-flop in Liberty file, this honors the timing type as the clock phase of the flip-flop.

`-SEnsitive`	Regards the library model descriptions as case-sensitive. The default case sensitivity is set by the input language. For example, the default case sensitivity for VHDL input is `-NOSEnsitive`, and the default case sensitivity for Verilog input is `-SEnsitive`.
`-NOSEnsitive`	This library model's descriptions are not case sensitive.
`-SUMmary_report`	Print out HDL rule check messages while reading the library or design file(s). *This is the default*.
`-NOSUMmary_report`	Do not print out HDL rule check messages while reading the library or design file(s).
`-STATEtable`	Enables support for Synopsys Liberty state tables. *This is the default.* **Note:** This option supersedes the `SET STATETABLE` command.
`-NOSTATEtable`	Disables support for Synopsys Liberty state tables.
`-SUPPLY`	Keeps all Verilog `supply0` and `supply1` type nets unchanged. *This is the default.*
`-NOSUPPLY`	Converts the Verilog `supply0` and `supply1` type nets to Verilog wire type nets.

`-UNCompress <zip_file_name>`

Reads in the specified compressed file. By default, the Conformal software uses the `gunzip` tool to uncompress the file into the `/tmp` directory. Files created with `gzip` or `compress` are supported.

If the compressed file cannot be uncompressed using the `gunzip` tool, you specify another tool by setting UNIX variable `CONFORMAL_UNCOMPRESS`

You can also set the UNIX variable `CONFORMAL_TMP` to a path other than the default `/tmp` directory.

`-UNZip <zip_file_name ...>`	This option has the same function as `-UNCompress` except that it can include a list of filenames. For example: `read library fileABC -UNZ zip1 zip2 zip3` The list of filenames end when a subsequent option is specified, or if it is at the end of the command line. **Note:** Specifying `-uncompress` or `-unzip` is optional for gzipped files because the Conformal software can automatically recognize the file type created by the `gzip` tool. This includes command file lists specified with `-file`.

`-Both`	Uses this library for both the Golden and Revised designs. *This is the default.*
	When you use this command, the tool issues the following commands, but the second command's parsing message is hidden:
	`read library -golden libraryname -liberty`
	`read library -revised libraryame -liberty`

`-Golden`	Uses this library with the Golden design.
`-Revised`	Uses this library with the Revised design.
`-VERBose`	Displays the verbose messages of parsing and translating each library module.

Examples

In the following example, `read library mod1.v mod2.v` will select blackbox module `sub1()` from `mod1.v`, and `read library -lastmod mod1.v mod2.v` will select blackbox module `sub2()` from `mod2.v`.

To avoid selecting blackboxes, use `read library -bboxsolver mod1.v mod2.v`, which will select `sub1()` from `mod2.v`, and `sub2()` from `mod1.v`.

```
File mod1.v:

module sub1(aa, oo);
 input aa;
 output oo;
 // This is a blackbox
endmodule
```

```
module sub2(aa, oo);
 input aa;
 output oo;
 assign oo = aa;
endmodule

File mod2.v:
module sub1(aa, oo);
 input aa;
 output oo;
 assign oo = !aa;
endmodule
module sub2(aa, oo);
 input aa;
 output oo;
 // This is a blackbox
endmodule
```

Related Commands

ADD NOTRANSLATE MODULES

ADD RENAMING RULE

ADD SEARCH PATH

DELETE NOTRANSLATE MODULES

DELETE RENAMING RULE

DELETE SEARCH PATH

READ DESIGN

READ POWER INTENT

REPORT DESIGN DATA

REPORT MESSAGES

REPORT MODULES

REPORT NOTRANSLATE MODULES

REPORT RENAMING RULE

REPORT RULE CHECK

REPORT SEARCH PATH

SET DIRECTIVE

SET RULE HANDLING

SET STATETABLE

SET UNDEFINED CELL

WRITE HIER_COMPARE DOFILE

WRITE LIBRARY

READ MAPPED POINTS

```
READ MApped Points
     <filename>
     [-EXACT_name]
     [-GOLDen_prefix <string>]
     [-REVIsed_prefix <string>]
     [-SEARCH]
     (LEC Mode)
```

Reads in the mapped point information you created with the `WRITE MAPPED POINTS` command. By default, Conformal automatically maps key points during the transition from Setup to LEC mode. And if the key points are already mapped, Conformal ignores any mapped point information in the file. Thus, to prevent Conformal from automatically mapping key points during the transition from Setup to LEC mode and enable Conformal to read in mapped point information completely from the file, do one of the following:

- Use the `set flatten model -nomap` command *before* you set the system mode to LEC.
- Use the `set system mode lec -nomap` command to suppress automatic mapping during the transition to LEC mode.

Use the tilde character (~) to shorten the file's path.

Note: Use the Golden and Revised prefix options to specify the hierarchy of the instance name. If a hierarchical submodule's map point information is written to a file, it can be read in at a higher level module or the top root module with the specified hierarchical prefix string.

Tcl Command

```
read_mapped_points
```

Parameters

`<filename>`	Reads mapped point information from the specified file.
`-EXACT_name`	Specifies an exact match for the names specified in the file. This can help speed up the process of reading files where many names are incorrect.

`-GOLDen_prefix <string>`	
	Appends this Golden prefix string to the instance names. This option lets Conformal read the mapped point file for a higher-level module containing hierarchy.
`-REVIsed_prefix <string>`	
	Appends this revised prefix string to the instance names. This option lets Conformal read the mapped point file for a higher-level module containing hierarchy.
`-SEARCH`	Search for the specified gates' drivers to find the corresponding keypoints for mapping.

Related Commands

SET FLATTEN MODEL

WRITE MAPPED POINTS

READ MEMORY PRIMITIVE

```
READ MEmory Primitive
     <<filename1> <filename2 ...>>
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Reads in Verilog memory primitive simulation models that were created using the Conformal memory primitive generator. This command creates a memory-friendly, synthesized view that you can use for comparison with a memory circuit.

Important

Read in designs that use the memory primitive *after* you use this command.

Tcl Command

```
read_memory_primitive
```

Parameters

`<filename1> <filename2 ...>`	Specifies the files to read in.

Related Command

READ DESIGN

READ PATTERN

```
READ PAttern
     <filename>
     [-Verilog | -SPice ]
     [-Golden | -Revised | -Both]
     [-SEnsitive | -NOSEnsitive]
     [-LAstmod]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Reads in the transistor description from a file that Conformal GXL applies to the Golden and Revised designs. (The file format type is either Verilog or SPICE.)

The transistor description represents a pattern that Conformal GXL seeks during abstraction. When Conformal GXL detects a pattern, it substitutes a user-specified functional model. Before this command can be executed correctly, you must provide the user-specified functional model using the `READ LIBRARY` command. The transistor description file and functional model have the same module and port names; port direction can be different, but, neither the transistor description file nor the library database can contain submodules: every cell must be flat.

Use the tilde character (~) to shorten the path of the file.

Tcl Command

`read_pattern`

Parameters

`<filename>`	A required filename. It contains the transistor abstraction information.
`-Verilog`	The transistor description file format is Verilog. *This is the default.*
`-SPice`	The transistor description file format is SPICE.
`-Golden`	Applies the file's sub-circuit information to the Golden design. *This is the default.*
`-Revised`	Applies the file's sub-circuit information to the Revised design.

`-Both`	Applies the file's sub-circuit information to both the Golden and Revised designs.
`-SEnsitive`	The transistor description file is case-sensitive. *This is the default.*
`-NOSEnsitive`	The transistor description file is not case-sensitive.
`-LAstmod`	If duplicate modules exist, Conformal uses the first module and ignores the later ones by default. Use this option to specify that Conformal use the last module and ignore earlier ones.

Examples

```
read pattern DLTCH1.v -verilog -revised
read pattern DLT2.v -verilog -both
read pattern xtran.spi -spice -golden

// Functional description of the transistor
// pattern
read library lib.v -verilog -golden

// Transistor pattern description
read pattern pattern.v -golden
read design design.v -golden
abstract logic -golden
```

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

REPORT CLOCK

REPORT MOS DIRECTION

REPORT NET ATTRIBUTE

REPORT PIN DIRECTION

RESOLVE

SET PATTERN MATCH

READ POWER INTENT

```
READ POWER Intent
     <filename>
     [-CPF | -UPF]
     [-NON_standard]
     [-GOLden | -REVised | -BOTH]
     [-REPlace]

     (1801 Options)
     [-1801]
     [-INSERT_ISOLATION]
     [-GOLden | -REVised | -BOTH]
     [-REPlace]
     (Setup Mode)
```

Note: This is a Conformal Low Power command.

Reads and elaborates design power intent. Use the `READ LIBRARY` command to read in library power intent.

- Conformal Low Power does not support mixed languages. You cannot mix CPF libraries and 1801 power intent.
- Some options are different for the 1801 flow. To enable this flow, use the `SET LOWPOWER OPTION -native_1801` option.
- In the 1801 flow, there are rules that can identify potential setup issues or critical issues in the power intent or design. The rule violations identified by these rules must be resolved before the rest of the rule checks are performed. These critical rules cannot be disabled, downgraded or ignored.
- In the 1801 flow, once reading power intent, all the rule checks will be performed and reported. Verification process is only terminated when there is linter error or critical issue in power intent or design.

If you used the CPF integrator to create integrated power intent, you must read the integrated CPF file back in using the `READ POWER INTENT <file> -replace` command followed by the `COMMIT POWER INTENT` command.

Tcl Command

```
read_power_intent
```

Parameters

`<filename>`	Specifies the name of the power intent file.
`-CPF`	Specifies that the file format is CPF.
`-UPF`	Specifies that the file format is UPF.
	Note: Currently, the tool supports reading UPF version 1.0.
`-NON_standard`	Allows the use of specific commands that are not legal 1801 (UPF), but are used by tools to define power intent features that cannot be specified with orignal UPF commands.
`-GOlden`	Read the power intent settings to the Golden design. *This is the default*
`-REVised`	Read the power intent settings to the Revised design.
`-BOTH`	Read the power intent settings to both the Golden and Revised designs.
`-REPLace`	Replace existing power intent.
(1801 Options)	The following options are for the 1801 flow.
`-1801`	Specifies that the power intent format is 1801/UPF and not the default CPF.
`-INSERT_ISOLATION`	Inserts isolation cells for all isolation strategies that are inserted for equivalence checking. Level shifters, switch cells, and retention are not inserted.
`-GOlden`	Read the power intent settings to the Revised design.
`-REVised`	Read the power intent settings to both the Golden and Revised designs.
`-REPLace`	Replace existing power intent.

Related Commands

COMMIT POWER INTENT

COMPARE POWER INTENT

READ LIBRARY

REPORT COMPARED INTENT

READ ROM PRIMITIVE

```
READ ROM Primitive
     <filename>
     [-CODE_FILE <codefilename>]
     [-CODE_FILE_FORMAT <BIN | HEX>]
     [-APPend]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Reads in Verilog ROM primitive simulation models that were created using the Conformal ROM primitive generator.

Tcl Command

`read_rom_primitive`

Parameters

`<filename>`	Specifies the file to read in.
`-CODE_FILE <codefilename>`	Specifies the code file that will initialize the ROM.
`-CODE_FILE_FORMAT <BIN | HEX>`	Specifies the format of the code file. If you do not specify the `-code_file_format` option, the software uses the format selected during primitive generation.
`-APPend`	Appends the read-in file to ROM primitive database. By default, if multiple ROM primitives are read, the later one replaces the existing ROM primitive.

Related Command

READ DESIGN

READ RULE CHECK

```
READ RULE Check
     <filename> <-EXClude |-INClude>
     [-Design | -Library]
     [-GOLden | -REVised]
     (Setup Mode)
```

Performs incremental rule checks. The first time you run a session, write the rule violations into a rule file using the `write rule check <filename> -golden` (or `-revised`) command. For later runs, exclude the violations already flagged with the `read rule check -exclude <filename>` command.

Use the tilde character (~) to shorten the path of the file.

Tcl Command

`read_rule_check`

Parameters

`<filename>`	Specifies the name of the file that contains rule violations from a previous session.
`-EXClude`	Excludes checks for violations noted in the specified file.This option works as a filter; therefore, use it *after* the `READ DESIGN` command.
`-INClude`	Includes checks for violations noted in the specified file. This option lets you reinstate violations that were previously excluded.
`-DEsign`	Reads only design rule check violations. If you do not specify `-design` or `-library`, Conformal reads rule check violations from both designs and libraries.
`-LIbrary`	Reads only library rule check violations. If you do not specify `-design` or `-library`, Conformal reads rule check violations from both designs and libraries.
`-Golden`	Applies this command to the Golden design. *This is the default.*
`-Revised`	Applies this command to the Revised design.

Examples

In the following example, the second `report rule check` will not report any rules.

```
read design g.v -golden
read design r.v -revised

write rule check rule.g -golden -replace
write rule check rule.r -revised -replace

read design g.v -golden -replace
read design r.v -revised -replace

report rule check -verbose -both

read rule check rule.g -exclude -golden
read rule check rule.r -exclude -revised

report rule check -verbose -both
```

Related Command

WRITE RULE CHECK

READ TESTCASE

```
READ TEstcase
     <filename>
     [-DIR_name <directory_name>]
     [-RUN]
     [-REPlace]
     (LEC Mode)
```

Reads in the encapsulated testcase file which is extracted with the `REPORT TESTCASE` command. The testcase file is described in data exchange format, Extensible Markup Language (XML). After reading in the testcase file, the embedded information is separated into files, including one generated dofile. Running the generated dofile can reproduce the problem in original design.

Tcl Command

`read_testcase`

Parameters

`<filename>`	Reads in the specified file.
`-DIR_name <directory_name>`	Specifies the name of the running directory. If not specified, the name of the directory is `LEC_testcase`.
`-RUN`	Runs the generated dofile.
`-REPlace`	Overwrites the existing running directory

Example

The following command reads in the testcase file named `testcase.xml` and separate out the embedded information into files under the `run_testcase` directory. It also runs the generated dofile to reproduce the problem in original design:

```
read testcase testcase.xml -dir run_testcase -replace -run
```

Related Command

REPORT TESTCASE

READ SETUP INFORMATION

```
REAd SEtup Information
     [<filename*> ...]
     [-TYPE <type>]
     [-EXTRACT_INFO_TO_FILE <file_name>]
     [-GOLden | -REVised]
     [-REPORT]
     [-RESET]
     [-APPLY_NAME_CHANGE]
     [-VERBOSE]
     (Setup Mode)
```

This command reads in setup information from the specified files to help in LEC setup and diagnosis.

Reading in setup information from RC log files can help resolve sequential constant, sequential phase inversion, and sequential merge related setup issues. This command supports the following:

For RC log files, sequential merge information (non-inverted and inverted), sequential constant information, and sequential phase inversion are supported. Note that using attribute information is recommended for sequential phase inversion. For more information on the RC LEC Verification Flow with Attribute Information and its latest enhancements, refer to the 13.1 web interface document titled *RC LEC Verification Flow with Attribute Information*.

For RC, Conformal uses sequential constant information to handle sequential elements that are not modeled as a sequential constants due to balanced modeling, but the synthesis tool propagated the constant value. LEC automatically identifies the flop as candidate and remodels it to constant after proof. Conformal also uses constant information to handle sequential elements that are not modeled as a sequential constants because they are merged into other sequential elements. LEC automatically identifies the flop as a sequential constant candidate and remodels it to constant after proof.

For RC, where the log file records the sequential constant value, LEC can also handle sequential constant X issues where the synthesis tool optimized some sequential constant X to zero and some to one.

Tcl Command

```
read_setup_information
```

Parameters

`<filename*>`	Reads in the specified setup information file.
`-TYPE <type>`	Specifies the type of the information files, where *`type`* can be one of the following values: `RCLOG`: RC log file. *This is the default*. `VSDC`: VSDC file. `CONFORMAL`: Conformal information file.
`-EXTRACT_INFO_TO_FILE <file_name>`	Writes the extracted information to the specified file. Uses `REAd SEtup Information xxx.json -type conformal [-golden|-revised]` to read in JSON information for setup.
`-GOLden`	Specifies that all information be applied to the Golden design.
`-REVised`	Specifies that all information be applied to the Revised design. If neither `-golden` or `-revised` is specified, LEC automatically applies the information to the appropriate design: that is, sequential constant, sequential merge and phase inversion information are applied to the Golden design, while sequential duplication and multibit information are applied to the Revised design.
`-REPORT`	Reports a summary of the information already read in.
`-RESET`	Clears all the information already read in.
`-APPLY_NAME_CHANGE`	Applies name change information in HRC database. By default, the name change information will not be applied.
`-VERBOSE`	Provides additional information.

Examples

To read in an RC log file (`rc.log`):

```
SETUP> read setup information rc.log
// Processing file rc.log ...
// 749 sequential constant information are recorded
```

To clear all the information already read in:

```
SETUP> read setup information -reset
// 6175 setup information are deleted
```

To report a summary of the information already read in:

```
SETUP> read setup information -report
=================================================================
Occurrences                 Source
-----------------------------------------------------------------
4 seq_merge
4 total                     rc.log1
-----------------------------------------------------------------
855 seq_const_0
169 seq_const_1
5143 seq_merge
4 seq_merge_inv
6171 total                  rc.log2
=================================================================
```

Related Commands

REPORT SETUP INFORMATION

READ VERIFICATION INFORMATION

```
READ VErification Information
     <verification_information_directory>
     [-GOLden <label>]
     [-REVised <label>]
     (Setup/LEC Mode)
```

Reads in verification information to help optimize the current verification.

Tcl Command

`read_verification_information`

Parameters

<verification_information_directory>	Name of the directory from which to read the verification information.
-GOLden *<label>*	Specifies the label of the set of information for the golden netlist. This label can be used later to refer to this set of information.
-REVised *<label>*	Specifies the label of the set of information for the revised netlist. This label can be used later to refer to this set of information.

Related Commands

SET VERIFICATION INFORMATION

WRITE VERIFICATION INFORMATION

REDUCE MOS

```
REDuce MOs
     [-ALL | -MODule <module_name> ...]
     [-Golden | -Revised]
     [-MERGE_TRAN]
     [-SD_GND]
     [-SD_VDD]
     [-DIODE]
     [-PARALLEL]
     [-GATE_ON]
     [-GATE_OFF]
     [-NOVERbose]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Performs varieties of reduction on MOS transistor(s) from the circuit.

Tcl Command

```
reduce_mos
```

Parameters

`-All`	Reduces transistor logic from all modules. *This is the default.*
`-MODule <module_name> ...`	Specifies the module(s) to reduce transistor logic.
`-Golden`	Reduces transistor logic from the Golden design. *This is the default.*
`-Revised`	Reduces transistor logic from the Revised design.
`-MERGE_TRAN`	Collapses the tran or rtran device into the wire.
`-SD_GND`	Removes MOS devices with source and drain on GND.
`-SD_VDD`	Remove MOS devices with the source and drain on VDD.
`-DIODE`	Collapses [PN]mos into the wire if the source is on [VDD|GND] and the gate and drain are on the same net.
`-PARALLEL`	Collapses parallel-connected MOS devices into one MOS.

`-GATE_ON`	Collapses non-weak MOS devices into the wire if the gate is a constant that causes current to flow.
`-GATE_OFF`	Removes non-weak MOS devices if the gate is a constant that does not cause the current to flow.
`-NOVERbose`	Do not report detailed statistical information.

Related Commands

ABSTRACT LOGIC

READ DESIGN `-spice`

REMODEL

```
REMODEL
     <-SEQ_MERGE | -SEQ_CONSTant | -DFF_CONST_ASYNC | -UNFOLD_DFF
       | -BBOX_MERGE | -RED_DLAT | -GATED_CLOCK | -REVERSE_SEQ_REDundant
       | -SEQ2BUFfer | -SEQ_CONSTANT_GROUP | -INSTANCE_EQuivalence
       | -TRANSFORM_SET_DOMINANT | -SEQ_CONSTANT_X_TO_DC
       | -REVERSE_LATCH_REDundant>
     <-UNMAPPED | -NOTMAPPED | -MAPPED | -ALL | <gate_id> ... | <instance_pathname>
     ...>
     [-MAX_UNMAP <number_of_keypoints>]
     [-BOTH | -GOLden | -REVised]
     [-REPEAT]
     [-VERBose]
     (LEC Mode)
```

Used in LEC mode and *after* mapping, this takes a set of key points and attempts to remodel them. Use this command in conjunction with the `SET FLATTEN MODEL` command to resolve mis-compares due to key point issues.

When you use this command, the Conformal software invalidates compare results (if they exist), closes schematics, and updates the Mapping Manager.

Note: For a more automated way of setting up the comparison, use the `ANALYZE SETUP` command.

Tcl Command

```
remodel
```

Parameters

`-SEQ_MERGE`	Merges common groups of sequential elements into one sequential element in a logic cone of a key point. **Note:** This modeling can only be applied to unmapped DFFs or D-latches.
`-SEQ_CONSTant`	Converts a DFF or D-latch to a ONE or ZERO gate. **Note:** This modeling can only be applied to unmapped DFFs or D-latches, and only when the `set flatten model -seq_constant` option is set.

`-DFF_CONST_ASYNC`	Converts a DFF to a ZERO or ONE gate due to its asynchronous set or reset condition. This can be applied to both unmapped or mapped DFFs.
`-UNFOLD_DFF`	Converts a DFF to 2 D-latches with a master/slave configuration. **Note:** For latch-based custom logic comparisons, this option might work better than `set flatten model -latch_fold`.
`-BBOX_MERGE`	Performs automatic blackbox merging.
`-RED_DLAT`	Collapses serial D-latches (even when there is logic between them) into the last latch on that clock phase. You cannot use this option with latches that have set or reset pins.
`-GATED_CLOCK`	Transforms the gated clock of DFF to a MUX feedback loop.
`-REVERSE_SEQ_REDundant`	Restores the sequential redundancy to the outputs of the specified DFFs or DLATs.
`-SEQ2BUFfer`	Remodels the specified DFF/DLAT into a buffer. Using this option means that the tool will no longer check these DFF/DLATs. This can change the behavior of the netlist.
`-SEQ_CONSTANT_GROUP`	Validates and remodels a group of registers to ZERO and ONE gates, which are dependent upon each other. **Note:** You must specify a correct group of registers for this modeling to be effective; otherwise, any non-constant register in the group will prevent the true constant group from being recognized. This modeling can only be applied when the `set flatten model -seq_constant` option is set.
`-INSTANCE_EQuivalence`	Defines the equivalence of D-latches, DFFs, and CUT gates, and merges them. The D-latches, DFFs, and CUT gates must be specified in pairs with the first one as the representative. The equivalences are verified at the end of the comparison when using the `ADD COMPARED POINTS -all` command.
`-TRANSFORM_SET_DOMINANT`	Transforms the set-dominant structure of a DFF/DLATs from the output of the DFF/LAT to the input of the DFF/DLAT.

`-SEQ_CONSTANT_X_TO_DC`	Converts a DFF or D-latch to don't care when its asynchronous set or reset and clk are all disabled. **Note:** This modeling can only be applied when X assignment(s) is treated as don't care for the design.
`-REVERSE_LATCH_REDundant`	Restores the latch redundancy to the outputs of the specified DLATs.
`-UNMAPPED | -NOTMAPPED | -MAPPED | -ALL | <gate_id> | <instance_pathname>`	Applies the specified remodeling to all unmapped key points, not-mapped key points, all mapped key points, all key points, or the specified gate or instance. *By default, Conformal remodels all unmapped points.* For definition of these options, refer to REPORT UNMAPPED POINTS. Note the following for these options: You cannot use `-unmapped` with the `-instance_equivalence` option. You cannot use `-mapped` with any of the following options: `-seq_merge`, `-seq_constant`, `-unfold_dff`, `-bbox_merge`, `-red_dlat`, and `-instance_equivalence`. You cannot use `-all` with any of the following options: `-seq_merge`, `-seq_constant`, `-dff_const_async`, `-unfold_dff`, `-bbox_merge`, `-red_dlat`, and `-instance_equivalence`.
`-MAX_UNMAP <number_of_keypoints>`	Specifies an upper limit of unmapped key points to be remodeled.
`-BOTH`	Applies the specified remodeling to the Golden and Revised designs. *This is the default*.
`-GOLden`	Applies the specified remodeling to the Golden design only.
`-REVised`	Applies the specified remodeling to the Revised design only.

`-REPEAT`	Repeats until no further modeling is possible. Except for sequential constant modeling, by default, the Conformal software attempts to remodel once. This option applies only when `-seq_merge`, `-seq_constant`, or `-bbox_merge` is used. **Note:** Use `Ctrl-C` to interrupt remodeling.
`-VERBose`	Provides additional information.

Examples

- Sample Implementation for REMODEL:

```
set flatten model -seq_constant
set system mode lec
remodel -seq_constant Q1_reg
```

- In the following example, the design can be mapped almost completely by name, but there are key points that have not been merged. To remodel replicated registers into a single register, use the `REMODEL` command. Then Conformal can remap key points and compare.

```
...
set flatten model -seq_merge
set map method -name only
set system mode lec
remodel -seq_merge -both -unmapped
set map method -name first
map key points
add compare points -all
compare
```

- In the following command adds two instance equivalences (`Q1_reg` and `Q2_reg`) and (`Q4_reg` and `Q3_reg`). `Q1_reg` and `Q4_reg` are the representatives.

```
remodel -instance_equivalence -gold Q1_reg Q2_reg Q4_reg Q3_reg
```

Related Commands

COMPARE

DELETE MAPPED POINTS

MAP KEY POINTS

SET FLATTEN MODEL

REMOVE

```
REMove
     <name* ...>
     [-Golden | -Revised]
     [-INSTance | -INS_Module]
     [-MODule <mod_name*> | -ALL]
     (Setup Mode)
```

Removes instances from the database.

Tip

To report instances removed with this command, run the `REPORT REMOVED INSTANCE` command.

Tcl Command

```
remove
```

Parameters

`<instance_name* ...>`	Specifies the name(s) of the instance(s) to remove. This accepts wildcards.
`-Golden`	Removes instances from the Golden design. *This is the default.*
`-Revised`	Removes instances from the Revised design.
`-INSTance`	Indicates that the `<name* ...>` specifies the instance names. This specifies to remove all instances whose instance names match `<name* ...>`. *This is the default.*
`-INS_Module`	Indicates that the `<name* ...>` specifies the module names. This specifies to remove all instances whose module names match `<name* ...>`.
`-MODule <module_name*>`	

	Specifies the module names. This accepts wildcards. This specifies to remove the specified instances from only the specified modules in `<module_name*>`. **Note:** If you do not specify `<module_name*>`, the `REMOVE` command removes the specified instances from only the root module. *This is the default.*
`-ALL`	Removes the specified instances from all modules in the design.

Example

For these lines:

```
module top (...);
  mod1 u01 (...); // inst1
  mod2 u02 (...); // inst2
  mod3 u03 (...); // inst3
endmodule
module mod3 (...);
  mod1 u01 (...); // inst4
  mod2 u02 (...); // inst5
endmodule
```

- The following command removes `u01` from the root module `top`:

  ```
  remove u01 -ALL              // remove inst1, inst4
  ```

- The following command removes u01 from module `mod3`:

  ```
  remove u01 -MODULE mod3      // remove inst4
  ```

- The following command removes all `mod1` instances from the root module `top`:

  ```
  remove mod1 -INS_Module      // remove inst1
  ```

- The following command removes all `mod1` instances from all modules:

  ```
  remove mod1 -INS_Module -ALL // remove inst1, inst4
  ```

Related Commands

REPORT REMOVED INSTANCE

REPORT ABSTRACT MODEL

```
REPort ABSTract Model
     [-ALL | -MODule <module_name ...>]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Note: This requires a Conformal GXL license.

If you used `SET ABSTRACT MODEL` command to abstract transistor logic from particular modules, this reports their abstraction conditions.

Tcl Command

`report_abstract_model`

Parameters

`-All`	Reports abstraction conditions for all modules.
`-MODule <module_name ...>`	Reports abstraction conditions for the specified modules.
`-Both`	Reports abstraction conditions for both the Golden and Revised designs. *This is the default.*
`-Golden`	Reports abstraction conditions for the Golden design.
`-Revised`	Reports abstraction conditions for the Revised design.

Related Commands

ABSTRACT LOGIC

RESET ABSTRACT MODEL

SET ABSTRACT MODEL

REPORT ALIAS

```
REPort ALias
     [<aliasname* ...>]
     (Setup / LEC Mode)
```

Displays a list of all or specified aliases you created with the `ADD ALIAS` command.

Wildcard: The wildcard (*) represents any zero or more characters in alias names.

Tcl Command

```
report_alias
```

Parameters

`<aliasname* ...>`	Reports the specified alias.

Related Commands

ADD ALIAS

DELETE ALIAS

REPORT BLACK BOX

```
REPort BLack Box
     [-Module | -Instance]
     [-DETail]
     [-Class <Full | User | System | UNDefined | UNSupported
        | EMPty | NOTranslate>]
     [-HIER | -NOHIER]
     [-HIDden | -NOHIDden]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays blackboxes from the Golden and Revised designs. The blackboxes either already existed in the design, or you previously added them with the `ADD BLACK BOX` command.

Tcl Command

```
report_black_box
```

Parameters

`-Module`	Reports only the blackbox modules. *This is the default.*
`-Instance`	Reports only the blackbox instances.
`-DEtail`	Displays details about blackboxes, where:

`USER`	Indicates that the blackbox was added by the `ADD BLACK BOX` command.
`SYSTEM (empty)`	Indicates that the module contains no logic.
`SYSTEM (GRAYBOX)`	Indicates that a Liberty cell has an incomplete functional definition. The cell has undriven outputs and a tied ouput, but no well-functioning output pin.
`SYSTEM (LEF)`	Indicates that the macro is defined in the LEF file, but not defined in library/design modules.
`SYSTEM (notranslate)`	Indicates that the blackbox was added by the `ADD NOTRANSLATE MODULES` command.

	`SYSTEM (undefined)`	Indicates that the blackbox was added by the `SET UNDEFINED CELL blackbox` command.
	`SYSTEM (unsupported)`	Indicates that the module contains unsupported statements.
`-Class`	Displays the specified class of blackboxes.	
	`Full`	Blackboxes from both the *User* and *System* classes. *This is the default.*
	`User`	Blackboxes previously added with the `ADD BLACK BOX` command.
	`System`	Blackboxes included in the original design.
	`UNDefined`	Blackboxes for undefined modules.
	`UNSupported`	Blackboxes for unsupported modules.
	`EMPty`	Blackboxes for empty modules.
	`NOTranslate`	Blackboxes for notranslate modules.
`-HIER`	Displays the hierarchical blackboxes. *This is the default.*	
`-NOHIER`	Do not display the hierarchical blackboxes.	
`-HIDden`	Displays all blackbox instances in the design hierarchy including those contained within other blackboxes. *This is the default.*	
`-NOHIDden`	Do not display blackbox instances that are contained within other blackboxes.	
`-Both`	Displays blackboxes from both the Golden and Revised designs. *This is the default.*	
`-Golden`	Displays blackboxes from the Golden design.	
`-Revised`	Displays blackboxes from the Revised design.	

Related Commands

ADD BLACK BOX

DELETE BLACK BOX

REPORT CLOCK

```
REPort CLock
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Reports all clocks from the Golden and Revised designs that were added with the `ADD CLOCK` command.

Tcl Command

```
report_clock
```

Parameters

`-Both`	Displays all added clocks from both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays all added clocks from the Golden design.
`-Revised`	Displays all added clocks from the Revised design.

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

READ PATTERN

REPORT COMMAND PROFILE

```
REPort COmmand PRofile
     [-Summary | -Detail]
     (Setup / LEC Mode)
```

Displays a profile of all of the commands you executed *after* you used `SET COMMAND PROFILE` with the `ON` option. (The default setting for `SET COMMAND PROFILE` is `OFF`.)

The profile report includes the order in which commands were executed and the memory use. The profile includes commands that were executed in the GUI mode.

Tcl Command

```
report_command_profile
```

Parameters

`-SUMmary`	Lists a summary table of all of the commands in alphabetical order. *This is the default.*
`-Detail`	Lists all commands in order of execution.

Related Commands

SET COMMAND PROFILE

SET LOG FILE

REPORT COMPARE DATA

```
REPort COmpare Data
     [-CLASS <EQuivalent | INVequivalent | NONEQuivalent
                        | ABort | NOTcompared | SYStem | USEr | FULL
                        | INSTance_eq | OUTput_eq | PIN_eq> ...]
     [-Type <PO | DFF | DLAT | BBOX | CUT | BBOX_INPUT> ...]
     [-Verbose | -SUMmary]
     [<identifier> [-INSTance_eq <identifier> |
                   -OUTput_eq <identifier> |
                   -PIN_eq <identifier>]]
     [-HIER_TO_Flat]
     [-LOng]
     [-Golden |-REvised]
     [-NOREPORT_BBox_input | -REPORT_BBox_input]
     [-UNReachable]
     [-CONCise]
     [-LIBrary_name | -NOLIBrary_name]
     (LEC Mode)
```

Displays a list of all or specified compared points. If no options are specified, Conformal identifies all equivalent and nonequivalent compared points and displays a summary.

The compared points are listed in pairs of rows with three fields in each row. The first row in each pair represents the Golden design. The second row in each pair represents the Revised design. The three fields in each row are:

- First–the gate identification number
- Second–the gate type
- Third–the instance path or pin path

Tcl Command

```
report_compare_data
```

Parameters

`-CLASS`	Displays the specified class of compared points: `EQuivalent`: Equivalent points `INVequivalent`: Inverted-equivalent points `NONEQuivalent`: Nonequivalent points `ABort`: Aborted points `NOTcompared`: All points not compared `SYStem`: Automatically mapped points `USEr`: User-mapped points `FULL`: Both automatically-mapped and user-mapped points `INSTance_eq`: Equivalent instances `OUTput_eq`: Equivalent outputs `PIN_eq`: Equivalent pins
`-TYPE`	Displays the specified type of compared points: `PO`: Primary output `DFF`: D flip-flop `DLAT`: D-latch `BBOX`: Blackbox `CUT`: Compared points with artificial gates to break combinational loops `BBOX_INPUT`: Blackbox inputs that are individually reported
`-Verbose`	Displays all compared points. *This is the default.*
`-SUMmary`	Lists a summary report of the compared points.
`identifier`	Lists compared points for the specified gate ID or path. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.

`-INSTance_eq <identifier>`

Displays compare data for a pair of instances previously identified with the `ADD INSTANCE EQUIVALENCE` command.

Note: ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.

`-OUTput_eq <identifier>`

Displays compare data for a pair of outputs previously identified with the `ADD OUTPUT EQUIVALENCES` command.

Note: ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.

`-PIN_eq <identifier>`

Display compare data for a pair of pins previously identified with the `ADD PIN EQUIVALENCE` command.

Note: ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.

`-HIER_TO_Flat`	Analyzes the verification information and the hierarchical comparison information to produce the report as if it was a flattened run. Verification information must have been enabled using `SET VERIFICATION INFORMATION`
`-LOng`	Displays pairs of points on separate lines.
`-Golden`	The specified identifiers are from the Golden design. *This is the default.*
`-REvised`	The specified identifiers are from the Revised design.

`-NOREPORT_BBox_input`

Do not report the blackbox input pins in the compare report results. *This is the default.*

`-REPORT_BBox_input`	Report the blackbox input pins in the compare report results (Equivalent, Nonequivalent, Abort, and Not-compared).
`-UNReachable`	Reports only unreachable compare points.

`-CONCise`	Reports the compared data in a concise format (for instance, non-equivalent pin information of the non-equivalent key point will not be displayed in this report).
`-LIBRARY_name`	Displays the library name in the key point name. *This is the default.*
`-NOLIBrary_name`	Do not show the library name in the key point name.

Examples

For a set of sample commands that shows this and related commands in context, see the example for the COMPARE command.

Related Commands

COMPARE

DIAGNOSE

PROVE

REPORT STATISTICS

REPORT COMPARE TIME

REPORT COMPARE TIME

```
REPort COmpare TIME
     [-ENABLE | -DISABLE]
     [-Golden | -Revised <identifier>]
     [-MAX <integer>]
     [-SORT]
     [-RTLINFO]
     [-ABORT_only]
     (LEC Mode)
```

Reports the CPU time consumed during a comparison.

You must enable this feature before starting a comparison; otherwise, Conformal does not record any information.

For example:

```
compare
report compare time -enable
compare
report compare time
```

In this example, Conformal records the CPU time for the second comparison only.

Note: Conformal does not record compare time for trivial cones.

Tcl Command

```
report_compare_time
```

Parameters

`-ENABLE`	Use this option to start recording CPU time. If you do not set this option before your comparison, Conformal does not record CPU time.
`-DISABLE`	Disables the recording of CPU time.
`-Golden <identifier>`	Specifies particular key points in the Revised to report. If you do not specify any key points, Conformal reports all key points.
`-Revised <identifier>`	Specifies particular key points in the Revised to report.

`-MAX <integer>`	Specifies the maximum number of key points to report.
`-SORT`	Sorts the information based on the recorded CPU time.
`-RTLINFO`	Reports RTL information inside the cone.
`-ABORT_ONLY`	Report only the abort points.

Examples

The following demonstrates how to use this command with other commands.

Note: Use these steps after you read in your library and design files.

1. Add your compare points.

```
LEC > add compared points -all
// 3113 compared points added to compare list
```

2. Enable the report compare time feature.

```
LEC> report compare time -enable
```

3. Start your comparison.

```
LEC> compare
===========================================================================
Compared points      PO     DFF    DLAT   BBOX      Total
---------------------------------------------------------------------------
Equivalent           123    2983   2      5         3113
===========================================================================
// Warning: 1 DFFs/DLATs have 1 disabled clock port: skipped data
   cone comparison
```

4. Report the compare time. In this example, Conformal sorts the information based on CPU time and only reports RTL information within the cone.

```
LEC>report compare time -sort -rtlinfo

CPU Time Used: 5.29, Result: Equivalent:
  (G) + 536   DFF  /cpu_core/CPU/cpu_dp/alu/v_alo_1l_reg[30]
  (R) + 1284  DFF  /cpu_core$CPU$cpu_dp$alu$v_alo_1l_reg_30_/U$1/U$1
  RTL modules at Golden:
  RTL modules at Revised:

  CPU Time Used: 4.64, Result: Equivalent:
  (G) + 533   DFF  /cpu_core/CPU/cpu_dp/alu/o_alvo_1l_reg
  (R) + 1364  DFF  /cpu_core$CPU$cpu_dp$alu$o_alvo_1l_reg/U$1/U$1
  RTL modules at Golden:
  RTL modules at Revised:

      .
      .
      .
```

Related Commands

ADD COMPARED POINTS

COMPARE

REPORT COMPARE DATA

REPORT COMPARED INTENT

```
REPort COMPared Intent
     [-CLASS <EQuivalent | NONEQuivalent | NOTcompared | NO_EFFECT | ALL> ...]
     [-COMMAND_name <command name> ...]
     [-FILTERED]
     [-FILTERED_BY <name>* ...]
     [-OBJECT_TYPE <type> ...]
     [-ATTRibutes]
     [-COMMAND_SUMMARY | -COMPARED_POINT_SUMMARY]
     [-FILTER <filter_expression>]
     [-DESIGN_OBJECT_FILTER <filter_expression>]
     [-REGEXp]
     [-FAIL_TYPE <type> ...]
     [-Summary | -Verbose]
     [-Golden |-REvised]
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Displays the results of the `COMPARE POWER INTENT` command.

Tcl Command

`report_compared_intent`

Parameters

`-CLASS`	Displays the specified class of compared points.
	`EQuivalent`: Equivalent points
	`NONEQuivalent`: Nonequivalent points
	`NOTcompared`: All points not compared
	`NO_EFFECT`: All nonequivalent points that have no effect on power intent
	`ALL`: All compared points
`-COMMAND_name`	Displays only compared points for commands that match the specified names. Wildcards are supported.
`-FILTERED`	Filter the results using the filters added by the `ADD POWER_INTENT_COMPARE FILTER` command.

`-FILTERED_BY <name>*`	Filter the results using the specified power intent compare filter(s): filters are added using the `ADD POWER_INTENT_COMPARE FILTER` command. This accepts wildcards.
`-OBJECT_TYPE <type>`	Reports only compared points for the specified power intent object type. For a list of supported options, see the section Power Intent Object Types in the REPORT POWER INTENT command page.
`-ATTRibutes`	Reports attributes of compared points.
`-COMMAND_SUMMARY`	Reports summary based on number of commands. *This is the default.*
`-COMPARED_POINT_SUMMARY`	Reports summary based on number of compared points.
`-FILTER <filter_expression>`	Reports only compared points for which the specified `filter_expression` evaluates to true. The syntax for `filter_expression` is the same as for the Tcl find command. Use the `-ATTRibutes` option to see the available attributes.
`-DESIGN_OBJECT_FILTER <filter_expression>`	Reports only design object compared points for which the specified evaluates to true. The syntax for `filter_expression` is the same as for the Tcl find command. Use the `-ATTRibutes` option to see the available attributes.
`-REGEXp`	Specifies that the pattern matching operators used inside filter expression (such as =~) should use regular expressions as patterns, rather than glob-style patterns. This option only has any effect if combined with `-FILTER` and/or `-DESIGN_OBJECT_FILTER`.
`-FAIL_TYPE <type>`	Reports only compared points which match the specified fail type. Use the -`ATTRibutes` option to see the fail types of compared points.
`-Summary`	A summary table of the power comparison results. *This is the default*.
`-Verbose`	Displays the results in detail.
`-Golden`	Only reports points for which the golden object exists.

`-REvised`	Only reports points for which the revised object exists.

Example

The following is a sample dofile that reads in two power intent files (in CPF format) and compares them:

```
set lowpower option -golden_netlist_style  logical
set lowpower option -revised_netlist_style logical
add notranslate module RamMod -both -library
read library      Lib/Waz.l -gol -lib
read power intent Des/RTL.c -gol -cpf
read library      Lib/Waz.l -rev -lib
read design Des/RTL.v -gol -v2k -root MyChip
read design       Des/Syn.v -rev -v2k -root MyChip
read power intent Des/Syn.c -rev -cpf
compare power intent
report compared intent -verbose
```

After you read in the designs, libraries, and power intent, use the `COMPARE POWER INTENT` command to compare the two power intent files, and the `REPORT COMPARED INTENT` to view the details of the comparison. In the following sample output, the two power intent files are not equivalent due to a missing power mode and mismatched values for a nominal condition:

```
SETUP> compare power intent
// Power Intent Compare finished and found 2 differences:
// power intent object name differences:              1
// power intent object option differences:            1

SETUP> report compared intent -verbose
// Power Intent Compare: Non-Equivalent
//
================================================================================
| Power intent differences
================================================================================
| 1) Supply Port 't1/l1/x1/VDD' missing in Revised:
|     Golden  file: 'Power_Intent/top.gol.upf', line 25
|     Revised file: not specified
|
| 2) Supply Port 't1/l1/x1/VSS' missing in Revised:
|     Golden  file: 'Power_Intent/top.gol.upf', line 26
|     Revised file: not specified
|
| 3) Supply Port 't1/l1/x2/VDD' missing in Revised:
|     Golden  file: 'Power_Intent/top.gol.upf', line 27
|     Revised file: not specified
|
| 4) Supply Port 't1/l1/x2/VSS' missing in Revised:
|     Golden  file: 'Power_Intent/top.gol.upf', line 28
|     Revised file: not specified
```

```
|
| 5) Supply Port 'tt1/ll1/xx1/VDD' missing in Golden:
|     Golden  file: not specified
|     Revised file: 'Power_Intent/top.rev1.upf', line 25
|
| 6) Supply Port 'tt1/ll1/xx1/VSS' missing in Golden:
|     Golden  file: not specified
|     Revised file: 'Power_Intent/top.rev1.upf', line 26
|
| 7) Supply Port 'tt1/ll1/xx2/VDD' missing in Golden:
|     Golden  file: not specified
|     Revised file: 'Power_Intent/top.rev1.upf', line 27
|
| 8) Supply Port 'tt1/ll1/xx2/VSS' missing in Golden:
|     Golden  file: not specified
|     Revised file: 'Power_Intent/top.rev1.upf', line 28
================================================================================

================================================================================
=                         Power Intent Compare Summary                         =
================================================================================
| Power intent command                                EQ      Non-EQ  |  Total
| -----------------------------------------------  --------  -------- | --------
| add_power_state                                        6         0 |        6
| associate_supply_set                                   6         0 |        6
| create_power_domain                                    2         0 |        2
| create_supply_net                                      3         0 |        3
| create_supply_port                                     3         8 |       11
| create_supply_set                                      2         0 |        2
| define_isolation_cell                                 17         0 |       17
| define_retention_cell                                 12         0 |       12
| name_format                                            1         0 |        1
| set_scope                                              1         0 |        1
| supply_net_handle (created by other commands)          4         0 |        4
--------------------------------------------------------------------------------
| Power intent command with no one-to-one mapping     EQ      Non-EQ  |  Total
| -----------------------------------------------  --------  -------- | --------
| set_port_attributes                                    4         0 |        4
--------------------------------------------------------------------------------
| Total                                                 61         8 |       69
================================================================================
```

Related Commands

ADD POWER_INTENT_COMPARE FILTER

COMPARE POWER INTENT

READ DESIGN

READ LIBRARY

READ POWER INTENT

REPORT COMPARED LIBERTY

```
REPort COMPared Liberty
     [-CLASS <-NONEQ | -EQ | -UNDEFINED | -ALL | -WAIVED>]
     [-TABULAR <max_pin_columns>]
     [-NO_COLLAPSE_LIB_PINS]
     [-SUPPORTED_ATTRIBUTES]
     [-Summary | -Verbose]
     (Setup/LEC Mode)
```

Description

Note: This is a Conformal Low Power command.

Displays the results of the `COMPARE LIBERTY` command.

Tcl Command

`report_compared_liberty`

Parameters

`-CLASS`	Specifies the class of results to display. `-NONEQ`: You can use either `-FAIL` or `-NON_EQ` to display the nonequivalent attributes. *This is the default*. `-EQ`: You can use either `-PASS` or `-EQ` to display the equivalent attributes. `-UNDEFINED`: Displays UNDEFINED attributes. `-ALL`: Displays all attributes. `-WAIVED`: Displays waived attributes.
`-TABULAR <max_pin_columns>`	Reports the results in a tabular format. Where `<max_pin_columns>` specifies the maximum number of columns to display. By default, there is no limit. Use this option to get a more readable report in case a library cell has many pins.

`-NO_COLLAPSE_LIB_PINS`	By default, bus pins are collapsed in the report. Use this option to disable the collapsing of bus pins.
`-SUPPORTED_ATTRIBUTES`	View supported attributes in the Liberty comparison. This option can be executed before reading in the Liberty files.
`-SUMMARY`	Display a summary table of the Liberty comparison results. *This is the default*.
`-VERBOSE`	Display a detailed report of the Liberty comparison results.

Example

The following is a sample dofile that reads in two Liberty libraries and compares them:

```
read liberty -liberty gold_sample.lib -golden -lp
read liberty -liberty revs_sample.lib -revised -lp
compare liberty
report compared liberty -verbose
```

After you read in the libraries, use the `COMPARE LIBERTY` command to compare the two libraries, and the `REPORT COMPARED LIBERTY` to view the details of the comparison.

Related Commands

ADD LIBERTY_COMPARE FILTER

COMPARE LIBERTY

DELETE LIBERTY_COMPARE FILTER

REPORT LIBERTY_COMPARE FILTER

REPORT COMPARED POINTS

```
REPort COmpared Points
     [-SUMmary | -PO | -DFf | -DLat | -Bbox | -Cut]
     [-UNReachable]
     (LEC Mode)
```

Displays the compared points that were added with the `ADD COMPARED POINTS` command.

The first row represents the Golden design; the second row represents the Revised design. It also shows a tabulated summary of the compared points for each design. This report includes the total number of compared points for primary outputs, DFFs, D-latches, blackboxes, and cut gates.

If you do not specify any options, Conformal lists all added compared points, and a tabulated summary appears at the end of the list. However, if you use the `-summary` option, Conformal displays only the tabulated summary.

Tcl Command

```
report_compared_points
```

Parameters

`-SUMmary`	Lists a summary table of all of the added compared points in the Golden and Revised designs. *This is the default.*
`-PO`	Lists all primary output compared points.
`-DFf`	Lists all D flip-flop compared points.
`-DLat`	Lists all D-latch compared points.
`-Bbox`	Lists all blackbox compared points.
`-Cut`	Lists all compared points for artificial gates that break combinational loops.
`-UNReachable`	Lists only the unreachable compare points.

Related Commands

ADD COMPARED POINTS

COMPARE

DELETE COMPARED POINTS

REPORT STATISTICS

REPORT COMPARED POWER_GRID

```
REPort COMPared POWER_GRID
     [-CLASS EQuivalent | NONEQuivalent | INconclusive | MApped | UNmapped]
     [-POINT <supply_point_name> [-LEAF_cell_type] [-DRIVER] [-LOAD]]
     [-INStance <instance_name>]
     [-BBOX_macro]
     [-GOLden | -REVised]
     [-Verbose]
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command and an 1801 feature.

Reports the result of power grid comparison performed by the COMPARE POWER GRID command.

Tcl Command

`report_compared_power_grid`

Parameters

`-BBOX_macro`	Report power grid comparison results on black-boxes and macros.
`-CLASS`	Displays the specified class of compared supply.
	`EQuivalent`: Supply sources which are equivalent.
	`NONEQuivalent`: Supply sources which are non-equivalent.
	`INConclusive`: Supplies sources with unmapped loads.
	`MApped`: Supply sources which are mapped.
	`UNmapped`: Supply points which cannot be mapped.
`-Verbose`	Displays all classes of compared supply when `-CLASS` is not specified.
`-POINT <supply_point_name>`	
	Specifies the supply point name to display the supply point information. Use `-GOLden` or `-REVised` to specify the supply point name is a golden or revised supply point. If `-GOLden` and `-REVised` are not specified, default is `-GOLden`.

`-LEAF_cell_type`	Reports the types of leaf cells which are inside the sub-hierarchy of the specified supply point and connected through the specified supply point.
`-DRIVER`	Reports the driving path from the supply source(s) to the specified supply point.
`-LOAD`	Reports the direct loads of the specified supply point.
`-INStance <instance_name>`	Specifies the instance name to display the supply points of the instance. Use `-GOLden` or `-REVised` to specify the instance name is a golden or revised instance. If `-GOLden` and `-REVised` are not specified, default is `-GOLden`.
`-GOLden`	Specifies that the supply point name or instance name is in the golden design. *This is the default*.
`-REVised`	Specifies that the supply point name or instance name is in the revised design.

Examples

- Use `-CLASS UNMAPPED` to view unmapped supplies detail information:

```
// Command: report_compared_power_grid -class unmapped -ver
Compare Power Grid: Non-Equivalent
Unmapped golden supplies:
1: (G) xMux/xSW/xSL/VDD (Virtual Supply Pin) (xMux/xSW/xSL:xMux/
MuxB.primary.power) (instance: xMux/xSW/xSL, unmapped)
2: (G) xMux/xSW/xSL/VSS (Virtual Supply Pin) (xMux/xSW/xSL:xMux/
MuxB.primary.ground) (instance: xMux/xSW/xSL, unmapped)
Unmapped revised supplies:
1: (R) xMux/xSWx.xSL/VDD (Physical Supply Pin) (xMux/xSWx.xSL:xMux/
Mux.primary.power) (instance: xMux/xSWx.xSL, unmapped)
2: (R) xMux/xSWx.xSL/VSS (Physical Supply Pin) (xMux/xSWx.xSL:xMux/
Mux.primary.ground) (instance: xMux/xSWx.xSL, unmapped)
3: (R) xMux/xSWx.xSLw/VDD (Physical Supply Pin) (xMux/xSWx.xSLw:xMux/
Mux.primary.power) (instance: xMux/xSWx.xSLw, unmapped)
4: (R) xMux/xSWx.xSLw/VSS (Physical Supply Pin) (xMux/xSWx.xSLw:xMux/
Mux.primary.ground) (instance: xMux/xSWx.xSLw, unmapped)
```

- To report the driving path from the supply sources to a specific supply, use `-POINT`, `-DRIVER`. Below examples shows how to view the diving supply source to a golden supply point, `xMux/xSW/xSL/VDD`

```
TCL_SETUP> report_compared_power_grid -point xMux/xSW/xSL/VDD -driver -golden
(G) xMux/xSW/xSL/VDD (Virtual Supply Pin) (xMux/xSW/xSL:xMux/MuxB.primary.power)
(instance: xMux/xSW/xSL, unmapped) (unmapped)
```

```
Driver path from source VDD:
|-VDD (Virtual Supply Pin) (/:Def.primary.power) (instance: /, mapped instance: /
) (mapped: VDD) (equivalent)
  |-xMux/VDD (Virtual Supply Pin) (instance: xMux, mapped instance: xMux) (mapped:
xMux/VDD) (equivalent)
    |-xMux/xSW/VDD (Virtual Supply Pin) (instance: xMux/xSW, unmapped) (unmapped)
      |-xMux/xSW/xSL/VDD (Virtual Supply Pin) (xMux/xSW/xSL:xMux/
MuxB.primary.power) (instance: xMux/xSW/xSL, unmapped) (unmapped)
```

- Report the direct loading supply points of the revised supply point `VDD`:

```
TCL_SETUP> report_compared_power_grid -point VDD -load -revised
(R) VDD (Physical Supply Pin) (/:Def.primary.power) (instance: /, mapped instance:
/) (mapped: VDD) (equivalent)
Loads:
|-VDD (Physical Supply Pin) (/:Def.primary.power) (instance: /, mapped instance: /
) (mapped: VDD) (equivalent)
  |-xReg/VDD (Physical Supply Pin) (xReg:Def.primary.power) (instance: xReg, mapped
instance: xReg) (mapped: xReg/VDD) (equivalent)
  |-xMux/VDD (Physical Supply Pin) (instance: xMux, mapped instance: xMux) (mapped:
xMux/VDD) (equivalent)
  |-xRamWrap/VM (Physical Supply Pin) (xRamWrap:Def.primary.power) (instance:
xRamWrap, mapped instance: xRamWrap) (mapped: xRamWrap/VM) (non-equivalent)
```

Related Commands

COMPARE POWER GRID

ADD MAPPED INSTANCE

READ KEYPOINT MAPPING

SET EQUIVALENT SUPPLY_SOURCES

ADD IGNORED GRID

DELETE IGNORED GRID

REPORT IGNORED GRID

REPORT CPF LOGIC

```
REPort CPf Logic
     [-ISOlation]
     [-Verbose]
     (Setup / LEC Mode)
```

Note: This is a Conformal Low Power command.

Reports the low power cells that were inserted by the Conformal Low Power software.

Tcl Command

`report_cpf_logic`

Parameters

`-ISOlation`	Reports the inserted isolation cells only. *This is the default.*
`-VERbose`	Reports detailed information of each defined CPF cell, including cell types and rules that triggered this cell to be inserted in the design.

Note: By default, this command reports all inserted low power cell types.

Example

The following commands read the `lib.cpf` and `design.cpf` files, performs low power cell insertion, and reports only the inserted isolation and level-shifter cells:

```
read power intent -cpf lib.cpf design.cpf
commit power intent -insert_isolation
report cpf logic -isolation -level_shifter
```

Related Commands

COMMIT POWER INTENT

READ POWER INTENT

REPORT CROSSING PATH

```
REPort CRossing Path
     [-SRC_supply <supplyset>]
     [-DEST_supply <supplyset>]
     [-DRIVER <pin*>]*
     [-GUI]
     [-ID [<id>]]
     [-LOAD <pin*>]*
     [-THROUGH <pin*>]*
     [-LIMit <n>]
     [-Verbose]
     [-GOLden | -REVised]
     (Setup Mode)
```

Note: This is a Conformal Low Power command and an 1801 feature.

Reports paths that are analyzed by the tool, including all potential domain crossings. In GUI mode, it can be used to visualize the reported paths in a schematic window.
The reported paths may contain objects that are not in the design, but are inferred by application of the insertion strategies in the power intent.

Not every path will be reported. For example, paths that lie completely within a power domain are not available in this report. The paths reported traverse across low-power cells, hierarchical boundaries, macro feedthroughs and certain buffers/inverters. They start and end at primary ports and/or logic leaf cells such as combinational cells or sequential cells, macros, and other blackboxes.

Tcl Command

```
report_crossing_path
```

Parameters

`-SRC_supply <supplyset>`	Report only paths whose start point has the specified related driver supply set.
`-DEST_supply <supplyset>`	Report only paths whose end point has the specified related receiver supply set.
`-DRIVER <pin*>`	Report only paths starting at the specified drivers.
`-GUI`	In GUI mode, this option opens a schematic window displaying the reported paths.

`-ID [<id>]]`	When specified without an argument, the command annotates every reported crossing path with its identifier (a string in the form "`crossing_<n>`"). When specified with a crossing path identifier, the command only reports the crossing path that has the specified identifier. If a number `<n>` is specified, the identifier "`crossing_<n>`" is assumed.
`-LOAD <pin*>`	Report only paths ending at the specified loads.
`-THROUGH <pin*>`	Report only paths traversing the specified pins. Note: When specified more than once, the paths reported can traverse any number of the specified pins, in any order.
`-LIMit <n>`	Specifies the maximum number of paths to report. Default is 0 (no limit).
`-Verbose`	Without this option, only the driver/load pairs are reported. With this option, all the path segments are reported.
`-GOLden`	Report only paths in the Golden design.
`-REVised`	Report only paths in the Revised design.

Example

To report all paths across u1/in[*] starting in a cell output with driver supply SS1:

```
report crossing path -src_supply SS1 -through u1/in[*]
```

To visualize them in the schematics and report all their intermediate points in the transcript window:

```
report crossing path -src_supply SS1 -through u1/in[*] -verbose -gui
```

Related Commands

REPORT LOWPOWER INFO

REPORT POWER INTENT

REPORT CUT POINT

```
REPort CUt Point
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays all cut points from the Golden and Revised designs that were added with the `ADD CUT POINT` command.

Tcl Command

```
report_cut_point
```

Parameters

`-Both`	Lists all cut points in both the Golden and Revised designs. *This is the default.*
`-Golden`	Lists all cut points in the Golden design.
`-Revised`	Lists all cut points in the Revised design.

Related Commands

ADD CUT POINT

DELETE CUT POINT

REPORT PATH

REPORT DATAPATH OPTION

```
REPort DAtapath Option
```
(*Setup / LEC Mode*)

Displays current datapath option settings.

Related Commands

ANALYZE DATAPATH

ANALYZE MODULE

REPORT MULTIPLIER OPTION

SET DATAPATH OPTION

SET MULTIPLIER OPTION

SET FLATTEN MODEL

REPORT DATAPATH RESOURCE

```
REPort DATapath REsource
     [-Verbose]
     [-Analyzed]
     [-Type <MULT | ADD | SUB | MERGED>]
     (LEC Mode)
```

Displays information about datapath resources from the Golden and Revised designs.

Tcl Command

```
report_datapath_resource
```

Parameters

`-Verbose`	Provides additional information, such as filename and line number.
`-Analyzed`	Provides information only for resources analyzed by `ANALYZE DATAPATH` command.
`-Type`	Provides information only for resources of the specified type. Choose one of the following: `MULT` : multipliers `ADD` : adders `SUB` : subtractors `MERGED` : merged operators

Related Commands

ANALYZE DATAPATH

ANALYZE MODULE

REPORT DESIGN DATA

REPORT DESIGN DATA

```
REPort DEsign Data
     [<module_name>]
     [-Summary | -Verbose]
     [-NOKey_point | -Key_point]
     [-Extra <INPut | Output | INOut>]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays design data on the Golden and Revised designs. It displays the number of design modules, library cells, inputs, outputs, primitives, and one-to-one mapped state points.

This report includes word-level information about the design in terms of the number of arithmetic/keyword operations. This report includes datapath elements such as WMUX, WAND, WXOR and other word-level representations of Boolean logic. It displays simpler representations of datapath logic that may need to be separated out for the comparison process.

Press `Ctrl-C` to interrupt the key point listing if you find that the report is too long.

Tcl Command

```
report_design_data
```

Parameters

`<module_name>`	Reports design data for the named module. By default, the Conformal software reports design data on the top root design module.
`-Summary`	Summarizes the design data for the total number of design modules, library cells, inputs, outputs, and primitives. *This is the default.*
`-Verbose`	Reports a detailed list of the design's total number of design modules, inputs, outputs, each different library cell, and each different primitive.
`-NOKey_point`	Do not report the total one-to-one mapped state points. *This is the default.*

`-Key_point`	Reports the total one-to-one mapped state points. **Note:** If you use the -verbose option in conjunction with this option, Conformal reports all one-to-one mapped state points. Otherwise, Conformal reports the total in summary.
`-Extra`	Reports the extra input, output, or I/O pins for pair-able modules between the Golden and Revised designs. `INPut` Specifies input pins. `Output` Specifies output pins. `INOut` Specifies inout pins.
`-Both`	Report design data on both the Golden and Revised designs. *This is the default.*
`-Golden`	Report design data on the Golden design.
`-Revised`	Report design data on the Revised design.

Related Commands

READ DESIGN

READ LIBRARY

REPORT DATAPATH RESOURCE

REPORT DESIGN SIMILARITY

```
REPort DEsign SIMilarity
     [-INStance <instance_name*>]
     [-GOLDen | -REVised]
     [-Verbose]
     (LEC Mode)
```

Displays the similarity degree of a design with reference to the other netlist. The similarity is measured by the number of corresponding points in the two designs. The value of similarity ranges from 0% to 100%. If the two designs are identical in structure, the similarity degree is 100%. The reporting can be used to evaluate the complexity of comparing two designs.

Tcl Command

```
report_design_similarity
```

Parameters

`-INStance <instance_name*>`	
	Displays the similarity degree of the netlist inside the specified instance. The similarity is evaluated with reference to the other netlist. If no instance is specified, the similarity is for the entire design.
`-Golden`	Specifies that the similarity evaluation is performed on the Golden design. The Revised netlist is used for reference. *This is the default.*
`-Revised`	Specifies that the similarity evaluation is performed on the Revised design. The Golden netlist is used for reference.
`-Verbose`	Displays detailed information of the comparison evaluation.

Examples

- The following command displays the similarity of the Golden design's netlist. The Revised design's netlist is used for reference.

  ```
  report design similarity
  ```

- The following command displays the similarities of the instances whose name begins with `mult` in the Golden design's netlist:

  ```
  report design similarity -instance mult*
  ```

Related Topic

Reporting Design Similarities

REPORT DONTTOUCH REGISTERS

```
REPort DOnttouch Registers
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Display the don’t-touch registers in Golden and/or Revised designs which were specified with the `add_donttouch_registers` command.

Tcl Command

```
report_donttouch_registers
```

Parameters

`-Golden`	Displays don’t-touch registers to the Golden design.
`-Revised`	Displays don’t-touch registers to the Revised design.
`-Both`	Displays don’t-touch registers to both the Golden and Revised designs

Related Commands

ADD DONTTOUCH REGISTERS

DELETE DONTTOUCH REGISTERS

REPORT DYNAMIC CONSTRAINTS

```
REPort DYnamic Constraints
     [-Both | -Golden | -Revised]
     (LEC Mode)
```

Displays all of the dynamic constraints you added to the Golden and Revised designs with the `ADD DYNAMIC CONSTRAINTS` command.

Tcl Command

```
report_dynamic_constraints
```

Parameters

`-Both`	Lists all dynamic constraints in both the Golden and Revised designs. *This is the default.*
`-Golden`	Lists all dynamic constraints in the Golden design.
`-Revised`	Lists all dynamic constraints in the Revised design.

Examples

For a set of sample commands that shows this and related commands in context, see the example for the COMPARE command.

Related Commands

ADD DYNAMIC CONSTRAINTS

COMPARE

DELETE DYNAMIC CONSTRAINTS

PROVE

REPORT ENVIRONMENT

```
REPort ENvironment
     [ |-Setup | -MOdeling | -MApping | -COMpare | -Diagnosis | -FUnctiondefault]
     [ |-BOTH | -GOLDen | -REVised]
     (Setup / LEC Mode)
```

Displays global settings for the Golden and Revised designs and system settings. The default is to report all global settings.

Tcl Command

```
report_environment
```

Parameters

`-Setup`	Reports environment related to Setup.
`-MOdeling`	Reports environment related to Modeling.
`-MApping`	Reports environment related to Mapping.
`-COMpare`	Reports environment related to Compare.
`-Diagnosis`	Reports environment related to Diagnosis.
`-FUnctiondefault`	Reports environment related to the default return value.
`-BOTH`	Reports environment related to Setup, Modeling, Mapping, Compare, Diagnosis, and the default return value in both the golden and revised designs. This is the default.
`-GOLDen`	Reports environment related to Setup in the golden design.
`-REVised`	Reports environment related to Setup in the revised design.

Related Commands

SET CASE SENSITIVITY

SET COMPARE EFFORT

SET CPU LIMIT

SET FLATTEN MODEL

SET GATE REPORT

SET IMPLEMENTATION

SET LOG FILE

SET MAPPING METHOD

SET NAMING RULE

SET ROOT MODULE

SET SCREEN DISPLAY

SET SYSTEM MODE

SET UNDEFINED CELL

SET UNDRIVEN SIGNAL

SET WIRE RESOLUTION

SET X CONVERSION

REPORT FLOATING SIGNALS

```
REPort FLoating Signals
     [-ROot | -Module <module_name> | -All]
     [-UNDriven | -UNUsed]
     [ | -Net | -Pin [-Direction] ]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays all floating signals in the Golden and Revised designs or in specified modules of a design. The reported floating signals are either nets or pins and are either undriven or unused. Use the `SET UNDRIVEN SIGNAL` command to specify the global behavior of the undriven floating signals in the Golden and Revised designs.

Tcl Command

```
report_floating_signals
```

Parameters

`-ROot`	Displays all floating signals in the root module. *This is the default.*
`-Module <module_name>`	Displays all floating signals in the specified module within the given defaults.
`-All`	Displays all floating signals in "all" design modules within the given defaults.
`-UNDriven`	Displays only undriven floating signals. *This is the default.*
`-UNUsed`	Displays only unused floating signals.
`-Net`	Displays only floating nets. If you do not specify `-net` or `-pin`, Conformal displays both floating nets and floating pins.

`-Pin [-Direction]`	Displays only floating pins. If you do not specify `-net` or `-pin`, Conformal displays both floating nets and floating pins. If `-Direction` is used, Conformal will also display the pin direction information.
`-Both`	Displays floating signals from both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays floating signals from the Golden design.
`-Revised`	Displays floating signals from the Revised design.

Related Commands

ADD TIED SIGNALS

SET UNDRIVEN SIGNAL

REPORT GATE

```
REPort GAte
     <identifier>
     [-INStance | -Pin | -Net | -ID]
     [-ALIAS]
     [-Golden | -Revised]
     [-SUPport]
     [-FRONTIER]
     [-FANIn <integer>]
     [-FANOut <integer>]
     [-UNReach]
     [-SHORT_list | -NOSHORT_list]
     [-SOURCE]
     [-Collapse]
     [-MAP_NAME]
     [-NODYNamic | -DYNamic]
     [-INDent <integer>]
     [-Type <PI | 0 | 1 | E | Z | BBOX | DFF | DLAT | CUT | OUT | COMB | PO>]
     [-RETention]
     [-CORRespondence]
     [-SUMmary]
     [-CONSTRAINT]
     [-ASSERTION]
     [-ALLDNET]
     [-SETUP_INFO]
     (LEC Mode)
```

Displays flattened gate information. By default, it reports the gate ID, type, name, and its fanins and fan-outs at the primitive level. After you specify options for the initial report, use the `REPORT GATE` command without options to generate a report on the same gates, or specify new options as needed.

Important

ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.

Tcl Command

```
report_gate
```

Parameters

`<identifier>`	If you do not specify one of the following options, Conformal automatically determines if the identifier is a number or a path. In the case of a number, Conformal uses the `-id` option; otherwise, Conformal searches for the gate with the `-instance`, `-pin`, or `-net` option; in this respective order.
`-INStance`	Instance path. *This is the default.*
`-Pin`	Pin path
`-Net`	Net path
`-ID`	Gate identification number The identification number is an integer assigned automatically by Conformal. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`-ALIAS`	Report the name alias for the specified gate.
`-Golden`	The identifier is in the Golden design. *This is the default.*
`-Revised`	The identifier is in the Revised design.
`-SUPport`	Reports the supported key points from the fanin cone.
`-FRONTIER`	Reports the frontier key points from the fan-out cone.
`-FANIn <integer>`	Reports this number of levels in the fanin cone. *The default value is 0.*
`-FANOut <integer>`	Reports this number of levels in the fan-out cone. *The default value is 0.*
`-UNReach`	Displays diagnosis information for unmapped points that were classified as unreachable.
`-SHORT_list`	Lists the first and last 20 gates of a long display list. *This is the default.*
`-NOSHORT_list`	Displays the entire display list.
`-SOURCE`	Reports the gate information, which includes the module name, instance name, filename, and source line.

`-Collapse`	Do not report inverters and buffers in the fanin cone. *The default is to report all inverters and buffers in the fanin/fan-out cone.*
`-MAP_NAME`	Displays the gate's used names during mapping.
`-NODYNamic`	Use this option in conjunction with the `-fanin` option. The fanin cone does not stop at a gate with dynamic constraints. *This is the default.*
`-DYNamic`	Use this option in conjunction with the `-fanin` option. The fanin cone stops at the gate with dynamic constraints.
`-INDent <integer>`	Displays this amount of whitespace when reporting the fanin and fan-out cones. *The default value is 2.*
`-Type`	Reports all gates with the specified gate type. The available gate types are as follows: `PI`: Primary inputs `0`: TIE-0 gates `1`: TIE-1 gates `E`: TIE-E gates `Z`: TIE-Z gates `BBOX`: Blackboxes `DFF`: D flip-flops `DLAT`: D-latches `CUT`: Artificial gates for breaking combinational feedback loops `OUT`: Artificial gates for the multiple outputs of blackboxes `COMB`: Combinational gates `PO`: Primary Outputs

`-RETention`	**Note:** This is a Conformal Low Power option. If the gate is a sequential element (DFF or DLAT) and belongs to the Golden Design, this option reports the tag-name (if any) associated with the DFF or DLAT. If the gate is a sequential element (DFF or DLAT) and belongs to the Revised Design, this option reports the power gating cell attribute (if any) associated with the DFF or DLAT. For non-sequential elements, nothing is reported.
`-CORRespondence`	Reports the correspondence gates in the other (Golden or Revised design) netlist. The correspondence gate is potentially equivalent with the gate specified in this command. Use the PROVE command to formally prove the equivalence.
`-SUMmary`	Reports gate type statistics. *The default is not to report the statistics.*
`-CONSTRAINT`	Displays the gates in the netlist that represent constraint functions.
`-ASSERTION`	Displays the gates in the netlist that represent assertion functions.
`-ALLDNET`	Displays all the design-level nets.
`-SETUP_INFO`	Reports setup information from the `READ SETUP INFORMATION` command for the specified gate.

Related Commands

ADD NAME ALIAS

BACKWARD

CHANGE GATE TYPE

FORWARD

REPORT PATH

SET GATE REPORT

REPORT GUIDE INFORMATION

```
REPort GUide Information
     [-SUMMary | -VERBose]
     (Setup Mode)
```

Reports guide information and processing status. See Example below.

Tcl Command

```
report_guide_information
```

Parameters

`-SUMMary`	Displays a summarized report of the guide information and processing status. *This is the default*. See Example below.
`-VERBose`	Displays a verbose report of the guide information and processing status. See Example below.

Example

In summary mode, this command categorizes the statistics for guide information. The following is an example of the default report:

```
======================+=========+==============================+==========
|         |          Processed          |
|         +---------+----------+---------+ Not Yet
Information Type      |  Total   | Applied | Rejected | Ignored | Processed
----------------------+---------+---------+----------+---------+----------
instance_module_name  |  100     | 100     | 0        | 0       | 0
uniquify              |  2300    | 2200    | 80       | 10      | 10
ungroup               |  1800    | 1800    | 0        | 0       | 0
group                 |  10      | 9       | 1        | 0       | 0
share transform       |  30      | 28      | 1        | 1       | 0
speculation transform |  15      | 10      | 2        | 0       | 0
tree-typed transform  |  10000   | 9999    | 0        | 1       | 0
sequential merge      |  280     | 0       | 0        | 0       | 280
======================+=========+=========+==========+=========+==========
```

When you use the `-VERBose` option, this command returns detailed information for each module:

```
module mid_w32 :
[A] 1  : tree  : t_rc.ovf:1
module tst :
[A] ungroup u1 : t_rc.ovf:22
[A] ungroup u2 : t_rc.ovf:29
[I] 1  : tree  : t_rc.ovf:1
[R] 2  : tree  : t_rc.ovf:1
[A] 3  : share : t_rc.ovf:36
```

Where the letter in square brackets indicates the processing status of the guide information:

A : Applied
R : Rejected
I : Ignored
N : Not yet be processed

Related Commands

READ GUIDE FILE

APPLY GUIDED TRANSFORMATIONS

REPORT HIER_COMPARE RESULT

```
REPort HIer_compare Result
     [-Summary | -Equivalent | -NONEQuivalent | -Abort
        | -UNcompared | -FLattened | -DYNamicflattened
        | -EXTRA_po | -ALL]
     [-DEtail]
     [-USage]
     (Setup / LEC Mode)
```

Displays the results of the hierarchical comparison. If the `WRITE HIER_COMPARE DOFILE` command is used, this command is automatically placed at the end of the hierarchical dofile script. It lists the summary results and any modules that are nonequivalent, aborted, or uncompared.

Tcl Command

```
report_hier_compare_result
```

Parameters

`-Summary`	Displays a summary table of the hierarchical comparison results. *This is the default.*
`-Equivalent`	Displays only the hierarchical modules that are equivalent.
`-NONEQuivalent`	Displays only the hierarchical modules that are nonequivalent.
`-Abort`	Displays only the hierarchical modules that had abort key points.
`-UNcompared`	Displays only the hierarchical modules that are not compared.
`-FLattened`	Displays only the hierarchical modules that were found to be nonequivalent and, as a result, were flattened. Use this option when you have used `WRITE HIER_COMPARE DOFILE -conditional`

`-DYNamicflattened`	Displays only the hierarchical modules that were either found to be nonequivalent, or were equivalent but caused nonequivalence at the parent level, and were automatically flattened. Use this option when performing hierarchical comparison with the `RUN HIER_COMPARE` command.
`-EXTRA_po`	Displays only the hierarchical modules that have extra (not-mapped) primary outputs.
`-ALL`	Displays the results of all of the modules.
`-DEtail`	Displays the results in detail. This option is only effective if you have executed the `SET PROJECT NAME` command at the start of the LEC run.
`-USage`	Displays the CPU use time for each module comparison.

Related Commands

RESET HIER_COMPARE RESULT

RUN HIER_COMPARE

SAVE HIER_COMPARE RESULT

WRITE HIER_COMPARE DOFILE

REPORT IGNORED GRID

```
REPort IGnored Grid
     [-Both | -Golden | -REvised]
     [-ALL | -SUPply | -MODule | -INStance]
     (Setup Mode)
```

Note: This is a Conformal Low Power command and an 1801 feature.

Report ignored supply objects, instances, or modules added by command `ADD IGnored Grid`.

Tcl Command

```
report_ignored_grid
```

Parameters

`-Both`	Reports ignored objects of both Golden netlist and Revised netlist. This is the default.
`-Golden`	Reports the ignored objects of Golden netlist.
`-Revised`	Reports the ignored objects of Revised netlist.
`-ALL`	Reports all ignored objects, including supply objects, instances, and modules. *This is the default*.
`-SUPply`	Reports ignored supply objects. Which are supply pins in the design netlist or virtual supply ports created in the 1801 power intent.
`-Module`	Reports ignored modules.
`-INStance`	Reports ignored instances.

Related Commands

ADD IGNORED GRID

DELETE IGNORED GRID

COMPARE POWER GRID

REPORT COMPARED POWER_GRID

REPORT IGNORED INPUTS

```
REPort IGnored Inputs
     [-ROot | -Module <module_name> | -All]
     [-Both | -Golden | -REvised]
     (Setup / LEC Mode)
```

Displays the input pins, which were added as ignored inputs, in the Golden and Revised designs. These pins were originally specified with the `ADD IGNORED INPUTS` command.

Tcl Command

`report_ignored_inputs`

Parameters

`-ROot`	Displays only the input pins in the root module. *This is the default.*
`-Module <module_name>`	
	Displays only the ignored input pins in the named module.
`-All`	Displays all ignored input pins in all modules within the given defaults.
`-Both`	Displays both the Golden and Revised added ignored inputs. *This is the default.*
`-Golden`	Displays the added ignored inputs from the Golden design.
`-REvised`	Displays the Revised ignored inputs.

Related Commands

ADD IGNORED INPUTS

DELETE IGNORED INPUTS

REPORT IGNORED OUTPUTS

```
REPort IGnored Outputs
     [-ROot | -Module <module_name> | -All]
     [-Both | -Golden | -REvised]
     (Setup / LEC Mode)
```

Displays the output or I/O pins, which were added as ignored outputs, in the Golden and Revised designs. These outputs were originally specified with the `ADD IGNORED OUTPUTS` command.

Tcl Command

```
report_ignored_outputs
```

Parameters

`-ROot`	Displays only the input pins in the root module. *This is the default.*
`-Module <module_name>`	Displays only the ignored output or I/O pins in the specified module.
`-All`	Displays all ignored input pins in all modules within the given defaults.
`-Both`	Displays both the Golden and Revised added ignored outputs. *This is the default.*
`-Golden`	Displays only the Golden added ignored outputs.
`-REvised`	Displays only the Revised ignored outputs.

Related Commands

ADD IGNORED OUTPUTS

DELETE IGNORED OUTPUTS

REPORT IMPLEMENTATION INFORMATION

```
REPort IMplementation Information
     [-SUMmary | -NOSUMmary]
     [-VERbose
       [<identifier>] [-FV_NAMes | -LEC_NAMes | -GATEIDs]
       [-BOTh | -GOLden | -REVised]
       [-USAge | -NOUSAge <ALL | USEd | UNUsed | REJected>]
       [-INFo_types | -NOINFo_types
         <ALL | PHASE | SEQ_Constant | SEQ_Merge | SEQ_Unreachable |
         SEQ_Duplication | BOUNDary_optimization |
         SYNTHesis_equation | LOOP_Breaker>
       ]
     ]
     (Setup/LEC Mode)
```

This command can only be used after the `READ_IMPLEMENTATION_INFORMATION` command. This command works in both flat and hierarchical flow.

Tcl Command

```
report_implementation_information
```

Parameters

`-SUMmary`	Displays the summary of the implementation information read through the `READ_IMPLEMENTATION_INFORMATION` command. *This is the default.*
`-NOSUMmary`	Does not display the summary.

`-VERbose`	Displays the detail information; this option must be specified to allow the following options to work. When `-VERbose` is specified without any of the following options, the default report is based on:
	`report_implementation_information -VERbose * -FV_NAMes -BOTh \`
	`-USAge UNUsed REJected \`
	`-INFo_types SEQ_Constant SEQ_Merge SEQ_Duplication SYNTHesis_equation LOOP_Breaker`

`<identifier>`	Specifies one or several identifier(s) to filter out the verbose information. Users can specify full or part of instance path names here. Wildcard is supported when using with `-FV_NAMes`. Multiple identifiers are supported. When more than one identifier is specified, LEC displays all the implementation info matches any one of the specified identifiers.
`-FV_NAMes`	The specified identifier is FV name. *This is the default*.
`-LEC_NAMes`	The specified identifier is LEC elaborated name. Design data needs to be ready to use this option.
`-GATEIDs`	The specified identifier is LEC gate id. Design data needs to be ready to use this option.
`-BOTh`	Displays both information that is applied to Golden and Revised. *This is the default*.
`-GOLden`	Only displays the information that is applied to Golden.
`-REVised`	Only displays the information that is applied to Revised.
`-USAge`	Displays the verbose information with the specified usage status. The default is `-USAge UNUSED REJected` when `-USAge` and `-NOUSAge` are not specified.
`USEd`	Indicates the information has been applied.
`UNUSEd`	Indicates the information is not used. Most likely that the corresponding objects cannot be found in the design.
`REJected`	Indicates the information has been proven incorrect.
`ALL`	All the usage status.
`-NOUSAge`	Displays all type of usages except the specified ones.
`-INFo_types`	Displays the verbose information with the specified information type. If `-INFo_types` and `-NOINFo_types` are not specified, the default value is `-INFo_types SEQ_Constant SEQ_Merge SEQ_Duplication SYNTHesis_equation LOOP_Breaker`.

`-NOINFo_types`	Displays all type of information except specified ones.
	`ALL`: All types of information
	`PHASE`: Sequential phase for mapping
	`SEQ_Constant`: Sequential constant
	`SEQ_Merge`: Sequential merge
	`SEQ_Unreachable`: Sequential unreachable
	`SEQ_Duplication`: Sequential duplication
	`BOUNDary_optimization`: Boundary optimization
	`SYNTHesis_equation`: Datapath synthesis equation
	`LOOP_Breaker`: Combinational loop breaker

Related Commands

READ IMPLEMENTATION INFORMATION

REPORT INSTANCE ATTRIBUTE

```
REPort INstance Attribute
     [-ROot | -Module <module_name> | -All]
     [-Summary | -Verbose]
     [-Both | -Golden | -REvised]
     (Setup Mode)
```

Displays the attributes placed on instances in the Golden and Revised designs. These attributes were originally specified with the `ADD INSTANCE ATTRIBUTE` command.

Tcl Command

```
report_instance_attribute
```

Parameters

`-ROot`	Displays only the added instance attributes in the root module. *This is the default.*
`-Module <module_name>`	Displays only the added instance attributes in the specified module.
`-All`	Displays all added instance attributes within the given defaults.
`-Summary`	Displays a summary message of the total number of added instance attributes. *This is the default.*
`-Verbose`	Displays all added instance attributes.
`-Both`	Displays the added instance attributes in both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the added instance attributes in the Golden design.
`-REvised`	Displays the added instance attributes in the Revised design.

Related Commands

ADD INSTANCE ATTRIBUTE

DELETE INSTANCE ATTRIBUTE

REPORT INSTANCE BINDING

```
REPort INstance Binding
     [-MODULEname]
     [-ORIGMODULEname]
     [-SOURCEfile]
     [-ALLinstance | -BOUndbycfg | -NOTBoundbycfg]
     [-BOTh | -GOLden | -REVised]
     (Setup / LEC Mode)
```

Displays design instance on the Golden and Revised designs.It display the design instance full hierarchical name (include root module name) and its binding module's user-defined name, Conformal uniquified/parameterized name, source file name, and library name.

The output format is
full hierarchical instance name + [user-defined module name] +
[conformal uniquified/parameterized module name] + [source file name] +
[library name] + [bound by sv configuration]

Tcl Command

```
report_instance_binding
```

Parameters

`-MODULEname`	Report reference module Conformal generated uniquified/ parameterized name.
`-ORIGMODULEname`	Report reference module user-defined name.
`-SOURCEfile`	Report reference module source filename.
`-LIBname`	Report the library name where the reference module is belonged.
`-ALLinstance`	CFM will list entire design instance with the other information `[-MODULEname]` `[-ORIGMODULEname]` `[-SOURCEfile]` `[-LIBname]` based on user setting. *This is the default*.
`-BOUndbycfg`	CFM will list instance which is bound by configuration file only, and with the other information `[-MODULEname]` `[-ORIGMODULEname]` `[-SOURCEfile]` `[-LIBname]` based on user setting.

`-NOTBoundbycfg`	CFM will list instance which is not bound by configuration file, and with the other information `[-MODULEname] [-ORIGMODULEname] [-SOURCEfile] [-LIBname]` based on user setting.
`-BOTH`	Report design instance data on both the Golden and Revised designs. *This is the default*.
`-Golden`	Report design instance data on the Golden design.
`-Revised`	Report design instance data on the Revised design.

Related Commands

REPORT INSTANCE CONSTRAINTS

```
REPort INstance Constraints
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays the constraints placed on instances in the Golden and Revised designs. These constraints were originally specified with the `ADD INSTANCE CONSTRAINTS` command.

Tcl Command

```
report_instance_constraints
```

Parameters

`-Both`	Displays the instance constraints in both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the instance constraints in the Golden design.
`-Revised`	Displays the instance constraints in the Revised design.

Related Commands

ADD INSTANCE CONSTRAINTS

DELETE INSTANCE CONSTRAINTS

REPORT INSTANCE EQUIVALENCES

```
REPort INstance Equivalences
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays the equivalences placed on instances in the Golden and Revised designs. These equivalences were originally specified with the `ADD INSTANCE EQUIVALENCES` command.

Tcl Command

```
report_instance_equivalences
```

Parameters

`-Both`	Displays the instance equivalences in both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the instance equivalences in the Golden design.
`-Revised`	Displays the instance equivalences in the Revised design.

Related Commands

ADD INSTANCE EQUIVALENCES

DELETE INSTANCE EQUIVALENCES

REPORT INSTANCE RENAMING

```
REPort INstance Renaming
     [<rule_name*> ... ]
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Reports the details for the specified instance renaming rule (added through the `ADD INSTANCE RENAMING COMMAND`). When used without any options, reports details for all instance renaming rules.

Tcl Command

```
report_instance_renaming
```

Parameters

<rule_name>*	Reports the specified instance renaming rule. Simple wildcard matching is supported.

Example

```
SETUP> report instance renaming *
================================================================================
Num       Rule Name          Original Pattern    Renamed Pattern
================================================================================
1         r1                 u1/x1               x1
2         r2                 u1/x2               x2
3         R1                 a/b/c               a/c
4         R2                 x/y/z               x/z
================================================================================
// Note: There are 4 instance renaming rule(s) found.

SETUP> report instance renaming R*
================================================================================
Num       Rule Name          Original Pattern    Renamed Pattern
================================================================================
1         R1                 a/b/c               a/c
2         R2                 x/y/z               x/z
```

```
================================================================================
// Note: There are 2 instance renaming rule(s) found.
```

Related Commands

ADD INSTANCE RENAMING

ADD MAPPING MODEL

DELETE INSTANCE RENAMING

DELETE MAPPING MODEL

MOS2BUFIF

REPORT MESSAGES

REPORT KEY POINT

```
REPort KEy Point
     [-DC]
     [-PROPerty [-SUMmary]]
     [-SIMULATION <file_name>]
     [[-TYpe <PI | E | Z | DFf | DLat | CUt | BBox | PO> ...
       | -NOTYpe <PI | E | Z | DFf | DLat | CUt | BBox | PO> ...]
       | -Mapped | -UNMapped | -UNReached]
     [-Golden | -REVised]
     (LEC Mode)
```

Report key points in the design.

Tcl Command

```
report_key_point
```

Parameters

`-DC`	Displays the number of DC gates in the compare cone	
`-PROPerty`	Displays support and fan-out key points for each key point	
	-SUMmary	Skips support and fan-out key points and displays only the summary table for each key point.
`-SIMULATION <file_name>`		
	Imports an input vector from the specified file and displays the corresponding simulation results of the key points. With `-verbose`, imports an input vector from the file named `<string>`, displays the input vectors, and displays the corresponding simulation results of the key points. This command can re-simulate the output file from the command `report test vector -file <file_name>`.	
`-TYpe`	Displays all key points with the specified type.	
`-NOTYpe`	Displays all key points except the specified type.	

	The available types for `-Type` and `-NOType` are as follows:
	`PI`: Primary inputs
	`E`: TIE-E gates
	`Z`: TIE-Z gates
	`DFf`: D flip-flops
	`DLat`: D-latches
	`CUt`: Artificial gates for breaking combinational feedback loops
	`BBox`: Blackboxes
	`PO`: Primary Outputs
`-Mapped`	Displays all mapped key points in the design.
`-UNMapped`	Displays all unmapped key points in the design.
`-UNReached`	Displays diagnosis information for unmapped key points that were classified as unreachable.
`-Golden`	Specifies that the report applies only to the Golden design. *This is the default.*
`-Revised`	Specifies that the report applies only to the Revised design.

Related Commands

MAP KEY POINTS

REPORT LIBERTY_COMPARE FILTER

```
REPort LIberty_compare Filter
     [<filter_name*> ... ]
     (Setup/LEC Mode)
```

Description

Reports the Liberty compare filters added by the `ADD LIBERTY_COMPARE FILTER` command.

Tcl Command

`report_liberty_compare_filter`

Paramters

`<filter_name* ...>`	Specifies the name of the filter(s) to report. By default, this command reports all Liberty compare filters. Wildcards are supported.

Examples

- The following command reports all Liberty compare filters:

  ```
  report liberty_compare filter
  ```

- The following command reports all Liberty filters whose name matches "`related_filter*`":

  ```
  report liberty_compare filter related_filter*
  ```

Related Commands

ADD LIBERTY_COMPARE FILTER

COMPARE LIBERTY

DELETE LIBERTY_COMPARE FILTER

REPORT COMPARED LIBERTY

REPORT LIBRARY DATA

```
REPort LIbrary Data
      [-SKIP_Unref]
      [-SORT <NAME | REFerence | INStance>]
      [-Source]
      [-Golden | -Revised]
      (Setup / LEC Mode)
```

Displays the following columns:

- *ID*: Specifies the cell ID.
- *Name*: Specifies the cell name.

  Cost: Specifies the cost of each library cell, which is the product of the number of instances of primitive gates within each library cell (`Ins`) and the number of times the library cell is instantiated (`Ref`). This total is calculated by *Ins* times *Ref*. If *Ins* is 3 and *Ref* is 3, the total is 9. If *Ins* is 3 and *Ref* is 0, the total is 0.
- *Ins*: Displays the number of instances of primitive gates within each library cell.
- *Ref*: Displays the number of times the library cell is referenced in the design.
- *TOT*: Displays the total number of gates per library cell.
- *DFF*: Specifies whether the cell contains a D flip-flop.
- *DLAT*: Specifies whether the cell contains a D-latch.
- *BUF*: Specifies whether the cell contains a buffer.
- *NOT*: Specifies whether the cell contains an inverter gate.
- *BBOX*: Specifies whether the cell contains a blackbox.
- *UDP*: Specifies whether the cell is a UDP.

By default, Conformal reports on the library for the Golden design if you do not specify options.

Note: When you read in a library, you can specify whether it is for the Golden or Revised design.

Tcl Command

```
report_library_data
```

Parameters

`-SKIP_Unref`	Do not display unreferenced library modules.	
`-SORT`	Sorts report data as specified:	
	`NAME`	Sorts report data alphabetically by library cell name.
	`REFerence`	Sorts report data according to the Reference column, in descending order.
	`INStance`	Sorts report data according to the Instance column in descending order.
`-SOURCE`	Displays the source filename and line number for each cell.	
`-Golden`	Displays the library data for the Golden design. *This is the default.*	
`-Revised`	Displays the library data for the Revised design.	

Related Commands

READ LIBRARY

READ DESIGN

REPORT DESIGN DATA

REPORT LOWPOWER CELLS

```
REPort LOwpower Cells
     [-Module | -Instance]
     [-Summary]
     (Setup / LEC Mode)
```

Note: This is a Conformal Low Power command.

Reports the low power cells used in the design.

Tcl Command

```
report_lowpower_cells
```

Parameters

`-Module`	Reports only the modules with low power cells. *This is the default.*
`-Instance`	Reports only the instances with low power cells.
`-Summary`	Displays a summary of low power cells.

Related Commands

ADD LOWPOWER CELLS

CHECK LOWPOWER CELLS

DELETE LOWPOWER CELLS

REPORT LOWPOWER DATA

SET LOWPOWER OPTION

REPORT LOWPOWER DATA

```
REPort LOwpower Data
     [-STatus <All | Pass | Fail | Notcheck>]
     [-SUMmary | -Verbose]
     (LEC Mode)
```

Note: This is a Conformal Low Power command.

Reports the low power data. These are the results of the low power check performed on low power cells using the `CHECK LOWPOWER CELLS` command.

For a description of the default rules that are added by the system, see CHECK LOWPOWER CELLS.

Tcl Command

```
report_lowpower_data
```

Parameters

`-STatus`	Specifies the status reporting. For retention-register cell types, the `-STatus` arguments are described as follows: `All`: Reports all the sequential pairs (LEC mapped points) that passed or failed the default rule or user rule. *This is the default.* `Pass`: Reports all the sequential pairs that passed the default rule or user rule `Fail`: Reports all the sequential pairs that failed the default rule or user rule `Notcheck`: Reports the sequential pairs that were not checked for retention-register consistency

For isolation and level-shifter cell types, the `-STatus` arguments are described as follows:

`All`: Reports all the low power cut gates that passed or failed the technology mapping check. *This is the default.*

`Pass`: Reports all the low power cut gates that passed the technology mapping check

`Fail`: Reports all the low power cut gates that failed the technology mapping check

`Notcheck`: Reports the low power cut gates that were not checked for isolation and level-shifter consistency

For the power domain consistency check, the `-STatus` arguments are described as follows:

`All`: Reports all the mapped sequential points that passed and failed the power domain consistency check. *This is the default.*

`Pass`: Reports the mapped sequential points that passed the power domain consistency check

`Fail`: Reports the mapped sequential points that failed the power domain consistency check

`Notcheck`: Reports the mapped sequential points that were not checked for power domain consistency

`-TYpe`	Specifies the module type reporting.

`All`	Reports on all low power cells. *This is the default.*
`Retention [-Rule <rulename>]`	Reports on only the low power state retention cells. `-Rule <rulename>` reports all the sequential pairs that passed or failed the specified `rulename`.
`-Isolation_cells`	Reports on only the low power isolation cells.
`-Level_shifter_cells`	Reports on only the low power level-shifter cells.

`-POWER_domain_check`

Reports the results of power domain consistency check for the mapped sequential points.

`-SUMmary`	Displays the status summary of the check performed on low power cells. *This is the default.*
`-Verbose`	For state retention cells, this reports the sequential pairs (LEC mapped points) that passed or failed the default rule or user rule. For each passed or failed sequential pair, the corresponding rule it passed or failed on is also reported. In addition, this reports any tag-name for the sequential element in the Golden Design, and any power gating cell attribute for the sequential element in the Revised Design. For isolation cells and level-shifter cells, this reports the `PASS` or `FAIL` status of low power cut gates that correspond to the isolation cells and level-shifter cells.

Related Commands

ADD LOWPOWER CELLS

CHECK LOWPOWER CELLS

DELETE LOWPOWER CELLS

REPORT LOWPOWER CELLS

SET LOWPOWER OPTION

REPORT LOWPOWER INFO

```
REPort LOwpower Info
     < <-INSTance | -PIN | -NET | -STRategy | -LIBcell> <name*>... >
     [-ALL | -DOMAIN_ASSIGNMENT | -SUPPLY_SET | -RELATED_PG_PIN
     | -PROPAGATED_SUPPLY | -CLAMP_VALUE | -MATCHING_STRATEGY
     | -TARGETING_STRATEGY
     | -LIBRARY_CELL ]
     [-DEDICATED_iso]
     [-ELABORATED]
     [-LIMit <num>]
     [-LP | -LSH | -ISO | -COMBO | -RET | -PSW | -AON | -MACRO]
     [-MAX_DRIver_load <num>]
     [-MAX_LIBcell_instance <num>]
     [-NON_DEDICATED_iso]
     [-REAL | -VIRTUAL]
     [-SINGLE_PIN_ret]
     [-SUPPLY | -LOGIC]
     [-TWO_PIN_ret]
     [-Verbose]
     [-ZERO_PIN_ret]
     [-Golden | -Revised | -Both]
```
(Setup/LEC Mode)

Note: This is a Conformal Low Power command and an 1801 feature.

Reports low power information for different types of design objects (real objects), objects inferred by low power intent (virtual objects), or low power strategies specified by power intent.

When using wildcards, the choice of objects for which to display selected information can be narrowed using filters, such as a filter to display only information for low power cells, filter to display only information about objects in the design or only objects inferred by power intent.

Tcl Command

```
report_lowpower_info
```

Parameters

`-INSTance`	Specifies that the object is an instance. Instances can be either design instances or instances inferred by power intent strategies.

`-PIN`	Specifies that the object is a pin. Pins can be design pins or pins inferred by power intent.
`-NET`	Specifies that the object is a net. Nets can be design nets or nets inferred by power intent.
`-STRategy`	Specifies that the object is a power intent strategy.
`-LIBcell`	Specifies that the object is a library cell.
`-ALL`	Display all available information for the specified object. *This is the default.*
`-DOMAIN_ASSIGNMENT`	Display the object's low power domain (applies only to instances).
`-SUPPLY_SET`	Displays the object's assigned supply set (applies only when the object is a logic pin or an instance). For instances, the information will be displayed for all logic pins of the instance. For a given logic pin, this option displays the driver and receiver supply sets. If `-Verbose` is used, this also lists the driver/ receiver pins. To limit the number of displayed driver/receiver pins, use `-MAX_driver_load`.
`-RELATED_PG_PIN`	Displays the related pg pins. Applies only when the object is a logic pin or an instance. For instances, the information will be displayed for all logic pins of the instance. This information is displayed only for objects with physical use.
`-PROPAGATED_SUPPLY`	Displays the propagated supply. Applies only when the object is a supply pin, supply net, or an instance. For instances, the information will be displayed for all supply pins of the instance. This information is displayed only for objects with physical use.
`-CLAMP_VALUE`	Displays the isolation cell clamping values. Applies only when the object is a cell with isolation clamping function.
`-MATCHING_STRATEGY`	Displays the low power strategies. Applies only when the object is a low power instance or a logic pin. For low power instances, this option reports the matching strategies used to infer or implement the instance. For a logic pin, this option reports the strategies that act on that pin.
`-TARGETING_STRATEGY`	Displays the low power strategies. Applies only when the object is a domain boundary pin. This option reports all strategies acting on the pin and, for each strategy, the specific filters or reasons that strategy could not be applied . When a strategy is implementable, it reports the instances inferred by the strategy.

`-LIBRARY_CELL`	Displays extracted low power cells. Applies only when the object is a low power instance.
`-DEDICATED_iso`	Displays information for dedicated isolation instances.
`-ELABORATED`	Displays information for library cells in the elaborated design.
`-LIMIT <num>`	Specifies the maximum number of objects to display when using wildcards in the object name. Default is 0, which is used to report all matching objects.
`-LP`	Displays information for low power instances.
`-LSH`	Displays information for level-shifter instances.
`-ISO`	Displays information for isolation instances.
`-COMBO`	Displays information for isolation and level-shifter combination instances.
`-RET`	Displays information for retention instances.
`-PSW`	Displays information for power-switch instances.
`-AON`	Displays information for always-on instances.
`-MACRO`	Displays information for macro instances.
`-MAX_DRIver_load <num>`	Specifies the maximum number of driver/receiver pins to display when using the `-verbose` option. Default is 1.
`-MAX_LIBcell_instance <num>`	Specifies the maximum number of library cell instances to display when reporting a given library cell. Default is 1 and assigning 0 will display all instances.
`-NON_DEDICATED_iso`	Displays information for non-dedicated isolation instances.
`-REAL`	Display information only for objects in the design.
`-VIRTUAL`	Display information only for objects inferred from power intent that are not already in the design.
`-SINGLE_PIN_ret`	Displays information for single-pin retention instances.
`-SUPPLY`	Displays information only for supply pins and supply nets.
`-LOGIC`	Displays information only for logic pins and logic nets.
`-TWO_PIN_ret`	Displays information for two-pin retention instances.

`-Verbose`	Displays additional information. For instance, prints all driver/receiver pins when reporting the supply set of logic pins or prints more detailed information when the object to report is a strategy (such as strategy filters, the elements of strategy as well as the low power instance names for implementable elements).
`-ZERO_PIN_ret`	Displays information for zero-pin retention instances.
`-Golden`	Reports information for the Golden design.
`-Revised`	Reports information for the Revised design.

Example

The following command displays information about strategy TDR.iso1.

```
SETUP> report lowpower info -strategy TDR.iso1 -v
```

The resulting report displays information about the following areas:

- Overridden Elements: Not inferred because they have been overridden by a higher precedence strategy.
- Multiple Strategies Elements: Not inferred because they have the same precedence in multiple strategies.
- Tied Signal Elements: Not inferred because the constant crossing is localized based on "`set_lowpower_option -constant_signal`" setting.
- Non-implementable Elements: Not inferred because they cannot be implemented without changing the design structure.
- Undefined Supplies Elements: Not inferred because the related supply set of a logic pin of the inferred instance cannot be determined from the strategy or using the applicable default.
- Bad Location Elements: Not inferred because the location specified by the strategy cannot be honored (ex: outside the design scope).
- Optimized Away Elements: Not inferred because the –instance option of the strategy specifies an empty string for instance name, meaning the instance has been optimized away and should not be re-implemented
- No Insertion Elements: Not inferred because the reported strategy has the highest precedence and it specifies one of the following options: `-no_isolation`, `-no_shift`, `-no_retention`

- Floating Port Elements: Not inferred because they do not have logic receivers
- Undriven Port Elements: Not inferred because they do not have logic drivers
- Analog Port Elements: Not inferred because they are connected to analog signals
- Zero Pin Retention Elements: Not inferred because isolation strategy applied to the zero-pin retention clock, set or reset pin is dropped. For isolation insertions applied to zero-pin retention clock, set, or reset pin, the tool derives the required isolation insertion strategy from the retention strategy. Hence, the isolation strategy applied to the clock/set/reset of the zero-pin retention cell is ignored
- Antenna Diode Elements: Not inferred because all receivers of the domain boundary pin are antenna diode pins.
- Clamp Value 'any' Elements: Not inferred because the isolation clamp value is 'any' and the tool does not know which isolation cell type should be implemented.
- Implementable Elements: Reports implementable elements, (L) means strategy acts on the lowconn of element, (H) means strategy acts on the highconn of element. Priority of the element for the given strategy is printed as well. For each element, the inferred instance name is printed. The (R) denotes a real instance, instance that is already implemented in the design, while (V) denotes a virtual instance, an instance inferred by power intent.

Related Command

REPORT POWER INTENT

REPORT CROSSING PATH

REPORT LP_CONTROL IGNORED

```
REPort LP_control Ignored
     [-Golden|-Revised|-Both]
     (Setup Mode)
```

Reports the specified ignored low-power control signals.

Users must enable `SET LOWPOWER OPTION -LP_CTRL_SIGNAL_COMPARED_PAIR` before `READ POWER INTENT` to use this command.

This command works after `READ POWER INTENT` for both designs.

Tcl Command

```
report_lp_control_ignored
```

Parameters

`-Golden`	Reports the specified ignored low-power control signals in the Golden design. *This is the default*.
`-Revised`	Reports the specified ignored low-power control signals in the Revised design.
`-Both`	Reports the specified ignored low-power control signals in both Golden and Revised designs.

Related Commands

SET LOWPOWER OPTION

ADD LP_CONTROL IGNORED

ADD LP_CONTROL PAIR

ANALYZE LP_CONTROL PAIR

REPORT LP_CONTROL PAIR

REPORT LP_CONTROL VERIFICATION

REPORT LP_CONTROL PAIR

```
REPort LP_control Pair
     [-Pair | -NOPair | -Summary]
     (Setup Mode)
```

Reports the specified low-power control compared pairs and not-paired low-power control signals.

Users must enable `SET LOWPOWER OPTION -LP_CTRL_SIGNAL_COMPARED_PAIR` before `READ POwer INTENT` to use this command.

This command works after `READ POWER INTENT` for both designs..

Tcl Command

```
report_lp_control_pair
```

Parameters

`-Pair`	Displays the specified compared pairs of low-power control signals. *This is the default*.
`-NOPair`	Displays the not-paired low-power control signals.
`-Summary`	Displays the paired, not-paired, and ignored low-power control signals.

Related Commands

SET LOWPOWER OPTION

ADD LP_CONTROL IGNORED

ADD LP_CONTROL PAIR

ANALYZE LP_CONTROL PAIR

REPORT LP_CONTROL IGNORED

REPORT LP_CONTROL VERIFICATION

REPORT LP_CONTROL VERIFICATION

```
REPort LP_control Verification
```
(*LEC Mode*)

Reports the comparison and not-compared result of low-power control signals.

Users must enable SET LOwpower Option -LP_CTRL_SIGNAL_COMPARED_PAIR before REAd POwer Intent to activate this command.

The sequence of reporting is:

1. Comparison results of paired low-power control signals between Golden and Revised designs.
2. Comparison results of paired low-power control signals in the Golden design.
3. Comparison results of paired low-power control signals in the Revised design.
4. Not-compared low-power control signals.
5. Summary table of comparison and not-compared result for control signals of low-power strategies.

Tcl Command

```
report_lp_control_verification
```

Related Commands

SET LOWPOWER OPTION

ADD LP_CONTROL IGNORED

ADD LP_CONTROL PAIR

ANALYZE LP_CONTROL PAIR

REPORT LP_CONTROL PAIR

REPORT LP_CONTROL IGNORED

REPORT MAPPED POINTS

```
REPort MApped Points
     [< <gate_id> | <instance_pathname*> | <pin_pathname* ...>>
     | [-TYpe < PI | E | Z | DFf | DLat | CUt| BBox | PO> ...]
     | [-NOTYpe < PI | E | Z | DFf | DLat | CUt| BBox | PO> ...]
     ]
     [-ALIAS_mapped]
     [-BBOX_DIFF_MODNAME]
     [-BBOX_NONCORRESPONDING_PINS]
     [-CLass <Full | System | User>]
     [-FUNCTIONAL_mapped | -NAME_mapped | -NETS_mapped]
     [-GRoup]
     [-INput]
     [-INVert_mapped]
     [-LOng]
     [-METHOD]
     [-OUTput]
     [-REName]
     [-RETention]
     [-SOURCE]
     [-SUMmary]
     [-UNReachable]
     [-Golden | -REVised]
     (LEC Mode)
```

Displays the mapped points that were automatically identified or added with the `ADD MAPPED POINTS` command. Each mapped point from the Golden and Revised design is displayed along with a summary of all Golden and Revised mapped points.

The summary includes the total number of primary inputs, primary outputs, DFFs, D-latches, TIE-Es, TIE-Zs, blackboxes, and cut gates.

If no options are entered, the command default is to display both the *User* and *System* classes of mapped points.

Wildcard: The wildcard (*) represents any zero or more characters in instance or pin paths of mapped points.

Tcl Command

```
report_mapped_points
```

Parameters

`<gate_id>`	Reports the mapped points for the identified gate.
	Note: ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`<instance_pathname*>`	Reports the mapped points for the named instance path(s). This accepts wildcards.
`<pin_pathname*>`	Reports the mapped points for the named pin path(s). This accepts wildcards.
`-Type`	Displays all mapped points with the specified type.
`-NOType`	Do not display mapped points with the specified type.
	The available types for `-Type` and `-NOType` are as follows:
	`PI`: Primary inputs
	`E`: TIE-E gates
	`Z`: TIE-Z gates
	`DFf`: D flip-flops
	`DLat`: D-latches
	`CUt`: Artificial gates for breaking combinational feedback loops
	`BBox`: Blackboxes
	`PO`: Primary Outputs
`-ALIAS_mapped`	Displays all alias-name mapped points
`-BBOX_DIFF_MODNAME`	Reports mapped blackboxes that have different names.
`-BBOX_NONCORRESPONDING_PINS`	
	Reports only mapped blackboxes that have non-corresponding input/output pins.
`-Class`	Displays the following class of mapped points.

`Full`	Mapped points from both the *User* and *System* classes. *This is the default.*
`System`	Mapped points from the original design.

`User`	Mapped points added with the `ADD MAPPED POINTS` command

`-FUNCTIONAL_mapped`	Displays all functionally mapped points.
`-NAME_mapped`	Displays all name mapped points.
`-NETS_mapped`	Displays all mapped points using net names.
`-GRoup`	Displays the mapping pairs in which either the Golden or Revised key point is a *group* of equivalent gates rather than a *single* gate. The group can be defined with the `ADD INSTANCE EQUIVALENCE` command or the `-seq_merge` option of the `SET FLATTEN MODEL` command. A key point group is counted as one key point.
`-INput`	Displays the input port mapping pairs of the specified DFF, DLAT, or blackbox gate.
`-INVert_mapped`	Displays all mapped points with inverted mapping.
`-LOng`	Displays pairs of mapped points on separate lines.
`-OUTput`	Displays the output port mapping pairs of the specified blackbox gate.
`-METHOD`	Shows the method used in mapping the key points.
`-REName`	Lists the keypoint with renaming rules applied to the names.
`-RETention`	**Note:** This is a Conformal Low Power option. If the mapped point is a sequential pair (DFF or DLAT pair), this option reports the status of the mapped point (Pass, Fail, or Unknown) in accordance with the state retention mapping rules. This also reports the tag-name (if any) associated with the Golden DFF or DLAT and the power gating cell attribute (if any) associated with the Revised DFF or DLAT. For non-sequential elements, nothing is reported.
`-SOURCE`	Displays the source file name of user mapped points.
`-SUMmary`	Displays a table summary of the mapped points in the Golden and Revised designs.
`-UNReachable`	Lists unreachable key points. Unreachable key points are those that do not eventually affect the primary output of the design.

`-Golden`	The mapped points are from the Golden design. *This is the default.*
`-REVised`	The mapped points are from the Revised design.

Related Commands

ADD MAPPED POINTS

DELETE MAPPED POINTS

MAP KEY POINTS

REPORT STATISTICS

REPORT UNMAPPED POINTS

SET MAPPING METHOD

SET NAMING RULE

REPORT MAPPING METHOD

`REPort MApping Method`

Displays the global setting of the mapping method and phase. For detailed information of each usage in the report, run man on `SET MAPPING METHOD` command.

Tcl Command

```
report_mapping_method
```

Related Commands

SET MAPPING METHOD

REPORT MAPPING MODEL

```
REPort MApping MOdel
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

The `REPORT MAPPING MODEL` command report the phase information added using the `ADD MAPPING MODEL` command.

Tcl Command

```
report_mapping_model
```

Parameters

`-Golden`	Reports the phase information for the modules in the Golden design. *This is the default*.
`-Revised`	Reports the phase information for the modules in the Revised design.
`-Both`	Reports the phase information for both the Golden and Revised designs.

Related Commands

ADD MAPPING MODEL

DELETE MAPPING MODEL

REPORT MAPPING INFORMATION

```
REPort MAPping Information
     [-Verbose]
     (LEC Mode)
```

This command reports the information of all the mapping key points that are read in using the command `SET ANALYZE OPTION -MAPPING_FILE`.

The report has three categories: Used, Unused and Rejected

- Used: Indicates the information has been applied.
- Unused: Indicates the information is not used because the corresponding object cannot be found in the design.
- Rejected: Indicates the information is rejected because the object is already mapped or violated the mapping rule of unreachable, sequential constant and sequential merge.

Tcl Command

```
report_mapping_information
```

Parameters

`-Verbose`	Lists mapping data in details for all categories. The format is: `<GOLDEN NAME> -> <PHASE> <REVISED_NAME> <FILE_LOCATION>` For example, `g_reg -> - r_reg map_file:8`

Examples

```
TCL_LEC> report_mapping_information -verbose
=============================================================================
            Used       Unused     Rejected
-----------------------------------------------------------------------------
map_info    1          1          0
=============================================================================
Unused information:
-----------------------------------------------------------------------------
g1_reg -> + r1_reg        mapping_file:1
=============================================================================
Used information:
-----------------------------------------------------------------------------
g1_reg -> - r1_reg        mapping_file:1
=============================================================================
```

Related Commands

SET ANALYZE OPTION

REPORT MESSAGES

```
REPort MEssages
     [-MOdeling | -MAPping | -Compare]
     [-NOSORT | -SORT]
     [-RUle <rule_name>]
     [-Summary | -Verbose]
     [-Both | -Golden | -REvised]
     (Setup / LEC Mode)
```

Displays either a summary or complete list of the warning messages that come from the modeling, mapping, or comparison process. (The modeling process occurs when Conformal exits the Setup mode.) A summary of the warning messages is always displayed when the modeling, mapping, or comparison process is in progress; however, this command displays each individual warning message for the Golden and Revised designs, according to your specifications.

See "Modeling Messages" in the *Conformal Equivalence Checking User Guide* for information on the Modeling Messages and the commands/options that trigger them.

Tcl Command

```
report_messages
```

Parameters

`-MOdeling`	Displays only warning messages from the processing and modeling of the Golden and Revised designs. *This is the default.*
`-MAPping`	Displays warning messages only from the automatic key point mapping process.
`-Compare`	Displays warning messages only from the comparison process.
`-NOSORT`	Do not sort messages. *This is the default.*
`-SORT`	Sorts messages alpha-numerically. (Use this option with the `-verbose` option.)
`-RUle <rule_name>`	Displays only the named rule.
`-Summary`	Displays only a summary message for common warning messages. *This is the default.*

`-Verbose`	Displays all warning messages.
`-Both`	Displays warning messages that come from both the Golden and Revised designs and libraries. *This is the default.*
`-Golden`	Displays warning messages that come from the Golden design and library.
`-REvised`	Displays warning messages that come from the Revised design and library.

Related Commands

READ DESIGN

READ LIBRARY

SET FLATTEN MODEL

REPORT MODULE ATTRIBUTE

```
REPort MOdule Attribute
     [-ALL | -PIPELINE_Retime | -COMPARE_Effort | -CPU_Limit |
     -ECO_module | -NOBBOXEMpty | -ISOlate_module | -MODULE <module_name* ...>]
     [-FLAT_ECO_module]
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Displays the module attributes in the Golden and Revised designs. These attributes were originally added with the `ADD MODULE ATTRIBUTE` command.

Tcl Command

`report_module_attribute`

Parameters

`-ALL`	Displays "all" added module attributes within the given defaults.
`-PIPELINE_Retime`	Displays only the modules added for pipeline-retiming.
`-COMPARE_Effort`	Displays only the modules that have specified compare effort levels.
`-CPU_Limit`	Displays the modules with a specified CPU time limit.
`-ECO_module`	Displays the ECO module(s) specified by the `ADD MODULE ATTRIBUTE -ECO_module` attribute. Any mismatched ports for these modules are ignored when running the `WRITE HIER_COMPARE DOFILE` command.
`-NOBBOXEMpty`	Displays the module(s) specified by the `ADD MODULE ATTRIBUTE -NOBBOXEMpty` attribute.
`-ISOlate_module`	Displays the modules marked for isolation (through the `ADD MODULE ATTRIBUTE -isolate_module` option).
`-MODULE <module_name* ...>`	Reports the attribute for the specified module. Wildcards are supported.
`-FLAT_ECO_module`	Displays the ECO module(s) specified by the `ADD MODULE ATTRIBUTE -FLAT_ECO_module` attribute.

`-Both`	Displays the module attributes for both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the module attributes for the Golden design.
`-Revised`	Displays the module attributes for the Revised design.

Related Commands

ADD MODULE ATTRIBUTE

ANALYZE HIER_COMPARE

DELETE MODULE ATTRIBUTE

READ DESIGN

READ LIBRARY

WRITE HIER_COMPARE DOFILE

REPORT MODULES

```
REPort MOdules
     [-ROot | <module_name> [-Down | -Up] | -All | -Top]
     [-INSTantiation [-PARAMinfo | -CONST]]
     [-LEVEL <value>]
     [-Library]
     [-NOINTERLeave |-INTERLeave]
     [-Source]
     [-USer]
     [-VHDLname]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays module information for the Golden and Revised designs. If you specify a module, Conformal displays additional information on modules and library cells up or down the hierarchy of the given module name.

Tcl Command

```
report_modules
```

Parameters

`-ROot`	Displays the name of the root module.
`<module_name>`	Reports module information on the specified module. An additional option lets you report on modules and library cells either up or down the hierarchy of the specified module name. *The default is to report modules and library cells down the hierarchy of the specified module name.* `-Down`: Reports on modules and library cells down the hierarchy of the specified module name. `-Up`: Reports on modules and library cells up the hierarchy of the specified module name.
`-All`	Displays all the modules within the given defaults. The top root module is denoted by (T).
`-Top`	Displays the top modules.
`-Instantiation`	Displays instances and modules.

`-PARAMinfo`	Display parameter value information for the instance of a Verilog parameterized module or a VHDL generic entity.
`-CONST`	Displays parameter and enum value information for top module and instances.
`-LEVEL <value>`	Shows modules in a hierarchical order up to the specified level.
`-Library`	Displays all of the library cells that are in the module hierarchy.
`-NOINTERLeave`	Reports the Golden and Revised module hierarchies separately, first list Golden modules, and then list Revised modules. *This is the default.*
`-INTERLeave`	Reports the Golden and Revised module hierarchies together.
`-Source`	Displays the source-code information identifying where the module is located.
`-USer`	Reports only the modules defined in the design files, and skips the internal modules which are not defined in the design files. **Note:** Some internal modules can be created by the Conformal tools after reading the design files. These internal modules are not defined in the design files.
`-VHDLname`	Displays the full name, rather than just the entity name. For example: `libname.entityname(architecturename)`.
`-Both`	Displays information for both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays module information for the Golden design.
`-Revised`	Displays module information for the Revised design.

Examples

This example shows the difference between running the `REPORT MODULES` command without any options versus running the command with the `-USer` option.

The following design file contains only one module named `test`. Module `VDW_mult_nbw_u8_u8_16` is internal module which is not defined in this file:

```
module test(aa, bb, oo);
input [7:0] aa, bb;
output oo;
  assign oo = aa * bb;
```

```
endmodule
```

Running the following command:

```
report modules
```

the Conformal software reports the `test` and `VDW_mult_nbw_u8_u8_16` modules. However, when running the following command:

```
report modules -user
```

the Conformal software reports only the `test` module.

Related Commands

REPORT MODULE ATTRIBUTE

REPORT PATH

REPORT MOS DIRECTION

```
REPort MOs Direction
     [module_name]
     [-BIdirection | -UNidirection ]
     [-Both | -Golden | -Revised]
     [-Summary | -Verbose]
     (Setup / LEC Mode)
```

Note: This requires a Conformal GXL license.

Displays the unidirectional and bidirectional transistor-MOS instances with their source and drain ports.

Tcl Command

`report_mos_direction`

Parameters

`module_name`	Reports on the named module.
`-BIdirection`	Displays only the bidirectional transistor-MOS instances. *This is the default.*
`-UNidirection`	Displays only the unidirectional transistor-MOS instances.
`-Both`	Displays MOS direction from both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays MOS direction from the Golden design.
`-Revised`	Displays MOS direction from the Revised design.
`-Summary`	Displays a summary message of the total number of unidirectional and bidirectional transistor-MOS. *This is the default.*
`-Verbose`	Displays all of the unidirectional and bidirectional transistor-MOS.

Related Commands

ABSTRACT LOGIC

REPORT MULTIPLIER OPTION

```
REPort MUltiplier Option
```
(*Setup / LEC Mode*)

Displays current multiplier option settings.

Related Commands

ANALYZE DATAPATH

ANALYZE MODULE

REPORT DATAPATH OPTION

SET DATAPATH OPTION

SET MULTIPLIER OPTION

SET FLATTEN MODEL

REPORT NAME ALIAS

```
REPort NAme Alias
     [-GOLden|-REVised]
     [-TYPE <type_name...>]
     [-MODULE <module_name...>]
     [-ALL]
     (LEC Mode)
```

The `ADD NAME ALIAS` command specifies a JSON data file that contains name aliases for changing keypoint names during mapping. These aliases are enabled with the "`set mapping method -alias`" command. The `REPORT NAME ALIAS` command will the status of the name alias, specifically whether the name alias has been applied. If the name alias has not been applied, the report will indicate the reason why it has not been applied.

This command can be used only after mapping with name alias enabled (`set mapping method -alias`).

Tcl Command

`report_name_alias`

Parameters

`-GOLden`	Report the name alias status for the Golden design.
`-REVised`	Report the name alias status for the Revised design.
`-TYPE <type_name>`	Report the name alias for the specified type.
`-MODULE <module_name>`	Report the name alias status for the specified module.
`-ALL`	Report the status for both not-applied and applied name aliases. By default, this command reports only name aliases that have not been applied.

Example

The following is a sample dofile and report from this command.

The following commands add name aliases from the na.json file and report the not-applied name alias data after mapping:

```
add name alias na.json
set mapping method -alias
map key points
report name alias
```

Report:

```
=====================================
File: na.json:1 module: top type: ins
name: abc_reg
alias: xyz_reg
Not applied because it cannot find the module
======================================
```

Related Commands

ADD NAME ALIAS

DELETE NAME ALIAS

SET MAPPING METHOD

REPORT GATE

WRITE MAPPED POINTS

REPORT NET ATTRIBUTE

```
REPort NEt Attribute
     [-ALL | -VDD | -GND | -CLOCK0 | -CLOCK1 | -DYNSTate]
     [-Module <module_name>]
     [-Both | -Golden | -Revised]
```

Note: This requires a Conformal GXL license.

Displays attributes on transistor-MOS nets. The attributes were originally added with the `ADD NET ATTRIBUTE` command.

Tcl Command

`report_net_attribute`

Parameters

`-ALL`	Displays "all" added net attributes within the given defaults.
`-VDD`	Displays only the added VDD net attributes.
`-GND`	Displays only the added GND net attributes.
`-CLOCK0`	Displays only the added Clock-0 net attributes.
`-CLOCK1`	Displays only the added Clock-1 net attributes.
`-DYNSTate`	Displays only the added dynamic state net attributes.
`-Module <module_name>`	Reports net attributes from the named module.
`-Both`	Displays net attributes from both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays net attributes from the Golden design.
`-Revised`	Displays net attributes from the Revised design.

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

READ PATTERN

REPORT CLOCK

REPORT MOS DIRECTION

REPORT PIN DIRECTION

RESOLVE

REPORT NET CONSTRAINTS

```
REPort NEt Constraints
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays all net constraints in the Golden and Revised designs that were added with the `ADD NET CONSTRAINTS` command.

Tcl Command

```
report_net_constraints
```

Parameters

`-Both`	Displays added net constraints in both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the added net constraints in the Golden design.
`-Revised`	Displays the added net constraints in the Revised design.

Related Commands

ADD NET CONSTRAINTS

DELETE NET CONSTRAINTS

REPORT NOBLACK BOX

```
REPort NOblack Box
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays all of the modules in the Golden and Revised designs that will not be included in the hierarchical dofile script generation. These modules were originally specified with the `ADD NOBLACK BOX` command.

Tcl Command

```
report_noblack_box
```

Parameters

`-Both`	Displays added noblackboxes in both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the added noblackboxes in the Golden design.
`-Revised`	Displays the added noblackboxes in the Revised design.

Related Commands

ADD NOBLACK BOX

DELETE NOBLACK BOX

WRITE HIER_COMPARE DOFILE

REPORT NONEQUIVALENT ANALYSIS

```
REPort Nonequivalent Analysis
     < | <gate_id> | <instance_pathname*>... [-Golden | -Revised]]
     [-CATEgory <category>
     [-SUMmary]
     (LEC Mode)
```

Reports the analyze result from the `analyze_nonequivalent -source` command.

Tcl Command

```
report_nonequivalent_analysis
```

Parameters

`<gate_id>`	Report nonequivalent analysis results for the specified gate.
`<instance_pathname*>...`	Report nonequivalent analysis result for the specified gate.
`-Golden`	The specified gate is in the Golden design. This is the default.
`-Revised`	The specified gate is in the Revised design.
`-CATEgory`	Report nonequivalent analysis result for the specified category.
`-SUMmary`	Only reports a summary table of the analysis.

Related Commands

ANALYZE NONEQUIVALENT

REPORT NOTRANSLATE FILEPATHNAMES

```
REPort NOtranslate Filepathnames
     [ | -Library | -Design]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays all of the library and design file pathnames originally added with the `ADD NOTRANSLATE FILEPATHNAMES` command. The Conformal software will not compile these modules defined in libraries and design files.

Tcl Command

```
report_notranslate_filepathnames
```

Parameters

`-Library`	Displays only the added library file pathnames.
`-Design`	Displays only the added design file pathnames.
`-Both`	Displays added file pathnames in both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the added file pathnames in the Golden design.
`-Revised`	Displays the added file pathnames in the Revised design.

Related Commands

ADD NOTRANSLATE FILEPATHNAMES

ADD NOTRANSLATE MODULES

DELETE NOTRANSLATE FILEPATHNAMES

DELETE NOTRANSLATE MODULES

REPORT NOTRANSLATE MODULES

REPORT NOTRANSLATED LINES

```
REPort NOtranslated Lines
     <-Golden | -Revised | -Both>
     (Setup / LEC Mode)
```

Report what has been specified by `add_notranslated_lines` command and the full path of files matched after reading in the design.

The command reports lines that are successfully skipped or lines that are not matched.

An (O) in the report means the line specification is fully matched, an (*) means partially matched, and an (X) means the line specification has not been matched due to a short design file.

Tcl Command

```
report_notranslated_lines
```

Related Commands

ADD NOTRANSLATED LINES

DELETE NOTRANSLATED LINES

REPORT NOTRANSLATE MODULES

```
REPort NOtranslate Modules
```
(*Setup / LEC Mode*)

Displays all of the library and design modules originally added with the `ADD NOTRANSLATE MODULES` command. Conformal will not compile these modules when reading in libraries and designs.

Related Commands

ADD NOTRANSLATE MODULES

DELETE NOTRANSLATE MODULES

READ DESIGN

READ LIBRARY

REPORT OUTPUT EQUIVALENCES

```
REPort OUtput Equivalences
     [-ROot | -Module <module_name> | -All ]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays the output pin equivalences in the Golden and Revised designs. These output pin equivalences were originally added with the `ADD OUTPUT EQUIVALENCES` command.

Tcl Command

```
report_output_equivalences
```

Parameters

`-ROot`	Displays all output pin equivalences from the root module. *This is the default.*
`-Module <module_name>`	Displays the output pin equivalences in the specified module.
`-All`	Displays "all" output pin equivalences in all modules within the given defaults.
`-Both`	Displays the output pin equivalences in both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the output pin equivalences in the Revised design.
`-Revised`	Displays the output pin equivalences in the Golden design.

Related Commands

ADD OUTPUT EQUIVALENCES

DELETE OUTPUT EQUIVALENCES

REPORT OUTPUT STUCK_AT

```
REPort OUtput Stuck_at
     [-ROot | -Module <module_name> | -All ]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays the output `stuck_at` values and pin names in the Golden and Revised designs. These output `stuck_at` values were originally added to pins with the `ADD OUTPUT STUCK_AT` command.

Tcl Command

```
report_output_stuck_at
```

Parameters

`-ROot`	Displays the output `stuck_at` values and pin names from the root module. *This is the default.*
`-Module <module_name>`	Displays the output `stuck_at` values and pin names from the specified module.
`-All`	Displays all output `stuck_at` values and pin names in all modules within the given defaults.
`-Both`	Displays the output `stuck_at` values and pin names in both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the output `stuck_at` values and pin names in the Revised design.
`-Revised`	Displays the output `stuck_at` values and pin names in the Golden design.

Related Commands

ADD OUTPUT STUCK_AT

DELETE OUTPUT STUCK_AT

REPORT PARTITION KEY_POINT

```
REPort PArtition Key_point
```

(*Setup / LEC Mode*)

Displays the partition key points originally added with the `ADD PARTITION KEY_POINT` command.

Related Commands

ADD PARTITION KEY_POINT

DELETE PARTITION KEY_POINT

WRITE PARTITION DOFILE

REPORT PARTITION POINTS

```
REPort PArtition Points
     [-Both | -Golden | -Revised]
     [-Verbose | -Summary]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Displays the partition points that were created with the `ADD PARTITION POINTS` command.

Tcl Command

```
report_partition_points
```

Parameters

`-Both`	Lists the partition points in both the Golden and Revised designs. *This is the default.*
`-Golden`	Lists the partition points in the Golden design.
`-Revised`	Lists the partition points in the Revised design.
`-Verbose`	Displays all added partition points. *This is the default.*
`-Summary`	Displays a summary message of the total number of added partition points.

Related Commands

ADD PARTITION POINTS

DELETE PARTITION POINTS

REPORT PARTITION RESULT

```
REPort PArtition Result
```
(*Setup Mode*)

Displays the results after running partition dofile.

Related Commands

ADD PARTITION KEY_POINT

DELETE PARTITION KEY_POINT

WRITE PARTITION DOFILE

REPORT PATH

```
REPort PAth
     <<source> <destination> | -Feedback | -SELF | -SELF <gate>>
     [-NET]
     [-SEQ_ASYNC]
     [-Source]
     [-Golden | -Revised]
     (Setup / LEC Mode)
```

Displays the paths between a source gate and a destination gate. The `-feedback` option displays all feedback paths for all CUT gates. The source and destination gates can be gate ID numbers, instance paths, or pin paths.

To report the feedback path on one CUT gate, use the same CUT gate ID, instance path, or pin path for both the source and the destination.

Tcl Command

`report_path`

Parameters

`<source>`	Specifies the gate ID number, instance path, or pin path of the source gate. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`<destination>`	Specifies the gate ID number, instance path, or pin path of the destination gate. **Note:** ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.
`-Feedback`	Reports the feedback path of all CUT gates, within the given defaults.
`-SELF <gate>`	Reports all loops to DFF and DLATs. If you specify a gate, it reports only loops to that gate.
`-NET`	Displays the corresponding net of the gate in the path.

`-SEQ_ASYNC`	Reports DFF/DLAT to DFF/DLAT paths passing through the asynchronous set or reset of a sequential element.
`-Source`	Displays the file and line number location of the gate in the path.
`-Golden`	Reports the specified path in the Golden design. *This is the default.*
`-Revised`	Reports the specified path in the Revised design.

Related Commands

ADD CUT POINT

DELETE CUT POINT

REPORT CUT POINT

REPORT GATE

REPORT PHYSICAL CELLS

```
REPort PHysical Cells
     [-ALL | <module* ...>]
     [-BOTh | -GOLden | -REVised ]
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Reports physical cells in the `physical_cell_list`, which is created by the `EXTRACT PHYSICAL CELL` command.

Tcl Command

```
report_physical_cells
```

Parameters

`-ALL`	Reports all cells from the `physical_cell_list`, which is created by the `EXTRACT PHYSICAL CELL` command. *This is the default*.
`<module_name* ...>`	Reports the specified module(s) from the `physical_cell_list`.
`-BOTh`	Reports module(s) in both the Golden and Revised designs. *This is the default.*
`-GOLden`	Reports cell(s) in the Golden design.
`-REVised`	Reports cell(s) in the Revised design.

Related Commands

ADD PHYSICAL CELLS

DELETE PHYSICAL CELLS

EXTRACT PHYSICAL CELLS

REPORT PIN BINDING

```
REPort PIn Binding
     [-Module <golden_module_name> <revised_module_name>]
     (Setup/LEC Mode)
```

Displays the list of pairs of pin names for pin binding in the mapping process. If you do not enter options, Conformal displays all binding pairs..

Tcl Command

```
report_pin_binding
```

Parameters

`-Module`	Specifies the module pair that the pin binding rules are applied in.

Related Commands

ADD PIN BINDING

DELETE PIN BINDING

REPORT PIN CONSTRAINTS

```
REPort PIn Constraints
     [-CLASS < ALL | USER | LOWPOWER >]
     [-ROot | -Module <module_name> | -All]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays the constraints placed on primary input pins in the Golden and Revised designs. These constraints were originally specified with the `ADD PIN CONSTRAINTS` command.

Tcl Command

```
report_pin_constraints
```

Parameters

`-CLASS`	`ALL`: Report all pin constraints. *This is the default*.
	`USER`: Report only constraints added using the `ADD PIN CONSTRAINT` command.
	`LOWPOWER`: Report only power and ground constraints inferred by power intent.
`-ROot`	Displays the pin constraints from the root module. *This is the default.*
`-Module <module_name>`	
	Displays the pin constraints from the specified module.
`-All`	Displays pin constraints in all modules within the given defaults.
`-Both`	Displays the constrained primary input pins from both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the constrained primary input pins from the Golden design.
`-Revised`	Displays the constrained primary input pins from the Revised design.

Related Commands

ADD PIN CONSTRAINTS

DELETE PIN CONSTRAINTS

REPORT PIN DIRECTION

```
REPort PIn Direction
     [-IO | -IN | -OUT]
     [<module_name>]
     [-Summary | -Verbose]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays the assigned pin directions for each module: pin directions are either assigned by the tool or assigned using the `ASSIGN PIN DIRECTION` command for I/O pins. *The default is to display only a summary message.*

Tcl Command

```
report_pin_direction
```

Parameters

`-IO`	Reports assigned module I/O pins. *This is the default.*
`-IN`	Reports assigned module input pins.
`-OUT`	Reports assigned module output pins.
`<module_name>`	Reports pin direction for the specified module. *The default is to report pin direction for all modules.*
`-Summary`	Displays only a summary message of assigned pin directions. *This is the default.*
`-Verbose`	Displays all assigned pin directions.
`-Both`	Reports the assigned pin directions in both the Golden and Revised designs. *This is the default.*
`-Golden`	Reports the assigned pin directions in the Golden design.
`-Revised`	Reports the assigned pin directions in the Revised design.

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

READ PATTERN

REPORT CLOCK

REPORT MOS DIRECTION

REPORT NET ATTRIBUTE

RESOLVE

REPORT PIN EQUIVALENCES

```
REPort PIn Equivalences
    [-ROot | -Module <name> | -All]
    [-Both | -Golden | -REvised]
    (Setup / LEC Mode)
```

Displays a list of added pin equivalences and inverted pin equivalences. These pin equivalences were originally added with the `ADD PIN EQUIVALENCE` command. Inverted pin equivalences are distinguished by a "-" next to the primary input pin name.

Tcl Command

```
report_pin_equivalences
```

Parameters

`-ROot`	Displays pin equivalences from the root module. *This is the default.*
`-Module <module_name>`	Displays pin equivalences from the specified module.
`-All`	Displays pin equivalences in all modules within the given defaults.
`-Both`	Displays pin equivalences from both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays pin equivalences from the Golden design.
`-REvised`	Displays pin equivalences from the Revised design.

Related Commands

ADD PIN EQUIVALENCES

DELETE PIN EQUIVALENCES

REPORT POWER CONSISTENCY

```
REPort POwer Consistency
     (1801 flow)
     [<instance_name | pin_name> [-Golden | -Revised]]
     [-CLASS <EQuivalent | NONEQuivalent| NOTcompared>]
     [-Type <PI | PO | DFF | DLAT | BBOX>]
     [-SUMmary | -Verbose]
     (CPF flow)
     [-STatus <All | Pass | Fail | Notcheck>]
     (LEC mode)
```

Note: This is a Conformal Low Power command and is for the 1801 flow. For CPF flow, this command is the same as "report lowpower data".

Reports the result of consistency checks performed by the `COMPARE POWER CONSISTENCY` command.

Tcl Command

`report_power_consistency`

Parameters

`instance_name`	Displays the specified instance of compared points.
`pin_name`	Displays the specified PI, PO, or the blackbox compared pins.
`-Golden`	Specifies that the specifed instance or pin is from the Golden design. *This is the default*.
`-Revised`	Specifies that the specifed instance or pin is from the Revised design.
`-CLASS`	Displays the specified class of compared points.
`-TYPE`	Displays the specified type of compared points.
	PI:Primary input
	PO: Primary output

	DFF: D flip-flop
	DLAT: D-latch
	BBOX: Blackbox data pins
`-SUMmary`	Lists a summary report of the power consistency comparison result. *This is the default*.
`-Verbose`	Displays a detailed report for each mapped key point and the corresponding supply set information and the power consistency result.

(CPF flow)

`-STatus <All | Pass | Fail | Notcheck>`

Displays the specified status of compared points.

`All` *is the default.*

Related Commands

REPORT POWER INTENT

REPORT POWER_INTENT_COMPARE FILTER

```
REPort POwer_intent_compare Filter
     [<filter_name*> ... ]
     (Setup/LEC Mode)
```

Note: This is a Conformal Low Power command.

Reports the power intent compare filters for the current session. Power intent compare filters are added using the `ADD POWER_INTENT_COMPARE FILTER` command.

Tcl Command

```
report_power_intent_compare_filter
```

Parameters

`<filter_name* ...>`	Specifies the name of the filter(s) to report. By default, the command reports all power intent compare filters. Wildcards are supported.

Example

- The following command reports all power intent compare filters:

```
report power_intent_compare filter
```

- The following command reports all power intent filters which name match "`iso_filter*`":

```
report power_intent_compare filter iso_filter*
```

Related Commands

ADD POWER_INTENT_COMPARE FILTER

COMPARE POWER INTENT

DELETE POWER_INTENT_COMPARE FILTER

REPORT COMPARED INTENT

REPORT POWER INTENT

```
REPort POwer Intent
     [<-type_name ...>]
     [<name* ...>]
     [-DUPLICATED]
     [-FILES]
     [-HIERarchical]
     [-IMPLICIT]
     [-INVALID]
     [-LIBrary]
     [-ONLY_IMPLICIT]
     [-ONLY_EXPLICIT]
     [-REFS]
     [-VERBOSE]
     [-GOLden | -REVised]
     (Setup Mode)
```

Note: This is a Conformal Low Power command.

Reports information about power intent objects and their attributes as well as design references stored in the power intent database.

Tcl Command

```
report_power_intent
```

Parameters

`<-type_name...>`	Selects the power intent objects to report by type. For a list of supported options, see the following section "Power Intent Object Options".
`<name* ...>`	Selects the power intent objects to report by pattern.
`-DUPLICATED`	Report duplicated library definitions.
`-FILES`	With this option, `report power intent` will report all power intent files read during `read power intent`.
`-HIERarchical`	Report only hierarchical (non-elaborated) power intent objects.
`-IMPLICIT`	Report implicit power intent objects. Unless this option is specified, only explicit power intent objects will be reported.
`-INVALID`	Report invalid power intent objects.

`-LIBrary`	Report only library power intent.
`-ONLY_IMPLICIT`	Report only implicit power intent objects. Implicit objects are those that are inferred by the power intent and created implicitly by the tool.
`-ONLY_EXPLICIT`	Report only explicit power intent objects. Explicit objects are those that are created by commands in the power intent.
`-VERBOSE`	Specifies verbose mode for reporting.
`-GOLden`	Reports objects in the golden design.
`-REVised`	Reports objects in the revised design.

Power Intent Object Types

The following tables list the supported power intent object types. These are also supported for the Tcl `find` command for Conformal Low Power.

CPF Flow: Power Intent Model Object Types

`bind_verification_component`	or `verification_component` Reports `bind_verification_component` commands.
`create_analysis_view`	or `analysis_view` Report analysis views defined by `create_analysis_view` commands.
`create_assertion_control`	or `assertion_control` Reports assertion controls defined by `create_assertion_control` commands.
`create_bias_net`	or `bias_net` Reports bias nets defined by `create_bias_net` commands.
`create_global_connection`	or `global_connection` Reports create_global_connection commands.

`create_ground_nets`	or `ground_net` Reports ground nets by `create_ground_nets` commands.
`create_isolation_rule`	or `isolation_rule` Reports isolation rules defined by `create_isolation_rule` commands.
`create_level_shifter_rule`	or `level_shifter_rule` Reports level shifter rules defined by `create_level_shifter_rule` commands.
`create_mode`	or `mode` Reports modes defined by `create_mode` commands.
`create_mode_transition`	or `mode_transition` Reports mode transitions defined by `create_mode_transition` commands.
`create_nominal_condition`	or `nominal_condition` Reports nominal conditions defined by `create_nominal_condition` commands.
`create_operating_corner`	or `operating_corner` Reports operating corners defined by `create_operating_corner` commands.
`create_pad_rule`	or `pad_rule` Reports pad rules defined by `create_pad_rule` commands.
`create_power_domain`	or `power_domain` Reports power domains defined by `create_power_domain` commands.
`create_power_mode`	or `create_power_mode` Reports power modes defined by `create_power_mode` commands.

`create_power_nets`	or `power_net` Reports power nets defined by `create_power_nets` commands.
`create_power_switch_rule`	or `power_switch_rule` Reports power switch rules defined by `create_power_switch_rule` commands.
`create_state_retention_rule`	or `retention_rule` Reports retention rules defined by `create_state_retention_rule` commands.
`define_always_on_cell`	or `always_on_cell` Reports always on cells defined by `define_always_on_cell` commands or extracted from Liberty.
`define_global_cell`	or `global_cell` Reports global cells defined by `define_global_cell` commands or extracted from Liberty.
`define_isolation_cell`	or `isolation_cell` Reports isolation cells defined by `define_isolation_cell` commands or extracted from Liberty.
`define_level_shifter_cell`	or `level_shifter_cell` Reports level shifter cell cells defined by `define_level_shifter_cell` commands or extracted from Liberty.
`define_library_set`	or `library_set` Reports library sets defined by `define_library_set` commands.
`define_open_source_input_pin`	or `open_source_input_pin` Reports cells that contain open source input pins defined by `define_open_source_input_pin` commands.

`define_pad_cell`	or `pad_cell` Reports pad cells defined by `define_pad_cell` commands or extracted from Liberty.
`define_power_clamp_cell`	or `power_clamp_cell` Reports diode cells defined by `define_power_clamp_cell` commands or extracted from Liberty
`define_power_clamp_pins`	or `power_clamp_pins` Reports clamp cells defined by `define_power_clamp_pins` commands or extracted from Liberty
`define_power_switch_cell`	or `power_switch_cell` Reports power switch cells defined by `define_power_switch_cell` commands or extracted from Liberty
`define_related_power_pins`	or `related_power_pins` Reports `define_related_power_pins` commands
`define_state_retention_cell`	or `retention_cell` Reports state retention cells defined by `define_state_retention_cell` commands or extracted from Liberty
`identify_always_on_driver`	Reports `identify_always_on_driver` commands.
`identify_power_logic`	Reports `identify_power_logic` commands.
`identify_secondary_domain`	Reports identify_secondary_domain commands.
`io_pad`	Reports IO pad cells defined by power intent commands.
`set_analog_ports`	or `analog_ports` Reports analog ports defined by `set_analog_ports` commands.

`set_design`	or `design` Reports design models defined by `set_design` commands.
`set_diode_ports`	or `diode_ports` Reports diode ports defined by `set_diode_ports` commands.
`set_equivalent_control_pins`	or `equivalent_control_pins` Reports `set_equivalent_control_pins` commands.
`set_floating_ports`	or `floating_ports` Reports floating ports defined by `set_floating_ports` commands.
`set_input_voltage_tolerance`	or `bias_group` Reports input supply voltage tolerance constraints defined by `set_input_voltage_tolerance` commands.
`set_instance`	or `design_scope` Reports design scopes defined by set_instance commands.
`set_macro_model`	or `macro_model` Reports macro models defined by `set_macro_model` commands or extracted from Liberty
`set_pad_ports`	or `pad_ports` Reports pad ports defined by set_pad_ports commands.
`set_power_source_reference_pin`	or `power_source_reference_pin` Reports `set_power_source_reference_pin` commands.
`set_power_target`	or `power_target` Reports `set_power_target` commands.

`set_sim_control`	or `sim_control` Reports `set_sim_control` commands.
`set_switching_activity`	or `switching_activity` Report `set_switching_activity` commands.
`set_wire_feedthrough_ports`	or `feedthrough_group` Report wire feedthrough ports defined by `set_wire_feedthrough_ports` commands

1801 Flow: Power Intent Model Object Types

`add_port_state`	or `port_state` Reports port states defined by `add_port_state` commands.
`add_power_state`	or `power_state_set` Reports `add_power_state` commands.
`add_pst_state`	or `pst_state` Reports pst states defined by `add_pst_state` commands.
`apply_power_model`	Reports `apply_power_model` commands.
`associate_supply_set`	or `supply_set_handle` Reports supply set handles defined by `create_power_domain` or `associate_supply_set` commands.
`begin_power_model`	or `power_model` Reports power models defined by `begin_power_model` commands.
`bind_checker`	Report `bind_checker` commands.

`connect_logic_net`	Reports `connect_logic_net` commands.
`connect_supply_net`	Reports `connect_supply_net` commands.
`connect_supply_set`	Reports `connect_supply_set` commands.
`create_assertion_control`	Reports assertion controls defined `create_assertion_control` commands.
`create_composite_domain`	or `composite_domain`
	Reports composite domains defined by `create_composite_domain` commands.
`create_hdl2upf_vct`	or `hdl2upf_vct`
	Reports hdl2upf VCD defined by `create_hdl2upf_vct` commands.
`create_logic_net`	or `logic_net`
	Reports logic nets defined by `create_logic_net` commands.
`create_logic_port`	or `logic_port`
	Reports logic ports defined by `create_logic_port` commands.
`create_power_domain`	or `power_domain`
	Reports power domains defined by `create_power_domain` commands.
`create_power_switch`	or `power_switch`
	Reports power switches defined by `create_power_switch` commands.
`create_pst`	or `pst`
	Reports power state tables defined by `create_pst` commands.

`create_supply_net`	or `supply_net` Reports supply net defined by `create_supply_net` commands.
`create_supply_port`	or `supply_port` Reports supply ports defined by `create_supply_port` commands.
`create_supply_set`	or `supply_set` Reports supply sets defined by `create_supply_set` commands.
`create_upf2hdl_vct`	`or upf2hdl_vct` Reports upf2hdl VCT defined by `create_upf2hdl_vct` commands.
`define_always_on_cell`	or `always_on_cell` Reports always on cell defined by `define_always_on_cell` commands or extracted from Liberty
`define_diode_clamp`	or `power_clamp_cell` Reports diode cells or cell pins defined by `define_diode_clamp` commands or extracted from Liberty.
`define_isolation_cell`	or `isolation_cell` Reports isolation cells defined by `define_isolation_cell` commands or extracted from Liberty.
`define_level_shifter_cell`	or `level_shifter_cell` Reports level shifter cells defined by `define_level_shifter_cell` commands or extracted from Liberty.

`define_power_switch_cell`	or `power_switch_cell` Reports power switch or ground switch cells defined by `define_power_switch_cell` commands or extracted from Liberty.
`define_retention_cell`	or `retention_cell` Reports state retention cells defined by `define_retention_cell` commands or extracted from Liberty.
`describe_state_transition`	or `state_transition` Reports state transitions defined by `describe_state_transition` commands.
`map_isolation_cell`	Reports `map_isolation_cell` commands
`map_level_shifter_cell`	Reports `map_level_shifter_cell` commands
`map_power_switch`	Reports `map_power_switch` commands.
`map_retention_cell`	Reports `map_retention_cell` commands.
`name_format`	or `settings` Reports 1801 settings defined by `name_format` commands.
`power_state`	Reports power states defined by add_power_state commands.
`set_design_attributes`	`or design_attributes` Reports `set_design_attributes` commands.

`set_equivalent`	or `equivalent` Reports `set_equivalent` commands.
`set_isolation`	or `isolation` Reports isolation strategies defined by `set_isolation` commands.
`set_level_shifter`	`or set_level_shifter` Reports level shifter strategies defined by `set_level_shifter` commands.
`set_pin_related_supply`	`or pin_related_supply` Reports pin related supplies defined by `set_pin_related_supply` commands.
`set_port_attributes`	or `port_attributes` Reports `set_port_attributes` commands.
`set_related_supply_net`	or `related_supply_net` Reports `set_related_supply_net` commands.
`set_repeater`	or `repeater` Reports repeater (buffer) strategies defined by `set_repeater` commands.
`set_retention_elements`	or `retention_elements` Reports named list of elements defined by `set_retention_elements` commands.
`set_retention`	or `retention` Reports retention strategies defined by `set_retention` commands.

`set_scope`	or `scope` Reports scopes defined by `set_scope` commands.
`set_sim_control`	Reports simulation controls defined by `set_sim_control` commands.
`set_simstate_behavior`	or `simstate_behavior` Reports `set_simstate_behavior` commands.
`supply_net_handle`	Reports supply net handles
`use_interface_cell`	or `interface_cell` Reports `use_interface_cell` commands.

Example

The following command reports all supply set power intent objects specified in the power intent:

```
report power intent –supply_set
```

The following command reports all supply set power intent objects specified in the power intent as well as the implicit supply sets created by the tool:

```
report power intent –supply_set -implicit
```

The following command reports power intent objects whose name matches pattern 'PD1':

```
report power intent PD1
```

Related Commands

READ POWER INTENT

REPORT CROSSING PATH

REPORT LOWPOWER INFO

REPORT PRIMARY INPUTS

```
REPort PRimary Inputs
     [-Class <Full | System | User>]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays primary input pins from the Golden and Revised designs.

Tcl Command

`report_primary_inputs`

Parameters

`-Class`	Displays the following class of primary inputs.	
	`Full`	Primary inputs from both the *User* and *System* classes. *This is the default.*
	`System`	Primary inputs from the original design
	`User`	Primary inputs added with the `ADD PRIMARY INPUT` command
`-Both`	Displays both the Golden and Revised primary inputs. *This is the default.*	
`-Golden`	Displays the Golden design primary inputs.	
`-Revised`	Displays the Revised design primary inputs.	

Related Commands

ADD PRIMARY INPUT

DELETE PRIMARY INPUTS

REPORT PRIMARY OUTPUTS

```
REPort PRimary Outputs
     [-Class <Full | User | System>]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays primary output pins from the Golden and Revised designs.

Tcl Command

```
report_primary_outputs
```

Parameters

`-Class`	Displays the following class of primary outputs.	
	`Full`	Primary outputs from both the *User* and *System* classes. *This is the default.*
	`System`	Primary outputs from the original design
	`User`	Primary outputs added with the `ADD PRIMARY OUTPUT` command
`-Both`	Displays both the Golden and Revised primary outputs. *This is the default.*	
`-Golden`	Displays Golden design primary outputs.	
`-Revised`	Displays Revised design primary outputs.	

Related Commands

ADD PRIMARY OUTPUT

DELETE PRIMARY OUTPUTS

REPORT PROJECT

```
REPort PRoject
     [-ALL]
     [-DB_INFO]
     [-Run <integer ...>]
     [-Verbose]
     (Setup / LEC Mode)
```

This command reports the LEC runs that have been tracked in the current LEC project, which is specified using `SET PROJECT NAME` command.

Tcl Command

`report_project`

Parameters

`-ALL`	Reports the information of all LEC runs. When using this option, the verbose mode is off by default.
`-DB_INFO`	Reports the verification database information of the LEC runs.
`-Run <integer ...>`	Reports the details of one or more LEC runs. When using this option, the verbose mode is on by default.
`-Verbose`	Reports the details for the run (including the host name and dofile for each LEC run).

Related Commands

DELETE PROJECT

SET PROJECT NAME

SET PROJECT OPTIONS

REPORT PST

```
REPort PST
     [-COMPACT [-FLATtened_report]]
     [-ID <state_id> ]
     [-LIMit <n>]
     [-STATE_COUNT]
     [-SUPply <name*> [-STATE <state_name*> | -VOLTAGE <voltage_value*>]]
     [-SUPPLY_IN_PST_ONLY]
     [-EXCLUDE_SUPPLY_IN_SUPPLY_SET_STATE_ONLY]
     [-GOLden | -REVised]
     (Setup Mode)
```

Note: This is a Conformal Low Power and an 1801 feature.

Reports a table that lists all of the globally consistent states.

The supply names, identified by the `add_pst_state` or `add_power_state` command, and their reporting indexes of the table is summarized at the beginning of the report. When multiple equivalent supplies exist, with different names identified in the PSTs, the same index will be assigned to those supplies, and the one with shortest name will be selected to represent the equivalent supply network and to be reported in the table.

Each row of the table presents the voltage values of the supplies. By default, the report is displayed with the alignment table format unless a compact view (-compact) is specified. When -compact is specified without -flattened_report, multiple global consistent states are grouped if some of their supplies share common voltage values. See the Example section for more details.

Each row is also associated with an ID, which can be used to further report the detailed information of the global state(s) when -id is specified.

If there is no global power state that is consistent with all the power state tables, an empty table will be reported.

Tcl Command

```
report_pst
```

Parameters

`-COMPACT`	Reports the table contents in a compact view. See the Example section for more details.

`-FLATtened_report`	Reports the table contents in a flattened view. Each row of the table will report a global consistent state and the reporting order will follow the supply list if specified.
`-ID <state_id>`	Specifies the ID of a row in the table to report the detailed information of the state.
`-LIMit <n>`	Specifies the maximum number of rows to be displayed in the table. Default is 0, which is used to report all global consistent states.
`-STATE_COUNT`	Specifies to report the number of globally consistent states.
`-STATE <state_name*>`	Specifies the supply states and verifies whether the supply states specified are globally consistent. The number of states must match the number of supplies specified and each state must be a valid supply state of the corresponding supply.
`-SUPply <name*>`	Specifies a list of supplies to be reported in the table. When the compact-group view is specified (with `-compact` and no `-flatten_report` option), only one supply is selected for the report when multiple equivalent supplies exist in the list and the order of supplies to be reported does not follow the same order as the specified supply list. When -supply is not specified, a table containing all supplies in the PSTs will be reported.
`-VOLTAGE <voltage_value*>`	Specifies the supply voltages and verifies whether the supply voltages specified are globally consistent. The number of supply voltages must match the number of supplies specified and each supply voltage must be a valid supply voltage value of the corresponding supply.
`-SUPPLY_IN_PST_ONLY`	Reports only supplies specified by `create_pst` or `add_power_state`. Supplies with `add_port_state` but not specified in any PST are not reported.
`-EXCLUDE_SUPPLY_IN_SUPPLY_SET_STATE_ONLY`	In the reporting, exclude the supplies specified only in power states of supply set object or supply set handle.

`-GOLden`	Report tables on the golden side.
`-REVised`	Report tables for the revised side.

Example

Example 1

The following reports all globally consistent states in the design:

```
SETUP>report pst
Column IDs of supply signals:
VDD : 1
VDD_AON : 2
VSS : 3
VDDSW : 4
i0/VDDSW : 5
Table of global consistent states:
------------------------------------------------
     ID
           | VDD
           |       | VDD_AON
           |       |       | VSS
           |       |       |       | VDDSW
           |       |       |       |       | i0/VDDSW
     ------------------------------------------------
     1     | 0.9   | 0.9   | 0     | 0.9   | 0.9
     2     | 0.9   | 0.9   | 0     | 0.9   | OFF
     3     | 0.81  | 0.81  | 0     | 0.81  | 0.81
     4     | 0.81  | 0.81  | 0     | 0.81  | OFF
     5     | 0.81  | 0.81  | 0     | OFF   | 0.81
     6     | 0.81  | 0.81  | 0     | OFF   | OFF
     7     | OFF   | 0.81  | 0     | OFF   | OFF
     8     | OFF   | 0.72  | 0     | OFF   | OFF
```

The following reports the same table with parameter -compact:

```
SETUP> rep pst -compact
Column IDs of supply signals:
VDD : 1
VDD_AON : 2
VSS : 3
VDDSW : 4
i0/VDDSW : 5
Table of global consistent states:
ID, VDD, VDD_AON, VSS, VDDSW, i0/VDDSW
1, 0.9, 0.9, 0, 0.9, {0.9, OFF}
```

```
2, OFF, {[0.81, 0, OFF], [0.72, 0, OFF]}, OFF
3, 0.81, {[0.81, 0, 0.81], [0.81, 0, OFF]}, {0.81, OFF}
```

In the results, if multiple voltage values of a signal are consistent with the voltages of other signals, those voltage values are collected and grouped by {} to be collapsed into one state. For example,

```
1, 0.9, 0.9, 0, 0.9, {0.9, OFF}
```

The resulting consistent states are:

```
0.9, 0.9, 0, 0.9, 0.9
0.9, 0.9, 0, 0.9, OFF
```

If multiple signals are found to have different voltage values but are consistent with the voltages of other signals, the voltage values of those signals are grouped by [ ]. For example, 2, OFF, {[0.81, 0, OFF], [0.72, 0, OFF]}, OFF

The resulting consistent states are:

```
OFF, 0.81, 0, OFF, OFF

OFF, 0.72, 0, OFF, OFF
```

If multiple groups of voltage values exist, each combination of the voltage values of the groups represents a global consistent state. For example,

```
3, 0.81, {[0.81, 0, 0.81], [0.81, 0, OFF]}, {0.81, OFF}
```

The resulting consistent states are:

```
0.81, 0.81, 0, 0.81, 0.81
0.81, 0.81, 0, 0.81, OFF
0.81, 0.81, 0, OFF, 0.81
0.81, 0.81, 0, OFF, OFF
```

Following is the reporting format when both -compact and -flattened_report are specified:

```
ID, VDD, VDD_AON, VSS, VDDSW, i0/VDDSW
1, 0.9, 0.9, 0, 0.9, 0.9
2, 0.9, 0.9, 0, 0.9, OFF
3, 0.81, 0.81, 0, 0.81, 0.81
4, 0.81, 0.81, 0, 0.81, OFF
5, 0.81, 0.81, 0, OFF, 0.81
6, 0.81, 0.81, 0, OFF, OFF
7, OFF, 0.81, 0, OFF, OFF
8, OFF, 0.72, 0, OFF, OFF
```

Example 2

The following reports states for specific supplies: the power of supply set SS_A and the power of supply set SS_B.

```
SETUP> report pst -supply SS_A.power SS_B.power
---------------------
     ID
```

```
| VDDA
|       | VDDB
---------------------
1     | 0.9   | 0.9
2     | OFF   | 0.9
3     | OFF   | OFF
```

Example 3

The following checks if there is a legal (global consistent) state where supply VDD_AON works with voltage 0.81 and VDD_SW is OFF:

```
SETUP> report pst -supply VDD_AON VDD_SW -voltage 0.81 OFF
```

REPORT PULSE GENERATOR

```
REPort PUlse Generator
    [-ALL | MODule <module_name>]
    [-Both | -Golden | -Revised]
    (Setup / LEC Mode)
```

Note: This requires a Conformal GXL license.

Reports the instances that were transformed with the `SET ABSTRACT MODEL -transform_pulse_generator_on` command.

Tcl Command

```
report_pulse_generator
```

Parameters

`-ALL`	Displays all instances. *This is the default.*
`-MODule <module_name>`	Displays a specified module that was transformed.
`-Both`	Applies to both the Golden and Revised designs. *This is the default.*
`-Golden`	Applies to the Golden design.
`-Revised`	Applies to the Revised design.

Related Commands

SET ABSTRACT MODEL

REPORT REMOTE DATA

```
REPort REmote Data
     [-Pid <integer> |-Watch_dir <string>] <report command...>
     (Setup / LEC Mode)
```

Execute the report command of targeted running LEC by process ID and watch directory.

Tcl Command

```
report_remote_data
```

Parameters

`-Pid <integer>`	The process ID of targeted LEC. It needs to be in the same machine.
`-Watch_dir <string>`	The watch directory of targeted LEC. It can cross different machines. The targeted LEC needs to run `set_watcher on -directory <string>` first.
`<report command>`	It supports report_mapped_points, report_unmapped_points, report_compare_data with their corresponding options.

Example

```
report remote data -pid 12345 report_compare_data -class abort
```

Related Command

SET WATCHER

REPORT MAPPED POINTS

REPORT UNMAPPED POINTS

REPORT COMPARE DATA

REPORT REMOVED INSTANCE

```
REPort REMoved Instance
     [-Golden | -Revised]
     (Setup / LEC Mode)
```

Report instances removed with the `REMOVE` command.

Tcl Command

```
report_removed_instance
```

Parameters

`-Golden`	Reports instances removed from the Golden design. *This is the default.*
`-Revised`	Reports instances removed from the Revised design.

Related Command

REMOVE

REPORT RENAMING RULE

```
REPort REnaming Rule
     [ |-MAp | -MOdule | -PIn | -INSTance]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays the list of renaming rules for mapping, module, pin, and instance renaming. These rules were originally added with the `ADD RENAMING RULE` command. The list displays a rule number along with a renaming rule. If you do not enter options, Conformal displays all renaming rules.

Tcl Command

`report_renaming_rule`

Parameters

`-MAp`	Displays only mapping renaming rules. If you do not specify `-map`, `-module`, or `-pin`, Conformal reports all renaming rules.
`-MOdule`	Displays only module renaming rules.
`-PIn`	Displays only pin renaming rules.
`-INSTance`	Displays only instance renaming rules.
`-Both`	Displays the renaming rules applied to both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays the Golden design renaming rules.
`-Revised`	Displays the Revised design renaming rules.

Related Commands

ADD RENAMING RULE

DELETE RENAMING RULE

READ DESIGN

READ LIBRARY

SET MAPPING METHOD

SET NAMING RULE

TEST RENAMING RULE

REPORT RETENTION MAPPING

```
REPort REtention_register Mapping
```
(*Setup / LEC Mode*)

Note: This is a Conformal Low Power command.

Reports the retention mapping rules. The set of rules reported include the user rules added using the `ADD RETENTION MAPPING` command and the default rule added by the system.

For a description of the default rules that are added by the system, see CHECK LOWPOWER CELLS.

Note: The default rule is always reported even if no user rule is added using the `ADD RETENTION MAPPING` command.

Tcl Command

```
report_retention_register_mapping
```

Related Commands

ADD RETENTION MAPPING

DELETE RETENTION MAPPING

REPORT RULE CHECK

```
REPort RUle Check
     [-All | -MODIfied | -RTL | -LP | -1801 | <rule_name*...> |
        NONEQuivalent | ABORT ]
     [-ATTRibutes]
     [-DEFAULT_SEVERITY]
     [ | -Design | -Library]
     [-Error]
     [-File <filename> [<linenumber>] ]
     [-FILTER <filter_expression> [-REGEXP]]
     [-FILTERED_BY <filter_name*>]
     [-HDL_CATegory [-XML <filename>]]
     [-HELP]
     [-Ignore]
     [-LIMit [<natural_number>]]
     [-KEYPoint <<instance_path>|<instance_pin_path>>
        [-REPort <instance|module>] [-INstance_scope <scope_path_name>]]
     [-MODUle <module_name>]
     [-NOHidden | -HIDden | -COMplete | -FILTERED_out | -NOFILTERED_out
        | -WAIved | -NOWaived]
     [-Note]
     [-OCCURRENCE_COUNT | -MAX_PRINT_COUNT <limit>]
     [-RULE_SEParation <num>]
     [-SETTING] ]
     [-STATUS < ALL | FAIL | PASS | IGNORED | NOT-APPLICABLE>]
     [-SUmmary |-Verbose | -OVERVIEW]
     [-Warning]
     [-Both | -Golden | -REvised]
     (Setup / LEC Mode)
```

Displays the list of rule violations after the designs and libraries have been read in. Use the `-summary` option to display all of the violated rules.

Use the `SET RULE HANDLING` command to change the handling of any of these reported rule violations.

See the *Conformal Equivalence Checking User Guide* for rule definitions and sample cases.

Rules with a severity of "Ignore" are not reported except with the `rule_name` or `-ignore` options.

Tcl Command

```
report_rule_check
```

Parameters

`-All`	Displays all predefined rule messages. *This option is the default.*
`-MODIfied`	Reports all the rule check violations that have a different severity level than the original default.
`-RTL`	Displays only RTL rule messages.
`-LP`	Displays power intent quality checks for the library or design, and all structural low power checks for the implemented logical/ physical design.
`-1801`	Displays 1801 rule checks.
`<rule_name* ...>`	Displays only the specified predefined rule messages. This accepts wildcards.
`NONEQuivalent`	This option is intended to be used with the `-KEYPoint` option. Display only RTL rules that potentially result in non-equivalent points in compare.
`ABORT`	This option is intended to be used with the `-KEYPoint` option. Display only RTL rules that potentially result in abort points in compare.
`-ATTRibutes`	Reports attributes of occurrences.
`-DEFAULT_SEVERITY`	Include the default rule severity. This option can only be used with `-OVERVIEW` option.
`-Design`	Reports the design rule violations. If you do not specify `-design` or `-library`, the Conformal software reports rule violations from *both* designs and libraries.
`-Error`	Displays violations that have a severity level of error.

`-File <filename> [<linenumber>]`

Reports all rule check messages in a file. With the `<linenumber>` option, you can report all rule check messages for a specific line number.

`-FILTER <filter_expression>`

	Report only occurrences for which the specified `filter_expression` evaluates to true. The syntax for `filter_expression` is the same as for the Tcl `find` command.
	This is a Conformal Low Power command option.

`-FILTERED_BY <filter_name*>`

	Reports only the occurrences filtered by the specified filter(s). Wildcards are accepted.
`-HDL_CATegory`	Reports RTL rules that fall under "Design intent mismatch" and Synthesis/simulation mismatch".
`-XML <`*`filename`*`>`	Writes the results of `-CATegory` to an XML file. The report is also written to the project directory (if set) and can be viewed through the web interface.
`-HELP`	Displays the rule name, default severity level, and message.
`-Ignore`	Displays violations that have a severity level of ignore.

```
-KEYPoint <<instance_path>|<instance_pin_path>>
  [-REPort <instance|module>][-INstance_scope <scope_path_name>]
```

Report messages associated with a particular keypoint:

`<<instance_path>|<instance_pin_path>>`:
Specify the keypoint. If the instance path is given, report all messages occurring inside the instance and its descendants. If the instance pin path is given, report messages occurring in the fanin cone of the pin.

`-REPort  <instance|module>`:
Report message occurrences on an instance basis or on a module basis.

`-INstance_scope <scope_path_name>`:
Sets the boundary under which to collect and report the rule check messages. By default, the instance scope is the current root module.

The `-KEYpoint` option works in conjunction with the following options of the `Report Rule Check` command:

`- <rulename...>, NONEQuivalent, ABORT`:
RTL rules to be checked. NONEQ and ABORT are special RTL rule names correspond to RTL rules that can potentially cause noneq or abort.

`-SUmmary`:
For each rule, print only the total instance(s) occurrence(s) count

`-Golden | -Revised`:
Report RTL rule(s) for Golden or Revised. The default is Golden.

`-Library` Reports the library rule violations.

If you do not specify `-design` or `-library`, the Conformal software reports rule violations from *both* designs and libraries.

`-LIMit [<natural_number>]`

Limits the verbose display of all the reported rules to the specified number of occurrences. Used with `-help`, it reports the rule checking limit set by the `set rule handling -limit` command.

`-Module <module_name>`

Reports the rule checks that are specific to the specified module.

`-NOHidden`	Reports only occurrences that are not hidden. *This is the default*.
`-Note`	Displays violations that have a severity level of note.
`-HIDden`	Reports only occurrences that are hidden.
`-COMplete`	Reports all occurrences regardless whether they are hidden or not
`-FILTERED_out`	Reports only occurrences excluded by filters (regardless whether they are waived or not).
`-NOFILTERED_out`	Reports only occurrences not excluded by filters (regardless whether they are waived or not).
`-WAIved`	Reports only occurrences that are waived (regardless whether they are excluded by filters or not).
`-NOWaived`	Reports only occurrences that are not waived (regardless whether they are excluded by filters or not).
`-OCCURRENCE_COUNT`	Report the number of occurrences even if it is zero.
`-MAX_PRINT_COUNT <limit>`	Limits the verbose display of all the reported rules to the specified number of occurrences. Used with `-help`, it reports the rule checking limit set by the `set rule handling -limit` command.
`-RULE_SEParation <num>`	`<num>` specifies the number of blank lines to print before each rule is reported. Without this option, no blank lines are printed before each reported rule.
`-SETTING`	Displays the current severity level for the rule.
`-STATUS`	Selects rules to be displayed based on their execution status. `ALL`: All rules. *This is the default*. `FAIL`: Rules with failure occurrences. `PASS`: Rules without failure occurrences. `IGNORED`: Rules that are not run because their severity is 'ignore'. `NOT-APPLICABLE`: Rules not applicable for the run.
`-Summary`	Displays the status in summary format. *This is the default.*

`-Verbose`	Displays the status in verbose format.
`-OVERVIEW`	Displays an overview of the rule checker status (includes rule name, severity, applied analysis style, and status).
`-Warning`	Displays violations that have a severity level of warning.
`-Both`	Report for both Golden and Revised. *This is the default*.
`-Golden`	Report for the Golden design only.
`-REvised`	Report for the Revised design only.

Related Commands

READ DESIGN

READ LIBRARY

SET RULE HANDLING

REPORT RULE FILTER

```
REPort RUle Filter
     <filtername* ...>
     (Setup / Verify Mode)
```

Reports the currently defined rule filters, in dofile format.

Tcl Command

`report_rule_filter`

Parameters

`<filtername* ...>`	Specifies the name(s) of the filter(s) to report.

Examples

The following command saves the `ABC` filter report to a file named `XYZ`:

```
report rule filter ABC > XYZ
```

Related Commands

ADD RULE FILTER

DELETE RULE FILTER

REPORT RUNTIME LIMIT

```
REPort RUntime Limit
     [-MODule [module_name]]
     [-HIER_compare]
     [-TIMEOUT_MODule]
     [-COMmand [command_name]]
     (Setup Mode)
```

This command reports the runtime for the specified module, `hierarchical_compare`, timeout modules and commands. The unit of time is in seconds.

Tcl Command

`report_runtime_limit`

Parameters

`-MODule`	Reports the runtime limit of LEC in the specified module. If the module name is not specified, the runtime limit of the root module will be reported
`-HIER_compare`	Reports the runtime limit of the hierarchical comparison and the file name of the specified dofile.
`-TIMEOUT_MODule`	Reports the timeout module compare in the hierarchical comparison and the command which reaches the runtime limit.
`-COMmand`	Reports the runtime limit of the specified command. If the command name is not specified, the runtime limit of all the commands will be reported.

Related Commands

DELETE RUNTIME LIMIT

SET RUNTIME LIMIT

REPORT SEARCH PATH

```
REPort SEarch Path
     [ | -Design | -Library | -POWER_intent]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Displays the paths Conformal searches to locate filenames included in the `READ DESIGN` and `READ LIBRARY` commands.

Tcl Command

```
report_search_path
```

Parameters

`-Design`	Reports the search path used by the `READ DESIGN` command. If you do not specify `-design` or `-library`, Conformal reports the search path used by both the `READ DESIGN` command and the `READ LIBRARY` command.
`-Library`	Reports the search path used by the `READ LIBRARY` command. If you do not specify `-design` or `-library`, Conformal reports the search path used by both the `READ DESIGN` command and the `READ LIBRARY` command.
`-POWER_intent`	This is a low power command option. Reports the search path(s) for power intent files.
`-Both`	Reports the search path used by both the Golden and Revised designs. *This is the default.*
`-Golden`	Reports the search path used by the Golden design and library.
`-Revised`	Reports the search path used by the Revised design and library.

Related Commands

ADD SEARCH PATH

DELETE SEARCH PATH

READ DESIGN

READ LIBRARY

REPORT SEQ_CORRESPONDENCE

```
REPort SEQ_CORRespondence
     [-ALL [-GOLden | -REVised]]
     [-GATe <identifier> [-GOLden | -REVised]]
     [-RETime < ALL | SUCCESS | FAIL>]
     [-UNAccounted]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Displays the sequential corresponding points that were automatically identified or added with the `ADD SEQ_CORR` command. Use this command to help debug general retiming by displaying the sequential correspondence and retiming results.

Tcl Command

```
report_seq_correspondence
```

Parameters

`-ALL [-GOLden | -REVised]`

Displays all pairs of the sequential corresponding points.

`-GOLden` displays pairs of the sequential corresponding points where the first elements of the pairs are from the Golden design. *This is the option default*.

`-REVised` displays pairs of the sequential corresponding points where the first elements of the pairs are form the Revised design

`-GATe <identifier> [ -GOLden | -REVised ]`

Displays the gates in the other design that sequentially correspond to the specified gate.

`-GOLden` indicates that the gate specified by `<identifier>` is from the Golden design. *This is the option default*.

`-REVised` indicates that the gate specified by `<identifier>` is from the Revised design.

`-RETime < ALL | SUCCESS | FAIL>`	
	Displays the general retiming results of the sequential corresponding points in the Revised design.
	`ALL` displays all the Revised state points' retiming results.
	`SUCCESS` displays only the state points where retiming is successful.
	`FAIL` displays only the state points where retiming failed.
`-UNAccounted`	Displays the registers in the Golden design that neither can be mapped by name to any register in the Revised design, nor have any sequential corresponding point in the Revised design.

Examples

- The following command reports a list of Golden registers that are neither mapped by name nor have sequential correspondence:

  ```
  report seq_corr -unaccounted
  ```

- The following command reports a list of failed general retiming:

  ```
  report seq_corr -retime fail
  ```

Related Commands

ANALYZE RETIMING

ADD SEQ_CORRESPONDENCE

DELETE SEQ_CORRESPONDENCE

SET RETIMING OPTION

REPORT SETUP INFORMATION

```
REPort SEtup Information
     [-SUMMARY]
     [-USAGE]
     [-GOLden | -REVised]
     [-VERBOSE]
     (Setup/LEC)
```

This command displays information about setup information that has been read in using the `READ SETUP INFORMATION` command.

Tcl Command

```
report_setup_information
```

Parameters

`-SUMMARY`	Reports a summary of the setup information that has already been read in.
`-USAGE`	Reports the usage of the setup information. The usage has three categories: Used, Unused and Rejected. Where: Used: Indicates the information has been proven correct and applied. Unused: Indicates the information is not used. Most likely the corresponding objects cannot be found in the design. Rejected: Indicates the information has been proven incorrect.
`-GOLden`	Reports setup information for the Golden design. By default, the report is for both the Golden and Revised designs.
`-REVised`	Reports setup information for the Revised design. By default, the report is for both the Golden and Revised designs.
`-VERBOSE`	Provides detailed information of what is Used, Unused, or Rejected.

Examples

- To view a summary of the setup information that has been read in, use the following command:

  ```
  SETUP/LEC> report setup information -summary
  ```

  This report displays the number of occurrences with setup information and the source of the setup information.

  ```
  SETUP/LEC> report setup information -summary
  =================================================================
  Occurrences               Source
  -----------------------------------------------------------------
  4 seq_merge
  4 total                   rc.log1
  =================================================================
  ```

- The following displays which occurrences of setup information are used in the current run, not used in the current run, and rejected:

  ```
  SETUP/LEC> report setup info -usage -verbose
  ================================================================================
                  Used    Unused     Rejected
  --------------------------------------------------------------------------------
  seq_merge       2       1          1
  ================================================================================
  ```

Related Commands

READ SETUP INFORMATION

REPORT STATISTICS

`REPort STatistics`
(*LEC Mode*)

Summarizes the mapping and comparison statistics for the Golden and Revised designs in a table.

Tcl Command

```
report_statistics
```

Related Commands

REPORT COMPARE DATA

REPORT COMPARED POINTS

REPORT MAPPED POINTS

REPORT UNMAPPED POINTS

REPORT SUPPLY

```
REPort SUpply
     [-PORT | -GLOBAL]
     [-ROOT| -Module <name_list*> | -ALL]
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Reports the power and ground pins in the design that were defined with the `ADD SUPPLY` command.

Tcl Command

```
report_supply
```

Parameters

`-PORT`	The reported object(s) must be the port(s) at the root or the specified module level. *This is the default.*
`-GLOBAL`	The reported object(s) could be the port(s) and wire(s) in the hierarchy of the root or the specified module.
`-ROOT`	Reports the supply attribute to the specified objects in the current scope and all hierarchy of this scope. *This is the default.*
`-Module <name_list*>`	Reports the attribute setting to the specified module. This accepts wildcards.
`-ALL`	Reports the attribute settings to the objects for all modules.
`-Golden`	Specifies that the listed names are from the Golden design. *This is the default.*
`-Revised`	Specifies that the listed names are from the Revised design.
`-Both`	Specifies that the listed names are from both the Golden and Revised designs.

Related Commands

ADD SUPPLY

DELETE SUPPLY

REPORT TCL ATTRIBUTES

```
REPort TCl Attributes
     [<obj_name* ...>]
     [-1801_OBJ]
     [-CONFORMAL_OBJ]
     [-CPF_OBJ]
     [-LIB_COMPARE_OBJ]
     [-LIB_OBJ]
     [-PIA_OBJ]
     [-PIC_OBJ]
     [-RTL_OBJ]
     [-RULE]
     [-RULE_OBJ]
     [-SDC_OBJ]
     [-XML <filename>]
     (Setup / Verify Mode)
```

Displays the Tcl attributes for the selected object.

Tcl Command

```
report_tcl_attributes
```

Parameters

`<obj_name* ...>`	Reports attributes of the specified objects. This accepts wildcards.
`-1801_OBJ`	Reports attributes of 1801 objects.
`-CONFORMAL_OBJ`	Reports Conformal objects.
`-CPF_OBJ`	Reports attributes of CPF objects.
`-LIB_COMPARE_OBJ`	Reports attributes of Liberty compare objects.
`-LIB_OBJ`	Reports attributes of Liberty objects.
`-PIA_OBJ`	Reports attributes of CPF and 1801 objects.
`-PIC_OBJ`	Reports attributes of power compare intent objects.
`-RTL_OBJ`	Reports attributes of RTL objects.
`-RULE`	Reports attributes of occurrences of rules.

`-RULE_OBJ`	Reports attributes of RULE objects (RuleSet, RuleGroup, RuleSrc, RuleInst, RuleOccr).
`-SDC_OBJ`	Reports attributes of SDC objects.
`-XML <filename>`	Writes attributes out to an XML file which can be open as an Excel spreadsheet.

Example

- The following command reports attributes of all objects:

```
report tcl attributes
```

- The following command reports attributes of all objects to an XML file:

```
report tcl attributes -xml attrs.xml
```

- The following command reports attributes of CPF objects which match 'power*'

```
report tcl attributes -cpf_obj power*
```

- The following command reports attributes of RTL and library objects:

```
report tcl attributes -rtl_obj -lib_obj
```

- The following command reports attributes of 'libpin'

```
report tcl attributes libpin
```

REPORT TESTCASE

```
REPort TEstcase
     < [-NONEQ]
       [-ABORT]
       [-Golden <<gate_id> | <instance_pathname> ...> ]
       [-Revised <<gate_id> | <instance_pathname> ...> ]
       [-DATAPATH_module <-INST_name <instance_name*> ...>
         | <-QUAlity <number>>]
     >
     [-DIR_name <directory_name>]
     [-FIle <filename>]
     [-KEYPOINT_DEPTH <number>]
     [-NAME | -NONAME]
     [-REPlace | -APPend]
     (LEC Mode)
```

Automatically extracts testcases for selected key points and generates a dofile and a file containing mapping information. Running the generated dofile can reproduce the problem in original design, such as nonequivalences and aborts.

The `-datapath_module` option applies the testcase extraction to datapath modules in the resource file. This option is used after datapath module-based analysis. The extracted testcase is encapsulated in the XML file.

Tcl Command

```
report_testcase
```

Parameters

`-NONEQ`	Specifies that all nonequivalent points will be selected for the generated testcase.
`-ABORT`	Specifies that all abort points will be selected for the generated testcase.
`-GOlden`	Applies the testcase extraction to the Golden design.
`-Revised`	Applies the testcase extraction to the Revised design.
`<gate_id>`	Specifies the gate ID of the testcase. You can specify multiple gate IDs.

`<instance_pathname>`	Specifies the instance pathname of the testcase. You can specify multiple instance pathnames.
`-DATAPATH_module`	Applies report testcase on datapath modules. The default is in the Revised design netlist. If only this option specified, datapath modules in the resource file will be listed.
`-INST_name <instance_name*> ...`	Specifies the instance name to report testcase on datapath modules. You can specify multiple instance names. This accepts wildcards.
`-QUAlity <number>`	Specifies a number to report testcase on datapath modules. The testcases whose evaluated quality less than or equal to the specified number will be reported.
`-DIR_name <directory_name>`	Specifies the name of the testcase directory. If not specified, the name of the directory will be `LEC_testcase`.
`-FIle <filename>`	Specifies the filename where the datapath modules will be written to. If not specified, the name of the file will be `datapath_module.xml`.
`-KEYPOINT_DEPTH <number>`	Specifies the depth of the key points to report. Starting from the selected key point, the closest key point in its fanin or fan-out cone is a depth of 1. The default is 3.
`-NAME`	Includes the names of design objects (nets, instances, ports) in the generated testcase. *This is the default.*
`-NONAME`	Do not includes the names of design objects (nets, instances, ports) in the generated testcase.
`-REPlace`	Overwrites the existing testcase directory. *This is the default.*
`-APPend`	Prepends the root module name to the specified filename to avoid overwriting the same file, especially used in hierarchical compare flow.

Examples

- The following command selects all nonequivalent points, all abort points, and key points with gate id 10, with instance `dlat` in the Golden design netlist and the key point with instance `dff` in the Revised design netlist, allowing the names of design objects to be included in the generated testcase:

```
report testcase -noneq -abort -golden 10 dlat -revised dff
```

- The following command will select key point with gate id 10 in Revised design netlist for testcase extraction. The names of design objects will use generic names (gate type and a serial number)

```
report testcase -revised 10 -noname
```

- The following command will select all nonequivalent points for testcase extraction, using key point depth:

```
report testcase -noneq -keypoint_depth 1
```

- The following command will report testcases on the datapath module whose instance name starts with `add` into the file `add.xml` under the directory `LEC_testcase`:

```
report testcase -datapath_module -inst_name add* -file \
  add.xml -dir_name LEC_testcase -replace
```

- The following command will report testcase on the datapath modules whose evaluated quality are less than or equal to 30% into the file `low_quality.xml` under the directory `LEC_testcase`:

```
report testcase -datapath_module -quality 30 -file low_quality.xml \
  -dir_name LEC_testcase -replace
```

Related Command

READ TESTCASE

REPORT TEST VECTOR

```
REPort TEst Vector
     <<gate_id> | <instance_pathname> | <pin_pathname>>
        [<<gate_id> | <instance_pathname> | <pin_pathname>> [-Index <integer>]]
        [-CUbe]
        [-File <file_name>]
        | [-NONEQ]
        [-NCSim]
        [-NUm <integer>]
        [-Golden | -Revised]
     (LEC Mode)
```

Displays error patterns for a specific nonequivalent compared point.

- The first argument is the nonequivalent compared point.
  This point can be identified with a gate identification number, instance path, or a pin path.
- The second argument, which is optional, is the diagnosis input point. These points are gates that connect directly to the input ports of the nonequivalent compared point where the logic cones are different. The display shows the diagnosis input point to the nonequivalent compared point; corresponding and non corresponding support key points with their simulation values; and final simulation result of the diagnosis input point
- The Index option is used to specify which error pattern is displayed after you use the command. If you do not specify an index number, Conformal displays the first error pattern.

You can also use this command with the `-noneq` option to report error patterns for *every* nonequivalent compared key point.

Tcl Command

```
report_test_vector
```

Parameters

`<gate_id> | <instance_pathname> | <pin_pathname>`

Specifies the gate identification number, instance path, or pin path of the nonequivalent compared point.

Note: ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.

`<gate_id> | <instance_pathname> | <pin_pathname>`

Specifies the gate identification number, instance path, or pin path of the *diagnosis input point* to the nonequivalent compared point.

Note: These options apply only to diagnosis points for DFF, DLAT, and BBOX. If you enter a point that is *not* a diagnosis point, Conformal will error out.

Note: ID numbers can differ from one version of Conformal to another. Always use the full path in dofiles and any time you rerun a design with a different Conformal version.

`-Index <integer>`	Displays the specified error pattern. *The default is to display the first error pattern.*
`-CUbe`	Displays Boolean cubes instead of minterms. By default, `REPORT TEST VECTOR` reports minterms, with each support assigned value of `0` or `1`. A *cube* covers a set of minterms, but does not display all supports.
`-File <file_name>`	Outputs the error pattern to the specified file. If the file exists, Conformal overwrites the file; otherwise, Conformal creates a new file with the specified name. The output error pattern is in the JSON file format. Note that this output error pattern can be re-simulated by the command '`report key point -simulation <file_name>`'.
`-NONEQ`	Displays the error pattern for every nonequivalent point. This option is useful for identifying supports that are most likely not related to the nonequivalence. For example, support 3 does not appear in cube `01x`; it covers two minterms: 011 and 010.

`-NCSim`	Enables the output of a report that can directly be consumed by NC-Sim simulator.
`-NUm <integer>`	Displays the specified number of test vectors.
`-Golden`	Specifies that the nonequivalent compared point is from the Golden design. *This is the default.*
`-Revised`	Specifies that the nonequivalent compared point is from the Revised design.

Example

Using the Report Generated by -ncsim

To generate a test vector report that can be consumed by NC-Sim, use the `-ncsim` option to output the test vectors to a Tcl File:

```
report test vector kp1 -ncsim > lecvec.tcl
```

Conformal will create a Tcl script of the test vectors. This file will contain two procedures: `golforce1` and `revforce1`, which you can use to put test vectors in the design. For example (after you load the Golden/Revised design into NC-Sim and source the Tcl file), you can use the following command:

```
source lecvec.tcl
golforce1 <index>
```

where <*index*> is the nth index of the test vector to insert. For example,

```
golforce1 2
```

puts the 2nd test vector onto the Golden design.

revforce1 1

puts the first test vector onto the Revised design.

You can then run the simulation.

Related Command

DIAGNOSE

REPORT TIED SIGNALS

```
REPort TIed Signals
     [-Class <Full | System | User | Lowpower>]
     [-ROot | -Module <module_name>]
     [ | -TIE0 | -TIE1 | -TIEZ | -TIEX]
     [ | -Net | -Pin [-Direction] ]
     [-Both | -Golden | -REvised]
     (Setup / LEC Mode)
```

Displays tied signals from the Golden and Revised designs.

Tcl Command

`report_tied_signals`

Parameters

`-Class`	Displays this class of tied signals:	
	`Full`	Tied signals from both the *User* and *System* classes. *This is the default.*
	`System`	Tied signals from the original design.
	`User`	Tied signals added with the `ADD TIED SIGNALS` command.
	`Lowpower`	Tied signals from the remodeling of Conformal Low Power. Example available in Example section.
`-ROot`	Displays tied signals in the root module. *This is the default.*	
`-Module <module_name>`		
	Displays tied signals in the specified module.	
`-TIE0`	Displays signals tied to logic 0. If you do not specify the logic, Conformal displays signals tied to logic 0, 1, Z, and X.	
`-TIE1`	Displays signals tied to logic 1. If you do not specify the logic, Conformal displays signals tied to logic 0, 1, Z, and X.	
`-TIEZ`	Displays signals tied to logic Z. If you do not specify the logic, Conformal displays signals tied to logic 0, 1, Z, and X.	

`-TIEX`	Displays signals tied to logic X. If you do not specify the logic, Conformal displays signals tied to logic 0, 1, Z, and X.
`-All`	Displays all net and instance names, within the given defaults, that have tied signals assigned to them. *This is the default.*
`-Net`	Displays net names that have tied signals assigned to them.
`-Pin`	Displays pin names that have tied signals assigned to them.
	If `-Direction` is used, Conformal will also display the pin direction information.
`-Both`	Displays tied signals from both the Golden and Revised designs. *This is the default.*
`-Golden`	Displays tied signals from the Golden design.
`-REvised`	Displays tied signals from the Revised design.

Examples

Reporting Tied Signals from the remodeling of Conformal Low Power

```
SETUP> report tied signal -class lowpower
Tied nets in Golden: LOWPOWER class
 In module SRDFFRQX1: RT <TIE0>
 In module SRDFFSRX1: RT <TIE0>
Tied nets in Revised: LOWPOWER class
 In module SRDFFRQX1: RT <TIE0>
 In module SRDFFSRQX1: RT <TIE0>
```

Reporting Tied Signals from Both the User and System Classes

```
SETUP> report tied signal -class full
Tied nets in Golden: SYSTEM class
 In module TIELO: Y <TIE0>
 In module SDFFRHQX1: U$1/S <TIE0>
Tied nets in Golden: USER class
 In module mac_tx_fifo_1: afull <TIE0>
Tied nets in Golden: LOWPOWER class
 In module SRDFFRQX1: RT <TIE0>
 In module SRDFFSRX1: RT <TIE0>
Tied nets in Revised: SYSTEM class
 In module TIELO: Y <TIE0>
 In module SDFFRHQX1: U$1/S <TIE0>
Tied nets in Revised: LOWPOWER class
 In module SRDFFRQX1: RT <TIE0>
 In module SRDFFSRQX1: RT <TIE0>
```

Related Commands

ADD TIED SIGNALS

DELETE TIED SIGNALS

REPORT UNMAPPED POINTS

```
REPort UNmapped Points
     [-SUMmary | -Extra | -UNReachable [-BLocked | -NOFanout] | -NOTmapped
       | [ -TYpe <PI | E | Z | DFf | DLat | CUt | BBox | PO> ...
            | -NOTYpe <PI | E | Z | DFf | DLat | CUt | BBox | PO> ...]
     ]
     [-GRoup]
     [-IGNORE_VERIFIED_PO_UNMAP]
     [-INPUT][-OUTPUT]
     [-LIBName | -NOLIBName]
     [-RETention]
     [-NODLAT_GATED_CLOCK]
     [-NOIGNORED]
     [-USER_ADD]
     [-GOlden | -Revised]
     (LEC Mode)
```

This report lists unmapped points, along with a summary of all of the unmapped points in the Golden and Revised designs.

Note: If you do not specify options, Conformal identifies all unmapped points *and* displays a summary. Furthermore, if you do not specify either Golden or Revised, Conformal reports unmapped points for both designs.

Tcl Command

```
report_unmapped_points
```

Parameters

`-SUMmary`	Lists a summary report of all of the unmapped points in the Golden and Revised designs. *This is the default.*
`-Extra`	Lists extra points. These points are unmapped because they do not map with a counterpart in the comparison design. Extra points do not affect the circuit.
`-UNReachable`	Lists unreachable unmapped points. Unreachable key points are those that do not eventually affect the primary output of the design.

	`-BLocked`	Lists unreachable unmapped points that have a path to primary outputs, but are blocked by other signals.
	`-NOFanout`	Lists unreachable unmapped points that do not have a path to primary outputs.
`-NOTmapped`	Lists "Not-mapped" unmapped key points. Not-mapped key points are those that failed to be mapped.	
`-Type`	Lists unmapped points of the specified type. Available types are as follows: `PI`: Primary input `E`: TIE-E `Z`: TIE-Z `DFf`: D flip-flop `DLat`: D-latch `CUt`: All unmapped points for artificial gates that break combinational loops `BBox`: Blackbox `PO`: Primary output	
`-NOType`	Do not list unmapped points of the specified type.	
`-GRoup`	Displays the unmapped groups in which either the Golden or Revised key point is a *group* of equivalent gates rather than a *single* gate. The group can be defined with the `ADD INSTANCE EQUIVALENCE` command or the `-seq_merge` option of the `SET FLATTEN MODEL` command. A key point group is counted as one key point.	
`-IGNORE_VERIFIED_PO_UNMAP`	Do not list unmapped POs with constraint relations that need to be proven, such as those that involve "`add output equivalence -user`" or "`add output stuck_at -user`".	
`-INPUT`	Lists unmapped inputs of blackbox or sequential elements.	
`-OUTPUT`	Lists unmapped outputs of blackbox or sequential elements.	

`-LIBName`	When displaying unmapped points, includes suffixes. *This is the default.*
`-NOLIBName`	When displaying unmapped points, does not include suffixes.
`-RETention`	**Note:** This is a Conformal Low Power option. If the unmapped point is a sequential element (DFF or DLAT) and belongs to the Golden Design, this option reports the tag-name (if any) associated with the DFF or DLAT. If the unmapped point is a sequential element (DFF or DLAT) and belongs to the Revised Design, this option reports the power gating cell attribute (if any) associated with the DFF or DLAT. For non-sequential elements, nothing is reported. The sequential unmapped points are not written out during the `CHECK RETENTION MAPPING` command, but if present, they are reported as 'Not-Checked' in the summary section.
`-NODLAT_GATED_CLOCK`	Do not report deglitching clock-gating DLATs under the Unreachable category.
`-NOIGNORED`	Do not report the ignored key points from the commands `ADD IGNORED INPUTS` and `ADD IGNORED OUTPUTS`.
`-USER_ADD`	Lists only primary inputs or outputs added using commands `ADD PRIMARY INPUT` or `ADD PRIMARY OUTPUT`.
`-GOlden`	Lists only the Golden unmapped points.
`-Revised`	Lists only the Revised unmapped points.

Related Commands

ADD MAPPED POINTS

DELETE MAPPED POINTS

MAP KEY POINTS

REPORT MAPPED POINTS

REPORT STATISTICS

SET MAPPING METHOD

SET NAMING RULE

REPORT VERIFICATION

```
REPort VErification
     [-COMPare_result]
     [-HIER]
     [-IGNORE_VERIFIED_PO_UNMAP]
     [-Summary]
     [-Verbose]
     (LEC/SETUP Mode)
```

Reports a table of all violated checklist items for the categories listed below.

- Non-standard modeling options used:
  - Tristated output: `checked | not checked`
  - Revised X signals set to E: `yes | no`
  - Floating signals tied to Z: `yes | no`
  - Command `add clock` for clock-gating used: `yes | no`
- Incomplete verification:
  - All primary outputs are mapped: `yes | no`
  - All mapped points added as compare points: `yes | no`
  - All compare points compared: `yes | no`
  - User added black box: `yes | no`
  - Black box mapped with different module name: `yes | no`
  - Command `add ignore outputs` used: `yes | no`
- Modification to design:
  - Change gate type: `yes | no`
  - Change wire: `yes | no`
  - Primary inputs added: `yes | no`
- Conformal Constraint Designer clock domain crossing checks recommended:
  - Multiple clocks in the design: `yes | no`
- Design ambiguity:
  - Duplicate module definition: `yes | no`

- ❑ Black box due to undefined cells: `yes | no`

- Compare results: `FAIL:INCOMPLETE | FAIL:NONEQ | ABORT | PASS`

  Compare data is examined for the following:

  1. All primary outputs and reachable blackbox inputs have been mapped
  2. All mapped points have been added as compared points
  3. All compared points have been compared
  4. No compared points result in nonequivalent.
  5. No compared points result in an abort

  If all statements are true, the compare result is `PASS`.

  If statements 1, 2, 3, and 4 are true, then the compare result is `ABORT`.

  If criterion 4 is false, the compare result is `FAIL:NONEQ`.

  For all other scenarios, the compare result is `FAIL:INCOMPLETE`.

Tcl Command

```
report_verification
```

Parameters

`-COMPare_result`	Prints out one line report on compare results.
`-HIER`	Prints out the hierarchical comparison result. When used for hierarchical comparisons, this command reports the same results as `REPORT HIER_COMPARE RESULT` unless the current module of the hierarchical compare is completely flattened. In that case, `REPORT VERIFICATION` will report the flattened comparison result.
`-IGNORE_VERIFIED_PO_UNMAP`	When enabled, unmapped verified pin equivalents and stuck-at primary outputs will be included in the report but will not be considered as violations that can cause an "`INCOMPLETE`" verification.

`-Summary`	Prints all items for each category and the violated items are marked with an asterisk (`*`).
`-Verbose`	Prints out each category and the count of violations.

Example

If your report includes the following line:

```
Always-false constraints detected
```

This specifies that there are some constraints that are always violated, and this may lead to incomplete verification. For example, in the following VHDL code, the range constraint is always violated.

```
signal a: std_logic_vector(1 downto 0);
signal a_num: integer range 4 to 9;
a_num <= conv_integer(a);
```

Related Command

COMPARE

REPORT VERIFICATION INFORMATION

```
REPort VErification Information
     [-ADVANCED_REPORT <USAge | DATapath | ABOrt | NONeq>]
     [-SECOND_RUN_STATUS]
     (Setup/LEC Mode)
```

Provides a report regarding verification information. Currently, this command only supports the reporting of information described in the web interface article titled Conformal Automatic Reporting in the LEC Web Interface.

This command can be used only when SET VERIFICATION INFORMATION is enabled.

Tcl Command

```
report_verification_information
```

Parameters

`-ADVANCED_REPORT`	Provides a text report of the information available from the web interface. For example, specifying `USAge` would display the data shown in the `USAge` tab of the web interface.
`-SECOND_RUN_STATUS`	Provides a table to report the status of each second run information category in verification information, include the existence in previous run, existence in current run, permission to apply and permission to record.

Related Command

SET VERIFICATION INFORMATION

RESET

```
RESET
```
(*Setup / LEC Mode*)

Resets the system to the initial state. All existing designs and libraries are deleted, and all previously issued commands are cancelled.

Tcl Command

```
reset
```

Related Commands

RESET HIER_COMPARE RESULT

EXIT

RESET ABSTRACT MODEL

```
RESet ABSTract Model
     [-ALL | -MODule <module_name>]
     [-Both | -Golden | -Revised]
     (Setup / LEC Mode)
```

Note: This requires a Conformal GXL license.

Resets the abstraction conditions that you set using the `SET ABSTRACT MODEL` command.

Tcl Command

```
reset_abstract_model
```

Parameters

`-All`	Resets abstraction conditions for all modules.
`-MODule <module_name>`	Resets abstraction conditions for the specified modules.
`-Both`	Resets abstraction conditions for both the Golden and Revised designs. *This is the default.*
`-Golden`	Resets abstraction conditions for the Golden design.
`-Revised`	Resets abstraction conditions for the Revised design.

Related Commands

ABSTRACT LOGIC

REPORT ABSTRACT MODEL

SET ABSTRACT MODEL

RESET HIER_COMPARE RESULT

`REset HIer_compare Result`
(*Setup / LEC Mode*)

Resets the results of the hierarchical comparison. It is useful when you do multiple hierarchical compare runs and you wish to display the results of each hierarchical compare separately.

Tcl Command

```
reset_hier_compare_result
```

Related Commands

RUN HIER_COMPARE

REPORT HIER_COMPARE RESULT

RESET

SAVE HIER_COMPARE RESULT

WRITE HIER_COMPARE DOFILE

RESOLVE

```
RESolve
     <module_name>
     [-All]
     [-Golden | -Revised]
     (Setup Mode)
```

Ungroups a module in the Golden or Revised design hierarchy. Resolving or ungrouping is the process of eliminating a module and promoting its content up one level of the hierarchy.

Tcl Command

```
resolve
```

Parameters

`<module_name>`	Resolves hierarchy for the specified module.
`-All`	Resolves the specified module within all hierarchies of the specified design.
`-Golden`	Resolves hierarchy in the Golden design. *This is the default.*
`-Revised`	Resolves hierarchy in the Revised design.

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD MOS DIRECTION

ADD NET ATTRIBUTE

ASSIGN PIN DIRECTION

DELETE CLOCK

DELETE MOS DIRECTION

DELETE NET ATTRIBUTE

RESTORE SESSION

```
**REStore SEssion**
     <session_name>
     (Setup / LEC Mode)
```

Restores a session you previously initiated and saved using the `SAVE SESSION` command.

Before entering this command, Conformal must be in its initial state. Therefore, you must either use the `RESET` command, or exit Conformal and restart it.

Important

You must run this restarted session on the same platform, same Conformal version, and same license configuration for which it was saved.

NOTE: `SAVE SESSION`/`RESTORE SESSION` command will be obsolete in the future. Use the new `CHECKPOINT/RESTART` command to save/restore complete process data.

Tcl Command

```
restore_session
```

Parameters

`<session_name>`	Restores the specified session.

Related Commands

INFO SESSION

RESET

SAVE SESSION

RUN HIER_COMPARE

```
RUN HIer_compare
     <dofile_name>
     [-ANALYZE_BOUNDARY_conditions | -NOANALYZE_BOUNDARY_conditions]
     [-BREAK_ABORT]
     [-BREAK_NONEQ]
     [-CHECK_NONEQ]
     [-COMPARE_String <string>]
     [-DYNamic_hierarchy | -NODYNamic_hierarchy]
     [-NOANALYZE_abort | -ANALYZE_abort]
     [-NODYNAMIC_MODULEs <golden_module> <revised_module> ...]
     [-NOREStart | -REStart]
     [-RETIMED_modules [-TOP | -NOTOP]]
     [-ROOT_BLACKBOX]
     [-ROOT_module <golden_module> <revised_module>]
     [-COMPARE_TOP_MODULE]
     [-VERBOSE]
     (Setup Mode)
```

Note: This requires a Conformal XL license.

Runs dynamic hierarchical comparison. This command on completion produces one of the following three statuses:

- *Equivalent*: all the compared modules are equivalent.
- *Nonequivalent*: at least one of the compared module is nonequivalent.
- *Inconclusive*: indicates one of the following conditions:
  - at least one of the compared module has abort points
  - at least one module is not-compared (for example, due to running the `ADD MODULE ATTRIBUTE -compare_effort none` command)
  - at least one module has incomplete compare result (for example, due to extra primary outputs)

  Note: When the status is *Inconclusive* the number of abort modules, not-compared modules, or modules that have incomplete compare result are reported.

During each module comparison, the `COMPARE` command is appended with the "`-abort_stop 1`" option to provide quick turnaround time.

For more information, see Dynamic Hierarchical Comparison in the *Conformal Equivalence Checking User Guide*.

Tcl Command

```
run_hier_compare
```

Parameters

`<dofile_name>`	Specifies the name of the hierarchical dofile that was generated with the `WRITE HIER_COMPARE DOFILE` command. **Note:** Manually editing or modifying this hierarchical dofile prior to running the `RUN HIER_COMPARE` command might lead to unexpected results. If you want to edit or modify the hierarchical dofile, use the static hierarchical comparison (`dofile hier.do`).
`-ANALYZE_BOUNDARY_conditions`	Reduces the number of flattened modules by resolving boundary constraints. *This is the default*.
`-NOANALYZE_BOUNDARY_conditions`	Do not perform resolution on boundary constraints.
`-BREAK_ABORT`	The comparison stops when it encounters an abort module. To continue comparing from the next module in the hierarchy, run the RUN HIER_COMPARE command.
`-BREAK_NONEQ`	The comparison stops when it encounters a nonequivalent module. To continue comparing from the next module in the hierarchy, use the RUN HIER_COMPARE command. Note: This command option is only supported when using the `-NODYNamic_hierarchy` option.
`-CHECK_NONEQ`	Identify NEQ modules before running the `ANALYZE DATAPATH` command and skip datapath analysis for these modules.

`-COMPARE_String <string>`
: Replaces the compare command in the hierarchical dofile with a string of compare commands while running hierarchical comparison.

  If the option `-COMPARE_String <string>` is also specified with the `WRITE HIER_COMPARE DOFILE` command, LEC still replaces the commands while running hierarchical comparison.

  Use the semi-colon character (`;`) to separate commands. Use double quotes to surround each compare command (see "Examples" section below).

`-DYNamic_hierarchy`

Auto-flattens the submodules to propagate any design errors to the top level. The flattened modules are merged to the next level in the hierarchy and compared at that level.

When using this option, -`noneq_stop 1` is automatically appended to the `compare` command during each module comparison.

This is the default.

`-NODYNamic_hierarchy`

Runs static hierarchical comparison without auto-flattening the submodules.

`-NODYNAMIC_MODULEs <golden_module> <revised_module>`

Does not perform dynamic flattening when the specified module pairs are the current root modules.

`-NOANALYZE_abort`
: Appends "`-abort_stop 1`" to the `compare` command during each module comparison. *This is the default*.

  However, if you use the `SET ANALYZE OPTION -auto` command, it will override this option.

`-ANALYZE_abort`
: Inserts the `ANALYZE ABORT -compare` command into each uncompared and aborted module's compare script.

  If this option is used, the `COMPARE` command is not appended with "`-abort_stop 1`" during each module comparison.

`-NORESTart`	Continues an interrupted session, preserving the previous compare results. *This is the default*. You can interrupt dynamic hierarchical comparison by pressing `Ctrl-c`.
`-RESTart`	Deletes the previous comparison results.
`-RETIMED_modules [-TOP | -NOTOP]`	Compares and blackboxes the submodules with the `PIPELINE_Retime` attribute. The `PIPELINE_Retime` attribute can be attached to a module using the ADD MODULE ATTRIBUTE command. For this option to work correctly, modules with `PIPELINE_Retime` attribute should exist in the hierarchical dofile script. `-TOP` runs the comparison of the top module such that submodules without the `PIPELINE_Retime` attribute are fully flattened. *This is the default* for `-RETIMED_module`. `-NOTOP` specifies that comparison stops after the modules with the `PIPELINE_Retime` attribute have been compared and blackboxed. The hierarchical result is reported as 'Inconclusive' because the entire design is not compared.
`-ROOT_BLACKBOX`	Blackboxes the root modules if the comparison result of the root module pair is equivalent.
`-ROOT_module <golden_module> <revised_module>`	Uses the specified modules as the root modules. This is similar to the `-Module` option with the `WRITE HIER_COMPARE DOFILE` command without having to the regenerate the dofile.
`-COMPARE_TOP_MODULE`	Re-compares the dofile top module or the user specified module by `-ROOT_module` option, even if the module has existed Equivalent or Non-equivalent comparison result.
`-VERBOSE`	Lists all the hierarchical constraints and additional information.

Examples

- The following command uses the `hier.do` dofile for hierarchical comparison:

```
run hier_compare hier.do
```

If the previous comparison run of the `hier.do` dofile resulted in three aborted modules, you can run a second comparison using the following command:

```
run hier_compare hier.do -analyze_abort
```

This command only operates on aborted modules from the previous run, and automatically runs the `ANALYZE ABORT -compare` command after the default `COMPARE` command.

- The following command uses m4 as the root module for both the Golden and Revised designs, deleting the previous comparison results:

```
run hier_compare hrcmod.do -root_module m4 m4  -restart
```

- The following command runs hierarchical comparison on modules with the `PIPELINE_Retime` attribute attached:

```
run hier_compare hier.do -retimed_modules
```

- In the following command, the compare command in the hierarchical dofile is replaced with two commands during each module comparison, `set compare effort low` and `compare -abort_stop 1 -noneq_stop 1`:

```
run hier_compare hier.do -compare_string \
"set compare effort low; compare -abort_stop 1 -noneq_stop 1"
```

Related Commands

ANALYZE ABORT

COMPARE

REPORT HIER_COMPARE RESULT

RESET HIER_COMPARE RESULT

SAVE HIER_COMPARE RESULT

WRITE HIER_COMPARE DOFILE

RUN LIBRARY CHECK

```
RUN LIbrary Check
     [-USED]
     (Setup Mode)
```

Note: This is a Conformal Low Power command and a native 1801 feature.

Performs 1801 library checkers. This command can be run without reading the design.

Tcl Command

`run_library_check`

Parameters

`-USED`	Applies library consistency checking to the cells used in the design instead of checking every library cell.

Example

The following performs 1801_LIB checkers at all read-in library files. Not require a design read-in.

```
TCL_SETUP > set_lowpower_option -native_1801
TCL_SETUP > read_library -liberty -lp <liberty files>
TCL_SETUP > run_library_check
```

The following performs 1801_LIB checkers at the library cells used at the design.

```
TCL_SETUP > set_lowpower_option -native_1801
TCL_SETUP > read_library -liberty -lp <liberty files>
TCL_SETUP > read_design <design>
TCL_SETUP > run_library_check -used
```

RUN PARALLEL COMPARE

```
RUN PArallel Compare
     [-ABORT_Print]
     [-ABORT_Stop <integer>]
     [-GATE_TO_GATE]
     [-NONEQ_Print]
     [-NONEQ_Stop <integer>]
     [-SUBMIT_OPTIONs <string>]
     [-TEST]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Runs equivalency checking comparison between the Golden and Revised designs on the added compared points using parallel processing. During the comparison, the Conformal software displays following information:

- A progress percentile number that shows the completion rate
- A running count that shows the number of key points that have been compared along with the total number of nonequivalent key points

Alternatively, you can run parallel comparison using multithreads; this method is easier to use and eliminates the complexity of setting up the parallel processing environment. Instead of `RUN PARALLEL COMPARE`, use `compare -threads` or `set parallel option -threads`.

Tcl Command

```
run_parallel_compare
```

Parameters

`-ABORT_Print`	Displays the abort points as they are found.
`-ABORT_Stop <integer>`	Stops the comparison after finding the specified number of abort points.
`-GATE_TO_GATE`	Enables an algorithm that might improve the run time of large gate-to-gate netlist comparisons.
`-NONEQ_Print`	Displays the nonequivalent points as they are found.

`-NONEQ_Stop <integer>`	Stops the comparison after finding the specified number of nonequivalent points.
`-SUBMIT_OPTIONs`	Specifies the options which will replace the keyword `<submit_options>` in the submit command line (see the SET PARALLEL OPTION `-SUBMIT_COMMAND_LINE` command).
`-TEST`	Launches qualification run to test if the environment is suitable for parallel processing.

Examples

- The following command tests if the environment is suitable for parallel processing:
  ```
  run parallel compare -test
  ```
- The following command will start the equivalency comparison using parallel processing with the parameters specified by the previously run `SET PARALLEL OPTION` command.:
  ```
  run parallel compare
  ```
- The following commands run the first parallel job in queue q1, and the second parallel run in queue q2.
  ```
  run parallel compare -submit_options "-q q1"
  run parallel compare -submit_options "-q q2"
  ```

Related Commands

COMPARE

SET PARALLEL OPTION

RUN PARTITION_COMPARE

```
RUN PARTition_compare
     [-Keypoint <identifier...>]
     [-Number <number>]
     [-THREADS <integer>[,<integer>]]
     [-VERBOSE]
     (LEC Mode)
```

Note: This requires a Conformal XL license.

Runs comparisons with functional partitioning. You can specify partitioned key points in the Golden design and the number of key points for a partition. If no key points are specified, this command will automatically choose appropriate key points for the partition.

Note: You do not need to switch to Setup mode to flatten the netlist in each partition iteration. With the constants assigned on the selected key points, comparison can become easier in each partition iteration.

For example, when abort points are encountered in comparison, you can run this command to do functional partitioning for the abort points.

`-Keypoint <identifier...>`	
	Specifies the partition key point in the Golden design. The key point can be specified by gate instance pathname or gate ID. If no key points are specified, this command will automatically choose appropriate key points for the partition.
`-Number <number>`	Specifies the number of key points for a partition. The maximum number of compare iterations is the base-2 exponent of the partitioned key point number. The default partitioned key point number is 8.
`-THREADS <integer>[, <integer>]`	
	Specifies the minimum and maximum number of threads that can be executed at the same time. If only one number entered, this specifies both the minimum and maximum number of threads. For example, '`-threads 2`' specifies two threads; '`-threads 2,4`' specifies a minimum of two threads, and a maximum of four threads.
`-VERBOSE`	Provides additional information in the functional partition.

Related Commands

COMPARE

SAVE DOFILE

```
SAVe DOfile
     <filename>
     [-Replace]
     (Setup / LEC Mode)
```

Saves the commands entered during the current session to a file. Use the saved dofile later as a batch file to repeat the Conformal session.

When running a Conformal session from a dofile, this command does not save individual commands included in a separate dofile (that is, Conformal saves the manually entered commands, which can include a `dofile <filename>` command).

Note: If the filename you specify already exists, you must use either the `-replace` or `-append` option.

Tcl Command

```
save_dofile
```

Parameters

`<filename>`	Writes the dofile to the specified file.
`-Replace`	Replaces the contents of the specified preexisting file.

Related Commands

DOFILE

SET COMMAND PROFILE

SET LOG FILE

SAVE HIER_COMPARE RESULT

```
SAVe HIer_compare Result
    (LEC Mode)
```

Saves the hierarchical comparison results of the module comparison. If the `WRITE HIER_COMPARE DOFILE` command is used, this command is placed after every module compared.

After the hierarchical comparison of all modules is complete, use the `REPORT HIER_COMPARE RESULT` command to display the results of the hierarchical comparison.

Related Commands

REPORT HIER_COMPARE RESULT

RESET HIER_COMPARE RESULT

RUN HIER_COMPARE

WRITE HIER_COMPARE DOFILE

SAVE SESSION

```
SAVe SEssion
     <session_name>
     [-REPlace]
     (Setup / LEC Mode)
```

Saves your session up to a current point and outputs the session file.

To restore the session later using the `RESTORE SESSION` command. You can use this command if priorities demand that another session preempt your session.

Important

When you use the `RESTORE SESSION` command, you must run the restarted session on the same platform and same Conformal version.

For more, refer to Saving and Restoring a Session in the *Conformal Equivalence Checking User Guide*.

NOTE: `SAVE SESSION` command only saves design data and will be obsolete in the future. Use the new `CHECKPOINT` command to save complete process data.

Tcl Command

```
save_session
```

Parameters

`<session_name>`	Attaches this session name to the saved session.
`-REPlace`	Replaces the existing session. If the session already exists, it will be overwritten and no backup copy will be created. By default, backup copies are created automatically. This option is useful if you want to save disk space and only need to save your session occasionally.

Related Command

INFO SESSION

RESTORE SESSION

SEARCH

```
SEArch
     [-USage] <string1> [<string2>] ...
     (Setup / LEC Mode)
```

Searches the database of commands and parameters, and displays those commands that match all of the specified strings. Strings can be specified in any order; however, every specified string must match.

Tcl Command

```
search
```

Parameters

`-USage`	Displays the commands that have parameters that match the search string. This outputs the entire command syntax for each command.
`<string1>`	Displays commands that match the specified string.
`<string2> ...`	Displays commands that match additional specified strings.

Related Command

HELP

SET ABSTRACT MODEL

```
SET ABSTract Model
     [-ALL |-MODule <module_name ...>]
     [-NOBUF_AMP | -BUF_AMP]
     [-NODOMINOLATch | -DOMINOLATch]
     [-NODYNSTate | -DYNSTate]
     [-NOKEEPER2PUllup | -KEEPER2PUllup]
     [-NOKEEPERSTate | -KEEPERSTate]
     [-NOIGNORE_DLAT_CONTENTION | -IGNORE_DLAT_CONTENTION]
     [-NOMEM_BL_EQualizer | -MEM_BL_EQualizer]
     [-NOMULTICLOCKPRECHARGE | -MULTICLOCKPRECHARGE]
     [-NOPRE_CHARGE_KEEP_Clock | -PRE_CHARGE_KEEP_Clock]
     [-NO_SEQUENTIAL_CONSTANT | -SEQUENTIAL_CONSTANT]
     [-NOWEAK_BUS_HOLDER | -WEAK_BUS_HOLDER]
     [-NOWEAKPULLDOWN | -WEAKPULLDOWN]
     [-NOWEAKPULLUP | -WEAKPULLUP]
     [-REPHASE_BY_NAME_POSitive <net_name>]
     [-REPHASE_BY_NAME_NEGative <net_name>]
     [ | -RESTRICT_Patterns | -RESTRICT_Modules]
     [-TRANSFORM_PULSE_GENERATOR_ON]
     [-Both | -Golden | -Revised ]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Specifies certain conditions for abstracting transistor logic. The design must be read in (`READ DESIGN` command) before using this command.

Refer to the *Conformal Equivalence Checking User Guide* for additional information about using this command in the Conformal GXL flow.

Tcl Command

```
set_abstract_model
```

Parameters

`-ALL`	Abstracts transistor logic from all modules within the given defaults. *This option is the default.*
`-NOBUF_AMP`	Do not handle buffered-type sense amplifiers, level shifters, pre-charge, and equalization. *This is the default.*

`-BUF_AMP`	Handles the following portions of a circuit: buffered-type sense amplifiers, level shifters, pre-charge, and equalization.
`-NODOMINOLATch`	Do not abstract pre-charge logic functions as a latch. *This is the default.*
`-DOMINOLATch`	Abstracts pre-charge logic functions as a latch. This assumes that data input is stable in active clocks.
`-NODYNSTate`	Do not regard tristate table nets as latches. *This is the default.*
`-DYNSTate`	Regards tristate table nets as latches.
`-NOKEEPER2PUllup`	Do not regard charge keepers as weak pull-up devices. *This is the default.*
`-KEEPER2PUllup`	Regards charge keepers as weak pull-up devices.
`-NOKEEPERSTate`	Do not regard charge keepers as latches. *This is the default.*

`-KEEPERSTate`	Regards charge keepers as latches.
	`add net attribute CLOCK0 | CLOCK1`
	`add clock 0 | 1`
	When you use `-pre_charge_keep_clock`, the resulting logic is equivalent to RTL that explicitly models the pre-charge condition, rather than RTL that models only the evaluate function. In the latter, the output function is not defined during pre-charging.

`-NOIGNORE_DLAT_CONTENTION`	Stops forming the D-Latch when contention on a net is detected. *This is the default.*
`-IGNORE_DLAT_CONTENTION`	Continues to try and form the D-Latch, even if contention on a net is detected. By default, the `SET ABSTRACT MODEL` command stops the execution of abstraction of latches and flip-flips (state elements) when a power to ground through a stack of active ON transistors is possible. Use this option to report the short and continue to abstract the state element.

`-NOMEM_BL_EQualizer`	Do not handle bit-line pre-charge, and equalization. *This is the default.*
`-MEM_BL_EQualizer`	Handles circuits that include bit-line pre-charge, and equalization.
`-NOMULTICLOCKPRECHARGE`	Do not propagate clocks through logic gates which have more than one clock input. *This is the default.*
`-MULTICLOCKPRECHARGE`	Propagates clocks through logic gates which have more than one clock input.
`-NOPRE_CHARGE_KEEP_Clock`	In domino logic, does not regard pre-charge clocks as part of the logic function. *This is the default.*
`-PRE_CHARGE_KEEP_Clock`	For domino logic, regards pre-charge clocks as part of the logic function. This option includes the defined pre-charge clock in the abstracted logic function (the default behavior removes the defined pre-charge clock from the abstracted logic). This is indicated when you define a precharge clock with one of the following commands:
`-NO_SEQUENTIAL_CONSTANT`	Do not propagate constant value through flip-flops. This is default.
`-SEQUENTIAL_CONSTANT`	Enables constant propagation through flip-flops.
`-NOWEAK_BUS_HOLDER`	Do not place weak attribute on the bus holder. *This is the default.*
`-WEAK_BUS_HOLDER`	Place weak attribute on the bus holder. With this option set, LEC can remodel the bus holder based on the attribute.
`-NOWEAKPULLDOWN`	Do not regard devices that are tied to NMOS as weak devices. *This is the default.*
`-WEAKPULLDOWN`	Regards devices that are tied to NMOS as weak devices.

`-NOWEAKPULLUP`	Do not regard devices that are tied to PMOS as weak devices. *This is the default.*
`-WEAKPULLUP`	Regards devices that are tied to PMOS as weak devices.
`-REPHASE_BY_NAME_POSitive <net_name>`	
	Gives `ABSTRACT LOGIC` a hint about the desired phase of state elements, such as D-latches and DFFs. When abstracting state elements, `ABSTRACT LOGIC` will choose a phase for each state element, where `<net_name>` specifies the net which will be driven by the 'Q' pin, if possible. If this is not possible, then abstraction will try to choose a net for the 'Qn' pin which has a name specified by the `-rephase_by_name_negative` option. **Note:** The `SET MAPPING METHOD` command's `-phase` option will allow mapping and comparison of state elements with different phases in the Golden and Revised designs. Consider running `SET MAPPING METHOD -phase` before using this option, as it requires less effort.
`-REPHASE_BY_NAME_NEGative <net_name>`	
	Specifies the net which will be driven by the 'Qn' pin, if possible.
`-RESTRICT_Patterns`	All patterns not in any set pattern match apply to all modules not specified. Only patterns in any set pattern match apply to the specified module(s).
`-RESTRICT_Modules`	Pattern remodeling is limited to the specified patterns and modules. Only pattern in any set pattern match apply to the specified module(s).
`-MODule <module_name ...>`	
	Abstracts transistor logic from the specified modules.
`-TRANSFORM_PULSE_GENERATOR_ON`	
	Enables pulse transformation.
`-Both`	Specifies abstraction conditions for both the Golden and Revised designs. *This is the default.*
`-Golden`	Specifies abstraction conditions for the Golden design.
`-Revised`	Specifies abstraction conditions for the Revised design.

Examples

Sample Dofile:

```
read design test.v -golden
set abstract model -keeper2pullup -weakpullup -golden
report abstract model -golden
abstract logic
```

Related Commands

ABSTRACT LOGIC

ADD CLOCK

ADD NET ATTRIBUTE

REPORT ABSTRACT MODEL

REPORT PULSE GENERATOR

RESET ABSTRACT MODEL

SET PATTERN MATCH

SET ABSTRACT OPTION

```
SET ABSTract Option
     [-NOENABLE_CELL_COMPARE | -ENABLE_CELL_COMPARE]
     [-CELL_COMPARE_DIRectory <dir_name>]
     [-CONSTRAINT_File <file_name>]
     [-BOTH | _Golden | -REVised]
     (Setup Mode)
```

Note: This requires a LEC Next-Gen license.

Sets advanced methodology and related options for abstracting transistor logic.

The affected commands are `ABSTRACT LOGIC`, `VALIDATE LIBRARY`

Tcl Command

```
set_abstract_option
```

Parameters

`-ENABLE_CELL_COMPARE`	Enables advanced abstraction method, Circuit Analysis Based Logic Abstraction, for standard cell libraries. The method analyzes transistor sequential behaviors to decide how to best perform abstraction.
`-NOENABLE_CELL_COMPARE`	Disables the advanced abstraction method, Circuit Analysis Based Logic Abstraction. *This is the default*.
`-CELL_COMPARE_DIRectory <dir_name>`	Specifies the working directory location that is created during abstraction with advanced abstraction method. The directory includes files which describe the detailed results for rejected modules during abstraction. The default location is "LEC_ABSTRACT_RES".
`-CONSTRAINT_File <file_name>`	Specifies the Verilog constraint file to be used with the Circuit Analysis Based Logic Abstraction method. Constraints need to be only applied on input pins.
`-Both`	Applies the options to both Golden and Revised designs. This is default.

`-GOLden`	Applies the options to the Golden design.
`-REVised`	Applies the options to the Revised design.

Examples

The following example demonstrates the usage of a Verilog constraint file for the Circuit Analysis Based Logic Abstraction method.

```
set abstract option -enable_cell_compare -constraint_file constraint.v

where the constraint.v is as follows,

module top(a,b,o);
  input a,b;
  output o;
  $constraint(a&b);
endmodule
```

Related Commands

ABSTRACT LOGIC

VALIDATE LIBRARY

SET ANALYZE OPTION

```
SET ANalyze Option
     [-ANALYZE_RENAMING_RULE]
     [-CUT | -NOCUT]
     [-DELETE_UNREACHable | -NODELETE_UNREACHable]
     [-EFFORT <MEDIUM | HIGH | ULTRA>]
     [-EFFORT_ANALYZE_ABORT <MEDIUM | HIGH >]
     [-EFFORT_ANALYZE_SETUP <MEDIUM | HIGH | ULTRA>]
     [-GATED_CLOCK | -NOGATED_CLOCK]
     [-LATCH_TRANSPARENT | -NOLATCH_TRANSPARENT]
     [-MAPPING_FILE <file_name>]
     [-NOABSTRACT_GATED_CLOCK | -ABSTRACT_GATED_CLOCK]
     [-NOANALYZE_ABORT_RESOLVE_XORTREE | -ANALYZE_ABORT_RESOLVE_XORTREE]
     [-NOANALYZE_ABORT_SNAPSHOT | -ANALYZE_ABORT_SNAPSHOP]
     [-NOAUTO
       | -AUTO
         [-ANALYZE_ABORT | -NOANALYZE_ABORT]
         [-ANALYZE_SETUP | -NOANALYZE_SETUP]
     ]
     [-NOLATCH_NO_HOLDING | -LATCH_NO_HOLDING]
     [-NOMODIFY_MAP | -MODIFY_MAP]
     [-NOPHASE_MAPPING | -PHASE_MAPPING]
     [-NOREPORT_MAP | -REPORT_MAP]
     [-NOSETUPINFO_SEQ_CONSTANT | -SETUPINFO_SEQ_CONSTANT]
     [-SEQ_MERGE | -NOSEQ_MERGE]
     [-SEQ_REDUNDANT | -NOSEQ_REDUNDANT]
     [-SNAPSHOT_DIR <SNAPSHOT_DIRectory>]
     [-TRANSFORM_SET_DOMINANT | -NOTRANSFORM_SET_DOMINANT]
     (Setup Mode)
```

Note: This requires a Conformal XL license.

When you use the `-AUTO` option, it automatically determines the best place to run the `ANALYZE SETUP` command. The `ANALYZE SETUP` command analyzes the netlists and sets up the flattened design for accurate comparison. This helps avoid false nonequivalences. This command can also analyze and remodel the following commonly-encountered setup issues: sequential constanting, sequential merging, loop cutting, clock gating, and phase mapping.

In addition, the Conformal software automatically runs the `ANALYZE ABORT` command whenever the comparison returns abort points but no nonequivalent points.

Tcl Command

```
set_analyze_option
```

Parameters

`-ANALYZE_RENAMING_RULE`	
	Enables renaming rule analysis during the flattening process. This applies only when balanced modeling is enabled.
`-CUT`	Enables cut point analysis during automatic setup. *This is the default*.
`-NOCUT`	Disables cut point analysis during automatic setup.
`-DELETE_UNREACHable`	Deletes functionally mapped unreachables during auto setup if they are non-equivalent. *This is the default*.
`-NODELETE_UNREACHable`	
	Do not delete functionally mapped unreachables during auto setup.
`-EFFORT <MEDIUM | HIGH | ULTRA>`	
	Specifies the effort level for automatic setup and abort point analysis. The higher the effort level, the better the analysis (however, it can increase analysis time as well). *Medium is the default when using the -auto option.*
`-EFFORT_ANALYZE_ABORT <MEDIUM | HIGH>`	
	Specifies the effort level for the `ANALYZE ABORT` command. `MEDIUM`: Default effort level. `HIGH`: Invokes functional partitioning for resource sharing in datapath analysis. It also invokes word-level datapath analysis for non-threaded '`ANALYZE ABORT`'.
`-EFFORT_ANALYZE_SETUP <MEDIUM | HIGH | ULTRA>`	

	Specifies the effort level for the `ANALYZE SETUP` command. The effort level controls the computation resources that LEC allocates during `ANALYZE SETUP`. If LEC reports that a proving process (such as, sequential constant) has aborted, increasing the effort may resolve the issue. Note: Specifically, with effort `ULTRA`, the `ANALYZE SETUP` command uses '`compare -effort light'` instead of '`compare -random`' to catch nonequivalences for modeling. The comparison can take significantly longer, but may resolve nonequivalences with one round of `ANALYZE SETUP`.
`-GATED_CLOCK`	Enables gated clock modeling. *This is the default*.
`-NOGATED_CLOCK`	Disables gated clock modeling.
`-LATCH_TRANSPARENT`	Converts transparent D-latches into buffers. *This is the default*.
`-NOLATCH_TRANSPARENT`	Automatic analysis will not convert transparent D-latches into buffers.
`-MAPPING_FILE <file_name>`	
	Specifies the mapping information file. With this file, you can provide mapping information before entering LEC mode. The mapping information in this file is used on an as needed basis and the result will depend on the `SET MAPPING METHOD` settings such as `-nounreach/-unreach` and `-bbox_name_match/-nobbox_name_match`.
`-NOABSTRACT_GATED_CLOCK`	
	Disable gated-clock abstraction during automatic setup. *This is the default*.
`-ABSTRACT_GATED_CLOCK`	
	Enable gated-clock abstraction during automatic setup.
`-NOANALYZE_ABORT_RESOLVE_XORTREE`	
	Disable resolving xortrees' aborts during command `ANALYZE ABORT` under multiple threads. *This is the default*.
`-ANALYZE_ABORT_RESOLVE_XORTREE`	
	Enable resolving xortrees' aborts during command `ANALYZE ABORT` under multiple threads.
`-NOANALYZE_ABORT_SNAPSHOT`	

	Sets the auto analyze abort snapshot for the RTL annotation OFF. *This is the default*
`-ANALYZE_ABORT_SNAPSHOT`	
	Sets the auto analyze abort snapshot for the RTL annotation ON. Use `SNAPSHOT_DIRectory` to specify the snapshot directory.
`-NOAUTO`	Disables automatic analysis. *This is the default.*
`-AUTO`	Enables automatic analysis.
`-ANALYZE_ABORT`	Enables automatic abort point analysis. When `-auto` is used with this option, the tool enables the `ANALYZE ABORT -compare` command when it encounters an abort during compare. *This is the default* when running this command with the `-auto` option.
`-NOANALYZE_ABORT`	Disables automatic abort point analysis.
`-ANALYZE_SETUP`	Enables automatic setup analysis. *This is the default* when running this command with the `-auto` option.
`-NOANALYZE_SETUP`	Disables automatic setup analysis.
`-NOLATCH_NO_HOLDING`	Automatic analysis will not model latches that do not have holding functions. *This is the default*.
`-LATCH_NO_HOLDING`	When enabled, automatic analysis models latches that do not have holding functions.
`-NOMODIFY_MAP`	Disable the modification of functional mapping after remodeling in automatic setup. *This is the default*.
`-MODIFY_MAP`	Enable the modification of functional mapping after remodeling in automatic setup.
`-NOPHASE_MAPPING`	Disables phase adjustment for DFF/D-LATCH mapping during automatic analysis. *This is the default*.

`-PHASE_MAPPING`	Enables phase adjustment for DFF/D-LATCH mapping during automatic analysis. Note: If the design has an inverter push, use "`set mapping method -phase`" for the first mapping (prior to "`set system mode lec`") because correct mapping is critical for modeling. Executing `set mapping method -phase` after entering lec mode cannot guarantee that all mapping issues will be resolved.
`-NOREPORT_MAP`	Do not report the mapping results after completing automatic analysis. *This is the default*.
`-REPORT_MAP`	Reports the mapping results after completing automatic analysis.
`-NOSETUPINFO_SEQ_CONSTANT`	Do not use constant guidance to perform group analysis.
`-SETUPINFO_SEQ_CONSTANT`	Use the sequential constant guidance to model golden, including sequential group and sequential constant x. When the option on, LEC analyzes and proves the constant group information from guidance.
`-SEQ_MERGE`	Enables the sequential merge modeling of DFF/D-LATCH during automatic analysis. *This is the default*.
`-NOSEQ_MERGE`	Disables the sequential merge modeling of DFF/D-LATCH during automatic analysis.
`-SEQ_REDUNDANT`	Enables removing redundant fan-out gates from DFFs and DLATs during automatic setup. *This is the default*.
`-NOSEQ_REDUNDANT`	Disables removing redundant fan-out gates from DFFs and DLATs during automatic setup.
`-SNAPSHOT_DIR <SNAPSHOT_DIRectory>`	Specifies the analyze abort RTL snapshot directory. User also needs to use `ANALYZE_ABORT_SNAPSHOT` to turn on the auto snapshot function.
`-TRANSFORM_SET_DOMINANT`	Balances the structure of set-dominant DFFs and DLATs during automatic setup. *This is the default*.
`-NOTRANSFORM_SET_DOMINANT`	

Disables structure balancing of set-dominant DFFs and DLATs during automatic setup.

Examples

Mapping file format:

The following illustrates the sample format of a mapping file (designated using this command's `-mapping_file` option):

`map.f`:

```
add mapped points dec_data[7] dec_data[7] \
  -type PI PI -module err_detect err_detect
add mapped points dec_data[6] dec_data[6]
  -type PI PI -module err_detect err_detect
add mapped points dec_data[5] dec_data[5]
  -type PI PI -module err_detect err_detect
add mapped points dec_data[4]
  dec_data[4] -type PI PI -module err_detect err_detect
```

Enabling automatic abort point analysis:

In the following example, `SET ANALYZE OPTION -auto` enables automatic abort point analysis.

```
set analyze option -auto
set system mode lec
add compare point -all
compare
```

If the tool encounters an abort after compare, it automatically enables the `ANALYZE ABORT -compare` command to resolve abort key points.

During the run, the tool displays something similar to the following:

```
========================================================
Compared points      PO     DFF       Total
--------------------------------------------------------
Equivalent           59     224       283
--------------------------------------------------------
Abort                0      2         2
========================================================
// Command: analyze abort -compare
========================================================
Compared points      PO     DFF       Total
--------------------------------------------------------
Equivalent           59     226       285
========================================================
```

Usage example:

For example, the following illustrates the usage of this command with the `-mapping_file` option in a flattened flow:

```
set analyze option -auto -mapping_file map.f
set system mode lec
add compare point -all
compare
```

The following illustrates the usage of this command with the `-mapping_file` option in a hierarchical flow:

```
set analyze option -auto -mapping_file map.f
write hier dofile hier.do ...
run hier hier.do
...
```

Related Command

ANALYZE ABORT

ANALYZE SETUP

SET_ATTR INPUT_PRAGMA_KEYWORD

```
SET_ATtr INPUT_PRAGMA_Keyword
     <string>
     (Setup Mode)
```

Specifies a keyword that the Conformal software must consider as an input pragma when it encounters it as the first word in a Verilog or VHDL source comment.

A pragma is a comment in the Verilog or VHDL source and is set off from ordinary comments by the pragma keyword. The pragma keyword is the first word listed in a pragma, and it notifies the Conformal software that the remainder of the comment is a command and not a comment. Changing this keyword allows you to set up compatibility with other tools.

Tcl Command

`set_attr_input_pragma_keyword`

Parameters

`string`	Specifies the name of the keyword for a tool vendor. *Default*: `pragma`, `synthesis`, `cadence`, `ambit`, `conformal`

Examples

Sample Dofile:

```
set_attr input_pragma_keyword rtl
set synthesis_off_command turn_off
set synthesis_on_command turn_on
```

After running these three commands, the Conformal and VHDL parsers will recognize the pragmas in the VHDL and Verilog Source files.

In a VHDL file, the code between `-- rtl turn_off` and `-- rtl turn_on` will not be translated.

In a Verilog file, the code between `// rtl turn_off` and `// rtl turn_on` will not be translated.

Related Commands

SET SYNTHESIS_OFF_COMMAND

SET SYNTHESIS_ON_COMMAND

SET CASE SENSITIVITY

```
SET CAse Sensitivity
     <OFf | ON>
     (Setup Mode)
```

Specifies whether names you enter are case sensitive. The system default is no case sensitivity for both the Golden and Revised designs.

Execute this command before `READ LIBRARY` and `READ DESIGN`. Use the `REPORT ENVIRONMENT` command to display the case sensitivity setting.

Tcl Command

```
set_case_sensitivity
```

Parameters

`ON`	Names that are entered are case sensitive.
`OFf`	Names that are entered are not case sensitive. *This is the system default.*

Related Command

REPORT ENVIRONMENT

SET CODE TRACE

```
SET COde Trace
     [OFf | ON | AUto]
     (Setup Mode)
```

Controls source code tracing functions.

Note: Enabling/disabling source code tracing functions must be done before reading in the design.

When source code tracing is enabled, the following advanced design browsing capabilities are available in the Source Code Manager:

1. Macro expansion: Macro expansion results are displayed in an information window when you click on a macro variable.
2. Parameter value annotation: Parameter values are annotated in an information window when you click on a parameter variable.
3. Symbol definition lookup: You can view the definition of a symbol by clicking on it and choosing *Definition* from the pop-up menu.
4. Code block folding/unfolding: Code blocks (such as modules, functions, tasks, always blocks, case switches, if-else, and loops) can be folded and unfolded for better readability.

   To fold a code block, right click on the first line and select *Fold* from the pop-up menu. This folds the whole code block into one line.

   To unfold a code block, right click on the block, and select *Unfold* in the pop-up menu. The code block is restored back to its original status.

For more information on these capabilities, refer to the 14.1 web interface article titled, *Conformal Verilog Design Tracing*.

Tcl Command

```
set_code_trace
```

Parameters

`OFf`	Turn off the source code tracing functions in the Source Code Manager. *This is the default.*

`ON`	Turn on the source code tracing functions in the Source Code Manager.
`AUto`	Automatically enables source code tracing functions in the Source Code Manager for RTL designs; automatically disables these functions for large scale gate-level netlists.

SET COMMAND ECHO

```
SET COmmand ECho
    <ON | OFF>
    (Setup / Verify Mode)
```

Controls whether the tool echos (or repeats on the screen) the command typed.

Tcl Command

`set_command_echo`

Parameters

`ON`	Enables command echoing.
`OFF`	Disables command echoing.

Example

For example, we save the following in the dofile and call it mydofile1.do:

```
read design ./ccr871626/test.v -verilog -root sub0
set command echo off
read design ./ccr871626/test.v -verilog -root sub0 -replace
set command echo on
read design ./ccr871626/test.v -verilog -root sub0 -replace
```

When you execute the dofile, the tool will only echo two of the `READ DESIGN` commands. It does not echo the second `READ DESIGN` command, because it was typed after the `SET COMMAND ECHO OFF` command was set:

```
// Command: dofile r
// Command: read design ./ccr871626/test.v -verilog -root sub0
// Parsing file ./ccr871626/test.v ...
// design root module is set to 'sub0'
// Note: Read VERILOG design successfully
// Read design summary: Error: 0, Warning: 10, Note: 0
// Note: Use 'report rule check -category library_design -verbose' for details.
// Command: set command echo off
// Warning: Existing design has been deleted
// Parsing file ./ccr871626/test.v ...
// design root module is set to 'sub0'
// Note: Read VERILOG design successfully
// Read design summary: Error: 0, Warning: 10, Note: 0
// Note: Use 'report rule check -category library_design -verbose' for details.
// Command: read design ./ccr871626/test.v -verilog -root sub0 -replace
// Warning: Existing design has been deleted
// Parsing file ./ccr871626/test.v ...
// design root module is set to 'sub0'
// Note: Read VERILOG design successfully
// Read design summary: Error: 0, Warning: 10, Note: 0
// Note: Use 'report rule check -category library_design -verbose' for details.
```

SET COMMAND PROFILE

```
SET COmmand PRofile
    [OFF | ON]
    (Setup / LEC Mode)
```

Starts or stops recording a profile of commands executed in Conformal. This command records the order of command execution and the memory use. The profile includes commands used in the GUI mode.

Use the `REPORT COMMAND PROFILE` command to view the profile.

Tcl Command

```
set_command_profile
```

Parameters

`OFF`	Stops tracking executed commands. *This is the default.*
`ON`	Starts tracking executed commands.

Related Commands

REPORT COMMAND PROFILE

SET LOG FILE

SET COMPARE EFFORT

```
SET COmpare Effort
     <Low | Medium | High | Auto | LIght | COMPlete>
     (Setup / LEC Mode)
```

Specifies the amount of effort equivalency checking applies to the key points comparison. If you know your designs have many complex key points, increase the effort level. However, when you raise the effort level, you also increase the amount of time involved in checking. Hence, you increase the total CPU time.

Use the `REPORT ENVIRONMENT` command to display the compare effort setting. *The system default is set to low compare effort.*

Tcl Command

```
set_compare_effort
```

Parameters

`Low`	Applies low effort to equivalency checking for each gate. *This is the default.*
`Medium`	Applies greater effort to equivalency checking for each gate.
`High`	Applies the maximum effort to equivalency checking for each gate.
`Auto`	Starts with low effort and automatically increases the compare effort if abort points are present in the design.
`LIght`	Applies minimal effort to equivalency checking for each gate.
`COMPlete`	Performs equivalency checking for each gate until the comparison results in an EQ or NONEQ result. With this option, the tool never returns an abort (in other words, if EQ or noneq is not returned, the compare will go on indefinitely).

Related Commands

COMPARE

REPORT ENVIRONMENT

SET COMPARE OPTIONS

```
SET COmpare Options
     [-GENLATCH | -NOGENLATCH]
     [-MAX_EFFORT <Auto | Low | Medium | High | COMPlete>]
     [-NOALLGENLATCH | -ALLGENLATCH]
     [-NOENHANCE_64BIT | -ENHANCE_64BIT]
     [-NOGATE_TO_GATE | -GATE_TO_GATE]
     [-NOREPORT_BBOX_INPUT | -REPORT_BBOX_INPUT]
     [-NOREPORT_SINGLE_LINE_SUMMARY | -REPORT_SINGLE_LINE_SUMMARY]
     [-NOSINGLE | -SINGLE]
     [-NOVERBOSE | -VERBOSE]
     [-THREADS <integer>[,<integer>]]
     [-TURBO | -NOTURBO]
     [-VERIFY_Disabled_ports]
     (Setup / LEC Mode)
```

Turns on options to the comparison process.

Tcl Command

`set_compare_options`

Parameters

`-GENLATCH`	Compares latches as generic latches by analyzing all logic cones simultaneously. The software automatically determines which latches are to be compared as generic latch, and the rest are compared by individual logic cones. *This is the default.* With this option, you can compare latches that are truly functionally equivalent, even though the logic cones of the separate input pins are not.

`-MAX_EFFORT <Auto | Low | Medium | High | COMPlete>`

	Sets the maximum amount of comparison effort when using effort auto (with `compare -effort auto` or `set compare effort auto`). When `-max_effort` is set to `auto` (which is the default), the tool will start with the lightest effort level and will keep increasing the compare effort if abort points are still present in the design. Otherwise, the tool will stop at the maximum effort set by this option.
`-NOGENLATCH`	Specifies that no latch is compared as a generic latch. All latches are compared by individual logic cones.
`-NOALLGENLATCH`	Do not compare all latches as generic latches. *This is the default.* This option has no effect when using `-nogenlatch`. **Note:** The input cones compared are set cones, reset cones, clock cones, and data cones.
`-ALLGENLATCH`	Compares all latches as generic latches. This option has no effect when using `-nogenlatch`.
`-NOENHANCE_64BIT`	Disable the 64-bit comparison enhancement. *This is the default.*
`-ENHANCE_64BIT`	Enhance the abort resolution capability of 64-bit comparison. Note: This command is only available on 64-bit executable and the enhancement may lead to fewer number of aborts than the 32-bit executable
`-NOGATE_TO_GATE`	Do not enable the gate-to-gate algorithm. *This is the default*.
`-GATE_TO_GATE`	Enables the gate-to-gate algorithm, which might improve the run time of large gate-to-gate netlist comparisons.
`-NOREPORT_BBOX_INPUT`	Do not report the blackbox input pins in the compare report results. *This is the default.*
`-REPORT_BBOX_INPUT`	Reports the blackbox input pins in the compare report results (Equivalent, Nonequivalent, Abort, and Not-compared).
`-NOREPORT_SINGLE_LINE_SUMMARY`	Do not print a single line summary of the compare results. *This is the default.*

`-REPORT_SINGLE_LINE_SUMMARY`	
	Prints a single line summary of the compare results.
`-NOSINGLE`	Compares key points in groups. *This is the default*.
`-SINGLE`	Compares each key point as a single point.
`-NOVERBOSE`	Do not display the detailed compare information, such as the logic duplication in multithreaded comparison. *This is the default.*
`-VERBOSE`	Display detailed compare information, such as the logic duplication in multithreaded comparison.
`-THREADS <integer>[,<integer>]`	
	Specifies the number of compare threads for *only* the `COMPARE` command. For example, '`-threads 2`' specifies two threads; '`-thread 2,4`' specifies a minimum of two threads, and a maximum of four threads. The `set parallel option -threads #` command specifies the number of threads for *both* the `COMPARE` and the `ANALYZE ABORT` commands. If you specify both the `SET COMPARE OPTIONS -threads` and the `SET PARALLEL OPTION -threads` commands, the `SET COMPARE OPTIONS -thread` command supersedes the setting of the `SET PARALLEL OPTION -thread`s command, but for just the number of threads in the `COMPARE` command. See example below.
`-TURBO`	Fine-grained partition for compare parallelization to utilize CPU more efficiently.
`-NOTURBO`	Specifies not using parallel turbo threads in compare command. *This is the default*.
`-VERIFY_Disabled_ports`	
	Compares data cones even if their clocks are disabled. By default, a data cone will not be compared if its corresponding clock port is tied to a constant (for DFFs) or to zero (for latches). **Note:** You should use this command option before running the first `COMPARE` command. If you use this after running `COMPARE`, this option has no effect.

Examples

LEC uses two threads for the `COMPARE` command and four threads for the `ANALYZE ABORT` command.

```
set parallel option -threads 4
set compare options -threads 2
```

Related Command

COMPARE

SET PARALLEL OPTION

SET CPU LIMIT

```
SET CPu Limit
     <integer> <-Days | -Hours | -Minutes>
     [-COMMAND <command_name>]
     [-NOKill]
     [-WALLTIME]
     (Setup / LEC Mode)
```

Specifies the time limit for the LEC session. The system default is 525,600 minutes. Set the time limit for minutes, hours, or days.

Use the `REPORT ENVIRONMENT` command to display the setting for the CPU time limit.

Note: When the Conformal software reaches the specified CPU limit, it stops all processing and exits.

Tcl Command

`set_cpu_limit`

Parameters

`<integer>`	Specifies a positive integer for the CPU time limit.
`-Days`	Specifies that the CPU time limit refers to days.
`-Hours`	Specifies that the CPU time limit refers to hours.
`-Minutes`	Specifies that the CPU time limit refers to minutes.
`-COMMAND <command_name>`	Specifies a Conformal command to run before exiting the software.
`-NOKill`	Prevents the software from exiting. This returns the command prompt.
`-WALLTIME`	Specifies that the time limit is in real clock time.

Examples

- The following commands show an example of using the `SET CPU LIMIT` command with and without the `-WALLTIME` option.

The time is 11:00 am and you start two Conformal sessions on the same machine, executing the same dofile, that will run more than 20 minutes. You set the time limit for session 1 in real clock time with the following command:

```
set cpu limit 10 -minutes -walltime
```

You set the time for session 2 at the same limit but without using the real clock time with the following command:

```
set cpu limit 10 -minutes.
```

At 11:10, session 1 will terminate because the real clock time has elapsed 10 minutes. However, session 2 might not terminate because the real time it consumed during this 10 minutes is less than 10 minutes if some of the time is consumed by other processes running on the machine.

- The following command specifies that before the software exits after reaching 1 minute CPU time limit, it will run the `USAGE` command.

  ```
  set cpu limit 1 -minute -command usage
  ```

Related Command

REPORT ENVIRONMENT

SET DATAPATH OPTION

```
SET DAtapath Option
     [-EFFort <MEDium | HIgh>]
     [-ENABLE_PARALLEL_DATAPATH_ANALYSIS]
     [-MERGE | -NOMERGE]
     [-NOADDERTREE | -ADDERTREE]
     [-NOAUTO | -AUTO ]
     [-NOMODULE | -MODULE [-RESOURCEFILE <filename>] [-ISOLATE_ABORT_MODULE]]
     [-NOSHARE | -SHARE]
     [-NOWORDLEVEL | -WORDLEVEL]
     [-NOFLOWGRAPH | -FLOWGRAPH]
     [-THREADS <integer>[,<integer>]]
     [-GOLDEN | -REVISED]
     [-Verbose]
     (Setup / LEC Mode)
```

Note: This command requires a Conformal XL license.

Specifies whether Conformal automatically analyzes the datapath on switching from Setup to LEC mode and whether to apply operator merging. The results of the analysis enable Conformal XL to automatically resolve multipliers, operator merging, and resource sharing problems. Note that portions of this command are global settings (specifically, anything not under `-auto`) and can override `ANALYZE DATAPATH` default settings (see example section).

Note: You cannot run datapath analysis without first mapping the Revised design key points to the Golden design key points.

Tcl Command

`set_datapath_option`

Parameters

`-EFFort <MEDium | HIgh>`

Specifies the effort level. Choose `medium` level(the default), or `high` level to help provide better analysis of some multipliers, but can increase the analysis run time.

`-ENABLE_PARALLEL_DATAPATH_ANALYSIS`

Enables parallel analysis for wordlevel datapath and flowgraph datapath analysis. *This option requires a Smart LEC license*.

`-MERGE`	Automatically applies the operator merging technique when switching from Setup to LEC mode. *This is the default.*
`-NOMERGE`	Do not automatically apply the operator merging technique when switching from Setup to LEC mode.
`-NOADDERTREE`	Do not automatically add parentheses to the input operands of adder trees when switching from Setup to LEC mode. *This is the default.*
`-ADDERTREE`	Automatically adds parentheses to the input operands of adder trees when switching from Setup to LEC mode.
`-NOAUTO`	Do not automatically analyze datapath when switching from Setup to LEC mode. *This is the default.*
`-AUTO`	Automatically analyzes datapath when switching from Setup to LEC mode. This also performs additional carry-save adder (CSA) analysis.
`-NOMODULE`	Do not automatically apply analysis on the datapath modules in the Revised design netlist.
`-MODULE`	Automatically applies analysis on the datapath modules in the Revised design netlist. This option must be used with `-AUTO`.
`-RESOURCEFILE <filename>`	Specifies the resource filename to analyze the datapath modules.
`-ISOLATE_ABORT_MODULE`	Isolates the module that is aborted during module-based datapath (MDP) analysis. The module's gate-level netlist will be abstracted into RTL for comparison at the top module.
`-NOSHARE`	Do not apply the resource sharing technique. *This is the default.*
`-SHARE`	Analyzes the design for datapath resource sharing.
`-NOWORDLEVEL`	Do not apply word-level datapath analysis. *This is the default.*
`-WORDLEVEL`	Applies word-level datapath analysis. This helps analyze advanced word-level optimizations on designs with complex adder-tree clustering, product-of-sum, XOR-tree, or datapath with control logics or constants.
`-NOFLOWGRAPH`	Do not apply flow-graph based datapath analysis. *This is the default*.

`-FLOWGRAPH`	Applies flow-graph based datapath analysis. This helps analyze complex and advanced datapath clustering, product-of-sum multipliers or datapath with control logics or constants.
`-THREADS <integer>[,<integer>]`	Specifies the minimum and maximum number of threads for module-based datapath analysis. If only one number entered, this specifies both the minimum and maximum number of threads. Currently it is only effective with the `-MODULE` option. For example, '`-threads 2`' specifies two threads; '`-thread 2,4`' specifies a minimum of two threads, and a maximum of four threads.
`-GOLDEN`	Applies to the Golden design. *This is the default*.
`-REVISED`	Applies to the Revised design.
`-Verbose`	Provides additional information.

Examples

- The following command applies module-based datapath analysis followed by the operator-level datapath analysis when switching from Setup to LEC mode:

```
set datapath option -auto -module -verbose
set system mode lec
```

- The following command applies operator-level datapath analysis when switching from Setup to LEC mode:

```
set datapath option -auto
set system mode lec
```

- Portions of this command are global settings (specifically, anything not under -auto) and can override `ANALYZE DATAPATH` default settings. For example, if you set `SET DATAPATH OPTION -wordlevel`, Conformal performs word level analysis despite `ANALYZE DATAPATH`'s default setting of bit level analysis. In this case, to perform bit-level analysis with `ANALYZE DATAPATH`, you need to manually set `ANALYZE DATAPATH -nowordlevel`.

```
set datapath option -wordlevel
analyze datapath -nowordlevel
```

Related Commands

ANALYZE DATAPATH

ANALYZE MODULE

REPORT DATAPATH OPTION

REPORT MULTIPLIER OPTION

SET MULTIPLIER OPTION

SET FLATTEN MODEL

SET DIRECTIVE

```
SET DIrective
     <ON | OFf>
     [ [synthesis | <vendor_name>] <directives>]
     [-file <filename*>]
     (Setup Mode)
```

Specifies whether to enable or disable the effects of the specified synthesis directives when reading in a Verilog or VHDL file. If you enter this command and do not specify any directives, this command enables or disables *all of the* directive effects. *The system default enables all directives.* Thus, if you want Conformal to enable all directives, no action is necessary.

Execute this command before `READ LIBRARY` and `READ DESIGN`.

For each disabled directive used in the HDL source code, Conformal responds as follows:

- If the directive is supported but disabled, Conformal returns a message stating the directive is disabled.
- If the directive is unsupported and disabled, Conformal returns a message stating that the directive is unsupported.

See Conformal Directive Examples in the *Conformal Equivalence Checking User Guide* for short descriptions and examples of supported Conformal directives.

Tcl Command

```
set_directive
```

Parameters

`ON`	Enables the specified directives. (The initial system default enables *all* directives.)
`OFf`	Disables the specified directives. If you do not specify directives, all directives are disabled.
`synthesis`	Enables (or disable) the specified Synplicity synthesis directives.

`<vendor_name>`	Enables or disable the specified synthesis directives when they are used with the specified `<vendor_name>` prefix. Supported vendors are `ambit`, `cadence`, `conformal`, `pragma`, `quickturn`, `synopsys`, and `synthesis`.
`<directives>`	Enables or disables the specified synthesis directives. If you do not specify any directives, all directives are enabled or disabled, accordingly. See Supported Directives in the *Conformal Equivalence Checking User Guide* for the list of synthesis directives.
`-File <filename*>`	Enables (or disables) a list of directives that are specified in a RTL file. This accepts wildcards. for the list of supported directives.

Examples

- When you employ the `SET DIRECTIVE` command and you do not specify a directive, the command applies to all directives. In the following example, the objective is to enable only the `parallel_case` directive. To do so, first disable all directives, then enable the specified directive (`parallel_case`).

```
//disable all directives
set directive off
//enable parallel_case
set directive on parallel_case
```

- In the following examples, we have 2 RTL files: `test.v` and `test1.v`.
  - In the following command, the synthesis directive `parallel_case` is on (enabled) in file `test.v`:

    ```
    set directive on parallel_case -file test.v
    ```

  - In the following command, the synthesis directive `parallel_case` is on (enabled) in file `test.v` and `test1.v`:

    ```
    set directive on parallel_case -file *.v
    ```

Related Commands

READ DESIGN

READ LIBRARY

SET DOFILE ABORT

```
SET DOfile Abort
     <ON | OFf | Exit>
     (Setup / LEC Mode)
```

Specifies how Conformal handles the dofile when an error message occurs.

- If the dofile abort handling is set to On, the dofile terminates when an error message occurs. *This is the default.*
- If the dofile abort handling is set to Off, the dofile continues even if an error message occurs.
- If the dofile abort handling is set to Exit, the session exits when an error message occurs.

Tcl Command

```
set_dofile_abort
```

Parameters

`ON`	Terminates the dofile if an error message occurs. *This is the default.*
`OFf`	Continues the dofile even if an error message occurs.
`Exit`	Exits the session if an error message occurs.

Related Commands

BREAK

CONTINUE

DOFILE

SET DW DEFINITION

```
SET DW Definition
     <-USER_FIRST [-BBOX] | -USER_ONLY | -BUILTIN_only | -DW_MULt_div>
     (Setup Mode)
```

Specifies the DesignWare (DW*) modules' definition that the Conformal software will use. By default, the software specifies using the user-defined DW* modules first.

If you are using the built-in directory (where the DW* files are located), you can set the path with the following command:

```
setenv DW_DEFINE <path_name>
```

By default, the Conformal software uses the following path:

```
<install_dir>/share/cfm/lec/library/verilog/dw
```

Tcl Command

```
set_dw_definition
```

Parameters

`-USER_FIRST [-BBOX]`	With the `-USER_FIRST` option, all references to DW* modules are resolved by searching first from the user-defined DW* modules. The remaining, undefined DW* modules, are resolved again by searching from the built-in DW* modules. This is the software default. Some of the user-defined DW* modules might be blackboxed during RTL parsing due to an empty module body or unsynthesizable constructs. The `-BBOX` option tells the tool to continue to resolve the blackboxed DW* modules. When you use the `-USER_FIRST` and `-BBOX` options together, all references to DW* modules are resolved by searching first from the user-defined DW* modules, and then the remaining undefined DW* modules; all blackboxed, user-defined DW* modules are resolved again by searching from the built-in DW* modules.
`-USER_ONLY`	Use only user-defined DW* modules.

`-BUILTIN_only`	Use only built-in DW* modules and skip user-defined DW* modules.
`-DW_MULt_div`	Use only user-defined DW* modules, except `DW02_MULT` and `DW_DIV` modules.

Example

The following shows an example of the RTL code in a file `absval.v`.

```
module sample( A, ABSVAL );
    input [7 : 0] A;
    output [7 : 0] ABSVAL;
DW01_absval #(width) U1 ( .A(A), .ABSVAL(ABSVAL) ); endmodule
```

The command to read in the design is:

```
read design absval.v
```

The `DW01_absval.v` file is automatically read in from the built-in directory because there is no definition provided.

SET EQUIVALENT SUPPLY_SOURCES

```
SET EQuivalent Supply_sources
     <<-Golden <supply_source>* -Revised <supply_source>*>
     | -Delete <id>>
     | -List>
     (Setup Mode)
```

Note: This is a Conformal Low Power command and a 1801 feature.

This command allows users to specify equivalent supplies by giving a list of supply source points from the golden design and the revised design, and report the specified equivalent supplies. Each of the command issued will be given an ID number such that it can be identified to be reported in a list or deleted anytime. Note that the supply list from a design must be equivalent in such design through either physical connection, virtual connection, or 1801 power intent set equivalent command.

Tcl Command

```
set_equivalent_supply_sources
```

Parameters

`-Golden <`*`supply source list`*`>*`	Specifies a list of the supply sources from the Golden design
`-Revised <`*`supply source list`*`>*`	Specifies a list of the supply sources from the Revised design
`-Delete <`*`id`*`>`	Deletes the equivalent supplies with the specified id
`-List`	Reports a list of equivalent supplies that are specified by the command

Examples

Specify two different switch chain output supply sources, x1/VSW and x2/VSW, are equivalent to a revised supply source, VSW.

```
TCL_SETUP> set_equivalent_supply_sources -golden {x1/VSW x2/VSW2} -revised {VSW}
```

To report the specified equivalent, use -list, an ID number of each specified equivalent is displayed:

```
TCL_SETUP> set_equivalent_supply_sources -list
1:
  (G): x1/VSW x2/VSW
  (R): VSW
```

Related Commands

COMPARE POWER GRID

REPORT COMPARED POWER_GRID

SET EXIT CODE

```
SET EXit Code
     [-CLEAR]
     [-VERBOSE]
     [ | -INTERNAL_ERROR | -NOINTERNAL_ERROR]
     [ | -COMMAND_ERROR | -NOCOMMAND_ERROR]
     (Setup / LEC Mode)
```

Controls and displays the exit code for the Conformal session. This command is useful when running a complex flow, such as hierarchical comparison and iterative comparison.

Tcl Command

`set_exit_code`

Parameters

`-CLEAR`	Clears the exit code to only reflect most current running status.
`-VERBOSE`	Displays a table listing the status codes.
`-INTERNAL_ERROR`	Sets the internal error bit.
`-NOINTERNAL_ERROR`	Clears the internal error bit.
`-COMMAND_ERROR`	Sets the command error bit.
`-NOCOMMAND_ERROR`	Clears the command error bit.

Examples

- The following command displays current exit code:

  ```
  set exit code
  ```

- If a failing comparison was followed by a passing comparison (after fixing some constraints), bit 4 in the exit code is still non-zero. However, the following command clears the exit code and displays a table listing the status codes and decimal exit code:

  ```
  set exit code -clear -verbose
  ```

- The following command set command error bit to 1:

  ```
  set exit code -command_error
  ```

SET FLATTEN MODEL

```
SET FLatten Model
     [-AUTO_MODELING | -NOAUTO_MODELING]
     [-DFF_DC | -NODFF_DC]
     [-DFF_TO_DLAT_FEEDBACK | -NODFF_TO_DLAT_FEEDBACK]
     [-DFF_TO_DLAT_ZERO | -NODFF_TO_DLAT_ZERO]
     [-DFT_CONSTRAINT_CHECK]
     [-IN_TO_INOUT | -NOIN_TO_INOUT]
     [-LATCH_SR_TO_D | -NOLATCH_SR_TO_D]
     [-Map | -NOMap]
     [-NOALL_INV_SEQ_Merge | -ALL_INV_SEQ_Merge]
     [-NOALL_SEQ_Merge | -ALL_SEQ_Merge]
     [-NOBALANCED_MODELING | -BALANCED_MODELING]
     [-NOBBOX_MERGE | -BBOX_MERGE]
     [-NOBREAK_IOPAD_PATH | -BREAK_IOPAD_PATH]
     [-NOCUT_REMOVE_REDUNDANT | -CUT_REMOVE_REDUNDANT]
     [-CUT_REVISED_USING_GOLDEN_CUT | -NOCUT_REVISED_USING_GOLDEN_CUT]
     [-NODFF_TO_DLAT_ONE_CLOCK | -DFF_TO_DLAT_ONE_CLOCK]
     [-NOECO | -ECO]
     [-NOENABLE_ANALYZE_HIER_COMPARE | -ENABLE_ANALYZE_HIER_COMPARE]
     [-NOEXTRACT_CONSTRAINT_LIBERTY_BBOX | -EXTRACT_CONSTRAINT_LIBERTY_BBOX]
     [-NOGATED_Clock | -GATED_Clock]
     [-NOGLOBALREFERENCE | -GLOBALREFERENCE]
     [-NOHIER_SEQ_MERGE | -HIER_SEQ_MERGE]
     [-NOKEEP_IGnored_PO | -KEEP_IGnored_PO]
     [-NOLATCH_Fold
       | -LATCH_Fold
         [-FOLD_LIBRARY_FIRST | -FOLD_LIBRARY_ONLY | -FOLD_AUTO]
     ]
     [-NOLATCH_FOLD_GATED_CLOCK | -LATCH_FOLD_GATED_CLOCK]
     [-NOLATCH_FOLD_Master | -LATCH_FOLD_Master]
     [-NOLATCH_MERGE_PORT | -LATCH_MERGE_PORT]
     [-NOLATCH_Transparent | -LATCH_Transparent]
     [-NOLIB_CONTENTION_X | -LIB_CONTENTION_X]
     [-NOLIB_KEEP_UNDRIVEN | -LIB_KEEP_UNDRIVEN]
     [-NOLIB_SEQ_Redundant | -LIB_SEQ_Redundant]
     [-NOLIBRARY_PIN_VERIFICATION | -LIBRARY_PIN_VERIFICATION]
     [-NOLOOP_AS_DLAT | -LOOP_AS_DLAT]
     [-NOPin_keep | -PIN_keep]
     [-NOREDUNDANT_INPUT_DLAT | -REDUNDANT_INPUT_DLAT]
     [-NOSEQ_Constant | -SEQ_Constant]
     [-NOSEQ_DUPLICATION_VALIDATION | -SEQ_DUPLICATION_VALIDATION]
     [-NOSEQ_Merge | -SEQ_Merge]
     [-NOSEQ_Redundant | -SEQ_Redundant]
     [-NOSEQ_REDUNDANT_HIGH_EFFORT | -SEQ_REDUNDANT_HIGH_EFFORT]
     [-NOSEQ_SIMPLIFY_Clock | -SEQ_SIMPLIFY_Clock]
     [-NOSPECIFIED_SETUP_ONLY | -SPECIFIED_SETUP_ONLY]
     [-OUT_TO_INOUT | -NOOUT_TO_INOUT]
     [-OUTPUT_Z | -NOOUTPUT_Z]
```

```
[-PRESERVE_SEQ_CONSTANT | -NOPRESERVE_SEQ_CONSTANT]
[-SEQ_CONSTANT_ENDPOINT]
[-SEQ_CONSTANT_FEEDBACK | -NOSEQ_CONSTANT_FEEDBACK]
[-SEQ_CONSTANT_X_TO <NONE | 0 | 1>]
[-Z_CONVERSION <0 | 1| DC>]
[-BOTH | -GOLDEN | -REVISED]
[-VERBOSE]
(Setup Mode)
```

Specifies certain conditions for the flattened model. Refer to the arguments table for a complete list of options and their effects.

Use the `REPORT ENVIRONMENT` command to display the settings for the flattened model, or you can run this command without any options (in either Setup or LEC mode) to report a complete list of flattened modeling options.

Tcl Command

`set_flatten_model`

Parameters

`-AUTO_MODELING`	Enables selective modeling for designs that can be mapped mostly by name. This option applies only to sequential constants. *This is the default.*
`-NOAUTO_MODELING`	Disables the auto modeling feature.
`-DFF_DC`	Insert Don't Care gates where there is interaction between the clock, reset, set, or data signals. This is necessary to prevent false nonequivalences. *This is the default*.
`-NODFF_DC`	Do not insert Don't Care gates where there is interaction between the clock, reset, set, or data signals. Note: Using this option may cause false nonequivalences.

`-DFF_TO_DLAT_FEEDBACK`	Converts a DFF to a DLAT if the Q output feeds back to the D input. *This is the default.*
`-NODFF_TO_DLAT_FEEDBACK`	Do not convert a DFF to a DLAT if the Q output has feedbacks to the D input.
`-DFF_TO_DLAT_ZERO`	Converts a DFF to a DLAT if the clock port is zero. *This is the default.*
`-NODFF_TO_DLAT_ZERO`	Do not convert a DFF to a DLAT if the clock port is zero.
`-DFT_CONSTRAINT_CHECK`	Enable DFT constraint analysis in `set_system_mode` lec. It checks if a DFF is in test mode, and that means there may be missing some constraint.
`-IN_TO_INOUT`	Models submodule input ports as inout ports. In other words, all submodule input ports are treated as bi-directional. *This is the default*.
`-NOIN_TO_INOUT`	Do not model submodule input ports as inout ports.
`-LATCH_SR_TO_D`	Converts the SR-latch to a D-latch when the DFF/DLAT has a disabled clock but a non-constant asynchronous set/reset function. *This is the default*.
`-NOLATCH_SR_TO_D`	Do not convert an SR-latch to a D-latch when the DFF/DLAT has a disabled clock but a non-constant asynchronous set/reset function.
`-Map`	Does automatic key point mapping. *This is the default.*
`-NOMap`	Skips the automatic key point mapping when the system mode is changed from Setup to LEC.
`-NOALL_INV_SEQ_Merge`	Do not merge state elements that are functionally inverted. *This is the default.*
`-ALL_INV_SEQ_Merge`	Merges state elements that are functionally inverted.
`-NOALL_SEQ_Merge`	Do not merge sequential elements that are functionally equivalent. *This is the default.*
`-ALL_SEQ_Merge`	Merges sequential elements that are functionally equivalent. See Examples section.

`-NOBALANCED_MODELING`	Disables the selective modeling feature.
`-BALANCED_MODELING`	Enables selective modeling for designs that can be mapped by name. This option applies only to sequential constants. See also `AUTO_MODELING`.
`-NOBBOX_MERGE`	Do not perform automatic blackbox merging. *This is the default.*
`-BBOX_MERGE`	Performs automatic blackbox merging.
`-NOBREAK_IOPAD_PATH`	Do not partition the I/O pad path. *This is the default.*
`-BREAK_IOPAD_PATH`	Partitions the I/O pad path (the input side of the logic block and the output side of the logic block). By default, I/O pads combine the input side of the logic block to the output side of the logic block. Using this option, the input side of the logic block and the output side of the logic block are handled separately. This could be helpful for abort resolution.
`-NOCUT_REMOVE_REDUNDANT`	Do not remove redundant cuts. *This is the default.*
`-CUT_REMOVE_REDUNDANT`	Removes as many redundant cuts as possible. Use this option if you suspect that the software inserted more cuts than necessary.
`-CUT_REVISED_USING_GOLDEN_CUT`	Uses cuts in Golden to cut Revised. *This is the default*.
`-NOCUT_REVISED_USING_GOLDEN_CUT`	Do not use cuts in Golden to cut Revised.
`-NODFF_TO_DLAT_ONE_CLOCK`	Do not convert a DFF to a DLAT if the clock port value is 1. *This is the default.*
`-DFF_TO_DLAT_ONE_CLOCK`	Converts a DFF to a DLAT if the clock port value is 1. **Note:** The clock to DLAT is tied to 0.

`-NOECO`	Do not preserve extra circuit information for model flattening. *This is the default.*
`-ECO`	Preserves extra circuit information during model flattening for subsequent `ANALYZE ECO` command runs.
`-NOENABLE_ANALYZE_HIER_COMPARE`	Disables the `ANALYZE HIER_COMPARE` command. *This is the default*.
`-ENABLE_ANALYZE_HIER_COMPARE`	Enables the `ANALYZE HIER_COMPARE` command in LEC mode (used for hierarchical dofile generation).
`-NOEXTRACT_CONSTRAINT_LIBERTY_BBOX`	Do not extract constraints for blackboxes coming from liberty cell. *This is the default*.
`-EXTRACT_CONSTRAINT_LIBERTY_BBOX`	Extract constraints for blackboxes from Liberty cells.
`-NOGATED_Clock`	Do not remodel gated-clock sequential instances. *This is the default.*
`-GATED_Clock`	Remodels the gated-clock logic of the clock port of a DFF. If the clock pin cannot be automatically determined, use the `ADD CLOCK` command to define the clock pin.
`-NOGLOBALREFERENCE`	Do not resolve globally-referenced variables during flattening. *This is the default*.
`-GLOBALREFERENCE`	Resolves globally-referenced variables during flattening.
`-NOHIER_SEQ_MERGE`	Disables user-specified sequential merge in hierarchical doilfe generation. *This is the default*.

`-HIER_SEQ_MERGE`	Enables user-specified sequential merge in hierarchical dofile generation. This option automatically enables balanced constraint extraction. The sequential merge will be proven after flattening and before constraint extraction. If non-equivalences are found or the representative's fan-in cone has don't cares, Conformal reflattens the designs and skips those sequential merges.
`-NOKEEP_IGnored_PO`	Do not retain the ignored primary outputs (added with the `ADD IGNORED OUTPUTS` command) in the flattened netlist. *This is the default.*
`-KEEP_IGnored_PO`	Retains the ignored primary outputs (added with the `ADD IGNORED OUTPUTS` command) in the flattened netlist. These ignored primary outputs appear as unreachable unmapped points in the design.
`-NOLatch_fold`	Do *not* fold a master-slave latch into a D flip-flop. *This is the default.*
`-Latch_fold`	Folds a master-slave latch into a D flip-flop.
`-FOLD_LIBRARY_FIRST`	Folds latches in libraries first. *This is the default* when -Latch_fold option is specified.
`-FOLD_LIBRARY_ONLY`	Only folds latches in libraries when -Latch_fold option is specified.
`-FOLD_AUTO`	Folds latches without any priority and restriction with respect to libraries when -Latch_fold option is specified.
`-NOLATCH_FOLD_GATED_CLOCK`	Do not fold a master-slave latch into a D flip-flop, when the clock port of the master/slave latch contains gated-clock logic. *This is the default.*
`-LATCH_FOLD_GATED_CLOCK`	Folds a master-slave latch into a D flip-flop, when the clock port of the master/slave latch contains gated-clock logic. This option must be used together with the `-Latch_fold` option to have any effect.
`-NOLATCH_FOLD_Master`	Do not convert two latches in an LSSD format into a DFF gate when the reset signal is connected only to the master. *This is the default.*

`-LATCH_FOLD_Master`	Converts two latches in an LSSD format into a DFF gate when the reset signal is connected only to the master.
`-NOLATCH_Merge_port`	Do not collapse multi-port latches into a single-port latch. *This is the default.*
`-LATCH_Merge_port`	Collapses multi-port latches into a single-port latch.
`-NOLATCH_Transparent`	Do not treat latches that are always enabled as transparent. *This is the default.*
`-LATCH_Transparent`	Converts D-Latches into buffers if the clock ports of the D-latches are always enabled.
`-NOLIB_CONTENTION_X`	Do not model the Liberty contention condition. LEC will ignore contention condition modeling in the flattened netlist. *This is the default*.
`-LIB_CONTENTION_X`	Models the Liberty contention condition as the X assignment. LEC will ignore contention conditions by default. User can further use `SET X CONVERSION` to switch the model of X.
`-NOLIB_KEEP_UNDRIVEN`	Do not keep undriven internal and output signals inside the library module as high-impedance (always driven by Z). *This is the default*.
`-LIB_KEEP_UNDRIVEN`	Keep undriven internal and output signals inside the library module as high-impedance (always driven by Z).
`-NOLIB_SEQ_Redundant`	Retains redundant fan-out gates from DFFs and DLATs in the Library. *This is the default.*
`-LIB_SEQ_Redundant`	Removes redundant fan-out gates from DFFs and DLATs in the Library.
`-NOLIBRARY_PIN_VERIFICATION`	Do not compare unconnected library module input pins (output floating) and output pins (input undriven). *This is the default.*
`-LIBRARY_PIN_VERIFICATION`	Enables comparisons of unconnected library module input pins (output floating) and output pins (input undriven). This option applies only to library modules that are referenced by the design files.

`-NOLOOP_AS_DLAT`	Do not use a DLAT to model a combinational loop. *This is the default.*
`-LOOP_AS_DLAT`	Uses a DLAT to model a combinational loop.
`-NOPin_keep`	In an effort to reduce memory use, Conformal does *not* keep certain gate pins. *This is the default.*
`-PIN_keep`	Keeps all gate pin information for gate reporting. Use this option when reporting gate information at the design level. *It will increase memory use.*
`-NOREDUNDANT_INPUT_DLAT`	Do not convert D-latches into buffers, which are redundant to the fanout DFFs or D-latches. *This is the default.*
`-REDUNDANT_INPUT_DLAT`	Converts D-Latches into buffers when D-latches are in the data cone of DFFs or other D-latches, and D-latches are redundant with respect to DFFs or fanout D-latches.
`-NOSEQ_Constant`	Do not propagate constant data through latches and registers. *This is the default.*
`-SEQ_Constant`	Propagates constant data through latches and registers.
`-NOSEQ_DUPLICATION_VALIDATION`	Disables the automatic modeling for sequential duplication validation. *This is the default*.
`-SEQ_DUPLICATION_VALIDATION`	Enables the automatic modeling for sequential duplication validation
`-NOSEQ_Merge`	Do not merge sequential elements in the clock cone of a DFF or D-latch. *This is the default.*
`-SEQ_Merge`	Takes common groups of sequential elements that are in the clock and set/reset cones of other sequential elements, and merges them into one sequential element in the clock cone of a DFF or D-latch. See Examples section.
`-NOSEQ_Redundant`	Do not remove redundant fan-out gates from DFFs and DLATs. *This is the default.*
`-SEQ_Redundant`	Removes redundant fan-out gates from DFFs and D-Latches.

`-NOSEQ_REDUNDANT_HIGH_EFFORT`	Specifies the standard effort level for sequential redundant. *This is the default*.
`-SEQ_REDUNDANT_HIGH_EFFORT`	Specifies a high effort level for sequential redundant. The high effort level can achieve better analysis, but also increases analysis time and memory usage.
`-NOSEQ_SIMPLIFY_Clock`	Do not fold master and slave latches when there is a redundant interaction between the clock and reset signals. *This is the default.*
`-SEQ_SIMPLIFY_Clock`	Folds master and slave latches when there is a redundant interaction between the clock and reset signals.
`-NOSPECIFIED_SETUP_ONLY`	Specifies that LEC performs sequential constant and sequential merge modeling analysis for all the keypoints. Also disables mapping from verification information. *This is the default*.
`-SPECIFIED_SETUP_ONLY`	Specifies that LEC only performs sequential constant and sequential merge modeling for the keypoints specified in the setup information (`READ SETUP INFORMATION`) or verification information (`SET VERIFICATION INFORMATION`). It also enables mapping from verification information.
`-OUT_TO_INOUT`	Models submodule output ports as inout ports. In other words, all submodule output ports are treated as bi-directional. *This is the default*.
`-NOOUT_TO_INOUT`	Do not model submodule ouput ports as inout ports.
`-OUTPUT_Z`	Checks for tri-state conditions at top-level output ports and inputs to blackboxes. *This is the default*.
`-NOOUTPUT_Z`	Do not check for tri-state conditions at top-level output ports and inputs to blackboxes.
`-PRESERVE_SEQ_CONSTANT`	Preserves sequential constant DFF/DLAT in sequential constant modeling. A constant gate is added at the output of the DFF/DLAT. *This is the default*.

`-NOPRESERVE_SEQ_CONSTANT`	Replace the sequential constant DFF/DLAT with a constant gate in sequential constant modeling. Note: *This is the default* in versions earlier than 14.10-s180.
`-SEQ_CONSTANT_ENDPOINT`	The option should be used with the option `-seq_constant`. It separates the sequential constant modeling into two steps. First, propagates constant data through latches and registers whose fanout cone contains other sequential elements. Then, propagates constant data through the rest of latches and registers. After that, turns off the sequential constant modeling in `analyze_setup`.
`-SEQ_CONSTANT_FEEDBACK`	Remodels registers that also have feedback to constants. Use this option with `-seq_constant`. *This is the default.*
`-NOSEQ_CONSTANT_FEEDBACK`	Do not remodel registers that also have feedback to constants. Use this option with `-seq_constant`.
`-SEQ_CONSTANT_X_TO`	Optimizes a flop to a constant value (either zero or one) when the flop is always in a *don't care* (`X`) state. Use this with the `-seq_constant` switch. NONE: Do not optimize. *This is the default*. 0: Optimizes a flop to a constant zero. 1: Optimizes a flop to a constant one.
`-Z_CONVERSION`	Converts Z(s) to the specified value. This option is used for memory abstraction. 0: Converts Z(s) to 0(s). 1: Converts Z(s) to 1(s). DC: Converts Z(s) to don’t care(s).
`-GOLDEN`	Applies settings to Golden design. When neither -Golden or -Revised are specified, settings are applied to both the Golden and Revised designs.

`-REVISED`	Applies settings to Revised design. When neither -Golden or -Revised are specified, settings are applied to both the Golden and Revised designs.
`-VERBOSE`	Display all verbose messages.

Examples

Sequential Merge

Given the following example, if you used the `-SEQ_MERGE` option, LEC merges common groups of sequential elements that are in the clock and set/reset cones of other sequential elements and prints out:

```
// (F19) Merged 1 DFF/DLAT(s) in clock cones due to functional equivalence
// (F19.1) Merged 1 DFF/DLAT(s) in set/reset cones due to functional equivalence
```

While with `-ALL_SEQ_MERGE` merges all the sequential elements that are functionally equivalent and prints out:

```
// (F20) Merged 3 DFF/DLAT(s) due to functional equivalence
```

Golden:

```
module top (input A, input rst, input clk, input clk2, output B, output C);
reg clk_reg0, clk_reg1;
reg rst_reg0, rst_reg1;
reg a_reg, b_reg;

always @(posedge clk) begin
  clk_reg0 <= clk2;
  clk_reg1 <= clk2;
  rst_reg0 <= rst;
  rst_reg1 <= rst;
end
always @(posedge clk_reg0 or posedge rst_reg0) begin
  if (rst_reg0)
   a_reg <= 1'b0;
  else
   a_reg <= A;
end

always @(posedge clk_reg1 or posedge rst_reg1) begin
  if (rst_reg1)
   b_reg <= 1'b0;
  else
   b_reg <= A;
end

assign B = a_reg;
assign C = b_reg;
endmodule
```

Revised:

```
module top (input A, input rst, input clk, input clk2, output B, output C);
reg clk_reg0;
reg rst_reg0;
reg a_reg;

always @(posedge clk) begin
  clk_reg0 <= clk2;
  rst_reg0 <= rst;
end

always @(posedge clk_reg0 or posedge rst_reg0) begin
 if (rst_reg0)
   a_reg <= 1'b0;
 else
   a_reg <= A;
end
assign B = a_reg;
assign C = a_reg;

endmodule
```

Related Commands

ANALYZE HIER_COMPARE

READ MAPPED POINTS

REMODEL

REPORT ENVIRONMENT

REPORT MESSAGES

SET GATE REPORT

SET FPGA TECHNOLOGY

```
SET FPga Technology
     [NONE | VIRTEX | VIRTEX2]
     (Setup Mode)
```

Note: This command is an FPGA command.

Turns on FPGA-specific processing. It is included in the `fpgaR2G.do` dofile.

Tcl Command

```
set_fpga_technology
```

Parameters

NONE	Do not turn on any FPGA-specific processing. *This is the default.*
VIRTEX	Turns on the Xilinx Virtex processing.
VIRTEX2	Turns on Virtex2 processing.

SET GATE REPORT

```
SET GAte Report
     [-DYNamic | -NODYNamic]
     [-FUNction | -STRucture]
     [-NOUSE_LIBRARY_PINNAME | -USE_LIBRARY_PINNAME]
     [-PRImitive | -DESign]
     (Setup / LEC Mode)
```

Specifies the detail level of gate reports in the Conformal gate information display. Gate report features include the following:

- Returns information at the design or primitive level
- Displays the dynamic constraints
- Displays the fanin cone of the zero/one gates

By default, this command reports gate information at the primitive level, displays the dynamic constraints, and does not display the fanin cone of the zero or one gates.

Use the `REPORT ENVIRONMENT` command to display the gate report level settings.

Note: If the gate report is set to Design, you must use the `SET FLATTEN MODEL` command with the `-pin_keep` option in the Setup system mode. The gate information is reported in the LEC system mode.

Tcl Command

```
set_gate_report
```

Parameters

`-DYNamic`	Displays the dynamic constraints in the gate report information. *This is the default.*
`-NODYNamic`	Do not display the dynamic constraints in the gate report information.
`-FUNction`	Do not display the fanin cone of the zero/one gates in the gate report information. *This is the default.*
`-STRucture`	Displays the fanin cone of the zero/one gates in the gate report information.

`-NOUSE_LIBRARY_PINNAME`	Do not use library pin name in gate report. *This is the default*.
`-USE_LIBRARY_PINNAME`	Uses library pin name in gate report if the gate comes from library.
`-PRImitive`	Displays the gate report information at the primitive level. *This is the default.*
`-DESign`	Displays the gate report information at the design level.

Related Commands

REPORT ENVIRONMENT

REPORT GATE

SET FLATTEN MODEL

SET GUI

```
SET GUi
     [ON [-MAPping]
     | OFf]
     (Setup / LEC Mode)
```

Switches Conformal to the GUI mode from the non-GUI mode or to the non-GUI mode from the GUI mode.

Tcl Command

```
set_gui
```

Parameters

`ON`	Switches to the GUI mode. *This option is the initial system default.*
`-MAPping`	Brings up the Mapping Manager automatically.
`OFf`	Switches to the non-GUI mode.

SET HDL OPTIONS

```
SET HDl Options
     [-AUTOPkgsearch <ON | OFF | PREFIX <string>| SUFFIX <string>>]
     [-BBOX_MODULE_WITH_NO_PI_USED <OFF | ON>]
     [-CONVERT_ONEBIT_VECTOR_TO_SCALAR <OFF | ON>]
     [-CONTINUOUSASSIGNment <BIdirectional | UNIdirectional>]
     [-COPY_VERILOG_TO_LIBERTY
         <USE_LIBERTY_FUNCTION | USE_VERILOG_ONLY_IF_LIBERTY_EMPTY |
          USE_VERILOG_FUNCTION_ALWAYS>]
     [-DEFINE <MACRO NAME>
     [-DFF_ASYNC_HOLD <OFF | ON>]
     [-ERROR_OUT_POSITIONAL_ASSOCIATION_ON_LIBERTY_CELL <OFF | ON>]
     [-FIX_PIN_DIRECTION <OFF | ON>]
     [-FV_TRIMINDex <OFF | ON>]
     [-FULLPATH <OFF | ON>]
     [-HIEREF_TO_PORT_CONN <OFF | ON>]
     [-INCLUDE_SRC_DIR <ON |OFF>]
     [-LEGACY_DC_SCRIPT <OFF | ON>]
     [-LIBERTY_MAP_FILE <name>]
     [-LRM_Compliance <INACTIVE|ON|OFF>]
     [-MAX_FOR_LOOP_SIZE <integer_value>]
     [-MAX_PARSE_STACK <integer_value>]
     [-MAXMEM_address_space <threshold>]
     [-MERGE_PHYsical_cell <AUTO|OFF|USER|AUTO_USER>]
     [-MULTILINERead <OFF | ON>]
     [-NO_NEGATIVE_INDEX_PORT <OFF | ZEROBASED | NOOVERLAPPED>]
     [-NOCONST_PORT_EXTend | -CONST_PORT_EXTend]
     [-NOPARSE_FILE_CMD | -PARSE_FILE_CMD <command>]
     [-OUT_LIBERTY_DIR <path>]
     [-OUT_LIBERTY_FILE <name>]
     [-PHYSICAL_CELL_EXTRACT <LIBERTY_ATTRIBUTE|ATTRIBUTE_AND_STRUCTURE>]
     [-PORT_SIZE_limit <size>]
     [-PORTMISmatch <UNconnect | EXTend>]
     [-PRIMITIVE_INPUT_Conversion <LSB | LOGIC>]
     [-READ_TRANSLATE_MSFF <ON | OFF>]
     [-REMOVE_CELL <cellnames*>]
     [-STANDARD_LIBERTY_FOR_BIAS_PIN_MAPPING <OFF | ON>]
     [-SV_2STATE_XZ_TO_0 <ON | OFF>]
     [-SYNTHESIS_executable <syn_exe>]
     [-UDP_IGN_VDD <module_name> <vdd_pin_name>]
     [-UNREACHABLE_STMT_CHECK <OFF | ON>]
     [-UNSIGNED_CONVERSION_OVERFLOW <OFF | ON>]
     [-UNSIGNED_EXPRESSION_OVERFLOW <OFF | ON>]
     [-USE_LIBRARY_FIRST <OFF | ON>]
     [-USER_DEFINED_PHYSICAL_CELL <cell_name*>]
     [-V_TO_VD <OFF | ON>]
     [-VARSIZE_limit <integer_value>]
     [-VERILOG_INCLUDE_DIR <string>]
     [-VERILOG_OUTOFBOUNDRead <PARTIAL_X | ALL_X | PARTIAL_0 | PARTIAL_1>]
```

```
[-VERILOG_OUTOFBOUNDWrite <Noeffect | X>]
[-VERILOG_TRIMINDex <OFF | ON>]
[-VERILOG_XL_LIBORDER <OFF | ON>]
[-VHDL_ELAB_UNSPECIFIED_CONFIG <ON | OFF>]
[-VHDL_OUTOFBOUNDWrite <X | Noeffect>]
[-VHDL_OUTOFBOUNDRead <ALL_X | PARTIAL_X | PARTIAL_0 | PARTIAL_1>]
[-VHDL_TRIMINDex <OFF | ON>]
[-WAND_GATE [FALSE | TRUE]]
[-WOR_GATE [FALSE | TRUE]]
[-ZERO_REPLICATE_AS_ZERO <ON | OFF>]
(Setup Mode)
```

Controls the interpretation of some RTL semantics.

This command sets global attributes that control the behavior of subsequent `READ LIBRARY`, `READ DESIGN`, and `ELABORATE DESIGN` commands. Libraries and designs that have been successfully read in and elaborated previously will not be affected by newly issued `SET HDL OPTIONS` commands.

Tip: Use `SET HDL OPTIONS` without any options to display the current settings for `SET HDL OPTIONS`.

Tcl Command

```
set_hdl_options
```

Parameters

`-AUTOPkgsearch <ON | OFF | PREFIX <`*`string`*`> | SUFFIX <`*`string`*`>>`

`ON`: *This is the default*. Searches the work directory for referred packages or entities.

`OFF`: Disables the automatic searching.

`PREFIX <string>`: Specifies that you want to search for packages/entities with the specified prefix. Separate multiple values with a comma (spaces are ignored).

`SUFFIX <string>`: Specifies that you want to search for packages/entities with the specified suffix. Separate multiple values with a comma (spaces are ignored).

When a prefix or suffix is not specified, the VHDL parser searches the working directory for `name.vhdl/ name.vhd`.

When a prefix or suffix is specified, the tool first searches for the packages with the specified prefix or suffix. If there are no results, the tool resumes default searching. If there are multiple prefix/suffix values available, the tool searches in the order specified in the string.

Note: You cannot specify both a prefix and a suffix.

`-BBOX_MODULE_WITH_NO_PI_USED`

`ON` specifies that Verilog modules or VHDL entities will be treated as a blackbox if none of the inputs are used in the module entity. This option is OFF by default.

`-CONVERT_ONEBIT_VECTOR_TO_SCALAR <OFF | ON>`

Default is OFF. When set to ON, this option tells Conformal to treat all single-bit vectors as scalar objects. For example, both dd and qq in following module t1 declaration are single-bit vectors. With this option, they are treated as scalar objects:

```
module t1(clk, dd, qq);
input [0:0] dd; // changes to "input dd;"
input clk;output reg [1:1] qq; //changes to "output
reg qq;"
always @ (posedge clk) begin qq <= dd;
end
endmodule
```

`-CONST_PORT_EXTend`	Perform sign-extention to port connections connected to a constant number, regardless of the PORTMISMATCH setting (see also -NOCONST_PORT_EXTend)

`-CONTINUOUSASSIGNment <BIdirectional | UNIdirectional>`

Specifies that continuous assignment in the design should be interpreted as uni-directional or bi-directional assignment.

`-COPY_VERILOG_TO_LIBERTY` Specifies the source from which the cell's logic function is derived.

`USE_LIBERTY_FUNCTION`: *This is the default*. Tool always takes the logic function from the Liberty library model.

`USE_VERILOG_ONLY_IF_LIBERTY_EMPTY`: If the cell's Liberty library model is not available, the tool looks for logic information in the Verilog model.

`USE_VERILOG_FUNCTION_ALWAYS`: Tool uses function information from the Verilog model, regardless of whether the Liberty model is available. However, PG pins (power/ground/bias) that are defined in Liberty but not in Verilog will be automatically added to stay consistent with the Liberty definition. To enable this feature, users need to read the liberty library into the library space first, then read the verilog library into library space with the -append option. For example:

```
set hdl option -copy_verilog_to_liberty \
  user_verilog_function_always
read library -liberty std_lib.libread library -
verily std_lib.v -append
```

If users read the verilog library into the design space, the logic function information of the verilog library will not be merged with the liberty library.

`-DEFINE <MACRO NAME>`	Helps the user define a macro which can work in each `READ DESIGN`. For example:
	
	Users can use `SET HDL OPTION -DEFINE AAA` instead of
	
	`read design -sv -define AAA 1.sv`
	`read design -sv -define AAA 2.sv`
	`read design -sv -define AAA 3.sv`
	
	On the other hand, command `RESET` will clear these defined macros, and command `REPORT HDL OPTION` will print the defined macros.

`-DFF_ASYNC_HOLD`	Controls how an asynchronous conditional self-assignment is implemented for an RTL design. By default, this option is OFF and asynchronous pins are used to implement the gate-level design. In some cases, a synthesizer may use asynchronous pins for implementation instead and cause RTL and gate-level designs non-equivalent. Setting this option ON may help equivalence checking for such cases. This setting applies to the entire RTL parsing.
`-ERROR_OUT_POSITIONAL_ASSOCIATION_ON_LIBERTY_CELL <OFF | ON>`	
	`OFF`: *This is the default*. `ON`: When specified, the tool checks whether the design includes one or more module instances that uses port positional association of Liberty cells (rule check HRC3.8a).
`-FIX_PIN_DIRECTION <OFF | ON>`	This changes the pin direction to the bi-directional (inout) type when an output port of a module is not driven by any logic inside the module, or when a input port of a module is driven by some logic inside the module.
`-FULLPATH <OFF | ON>`	
	Records the file name's absolute path name; when files are read in, the file source is stored in its absolute path (opposed to a relative path or a path added by the `ADD SEARCH PATH` command). Default is `OFF`.33
`-FV_TRIMINDex <OFF | ON>`	

Default is OFF. When set to ON, this option tells Conformal to use FV implementation information as a guidance to perform index range truncation.

`-HIEREF_TO_PORT_CONN <OFF | ON>`

When set to ON, LEC will create port(s) for hierarchical variable reference's corresponding instance module(s) and replace the hierarchical variable reference with an instance port connection plus internal net connection. Default is OFF.

`-INCLUDE_SRC_DIR <ON |OFF>`

For Verilog `` `include `` directives, LEC searches and reads the file in source file's relative directory before any paths added by `ADD SEARCH PATH`. Default is `ON`.

`-LEGACY_DC_SCRIPT <OFF | ON>`

Controls the contents of the non-synthesizable DW script.

OFF : Include verification priority setting. Do not include ungroup option setting. *This is the default*.

ON : Include ungroup option setting. Do not include verification priority setting.

`-LIBERTY_MAP_FILE <`*`name`*`>`

Specifies the name and location of the file that contains the information of the original Liberty file name and the new simplified file name. If the option is not specified, no mapping file will be generated. If the option is specified without a value, the tool will report that this option is not fully specified.

For more information on this option, refer to the web interface article titled *Liberty File Optimization for Verification.*

`-LRM_Compliance <INACTIVE|ON|OFF>`

	Specifies how to treat LRM non-compliance violations when reading in the design: INACTIVE: Do not check. *This is the default*. ON: Report as errors OFF: Report as warnings
`-MAX_FOR_LOOP_SIZE`	Sets the maximum number of iterations when unfolding a loop construct. By default, the maximum is 8192 iterations.

`-MAX_PARSE_STACK <integer_value>`

Specifies the maximum parser stack size for a variable. By default, the maximum parser stack size is 100000.

For example, to increase the maximum parser stack size:

```
set hdl option -MAX_PARSE_STACK 200000
```

`-MAXMEM_address_space <threshold>`

Specifies the threshold of the maximum memory-address space. Modules whose RAM size is greater than or equal to the threshold will be blackboxed.

The default threshold value is 65536.

For example, to specify a maximum memory address space of 1024:

```
set hdl option -MAXMEM_address_space 1024
```

`-MERGE_PHYsical_cell <AUTO | OFF | USER | AUTO_USER>`

Controls physical cell instance merging behavior.

AUTO: Performs instance merging on physical cells that are recognized by Conformal. *This is the default.*

OFF: Do not perform physical cell instances merging.

USER: Performs instance merging on user-specified cells only.

AUTO_USER: Performs instance merging on user-specified and tool recognized physical cells.

`-MULTILINERead <OFF | ON>`

Enables support for multi-lined `READ DESIGN` commands. Default is `OFF`.

For example, when this option is enabled, the following is supported:

```
read design -vhdl -mapfile lib1 ent.vhdl -
noelaborate
read design -vhdl -mapfile lib1 arch.vhdl -
noelaborate
```

Note: Do not place other commands between the `READ DESIGN` commands.

`-NO_NEGATIVE_INDEX_PORT <OFF | ZEROBASED | NOOVERLAPPED>`

Changes the index data for negative ports.

`OFF`: *This is the default*. Do not change the index data.

`ZEROBASED`: For an array index of [-9:-1] or [-4:4], shifts the negative range to a positive range that starts at zero.

`NOOVERLAPPED`: For an array index of [-9:-1] (where both MSB and LSB are less than zero), shifts the negative range to a positive range that starts at zero. For an array index of [-4:4] (where MSB is less than zero, and LSB is greater than zero, or MSB is greater than zero and LSB is less than zero), changes the range by shifting the negative index to a positive index +1, and increases the positive index with size data.

`-NOCONST_PORT_EXTend` Do not sign-extend port connections connected to a constant number. *This is the default.* (see also -CONST_PORT_EXTend)

`-NOPARSE_FILE_CMD` Do not run any shell commands before parsing the file. *This is the default*.

`-OUT_LIBERTY_DIR <`*`path`*`>`

Specifies the directory where the simplified Liberty files will be placed. If the specified directory does not exist, it will be created. If this option is not specified, no action will be taken. If the option is specified without a value, the default output directory is the current working directory.

Default is the working directory.

For more information on this option, refer to the web interface article titled *Liberty File Optimization for Verification.*

`-OUT_LIBERTY_FILE <name>`

Specifies the prefix for the filenames. Existing files will be replaced. Each file will be assigned a name as `<prefix>_number` where number starts at 00001, incremented by 1 for each new file up to 99,999. The library name is also modified and will have the same value as the new file name. If this option is not specified, the original filename will be used. If the option is specified without a value, the tool will report that this option is not fully specified.

For more information on this option, refer to the web interface article titled *Liberty File Optimization for Verification.*

`-PHYSICAL_CELL_EXTRACT <LIBERTY_ATTRIBUTE | ATTRIBUTE_AND_STRUCTURE>`

Controls the physical cell recognitions.

LIBERTY_ATTRIBUTE: Extracts the cell defined by Liberty physical cell attributes. *This is the default*.

ATTRIBUTE_AND_STRUCTURE: Extracts the cell defined by Liberty physical cell attributes or by Conformal physical cell structure rules.

`-PORT_SIZE_limit <size>`

Specifies the maximum size for a port. By default, the maximum port size is 524288.

For example, to increase the maximum port size:

```
set hdl option -PORT_SIZE_limit 1048576
```

`-PARSE_FILE_CMD <`*`command`*`>`

Runs a shell command or executable file after printing out the "`Parsing file ...`" message and before Conformal parses the file during the `READ LIBRARY` or `READ DESIGN` commands.

Each current parsing file name will be stored in an environment variable `CFM_READ_FILE` before invoking the command.

`-PORTMISmatch <UNconnect | EXTend>`

When the port connection widths between the module and an instantiation do not match, this option controls how the tool models the extra bits.

`UNconnect` leaves the extra bits unconnected. *This is the default*.

`EXTend` applies zero- or sign-extending to the value, depending on the expression signedness. See Examples section.

`-PRIMITIVE_INPUT_Conversion <LSB | LOGIC>`

Specifies how the software's Verilog parser handles multiple-bit expressions.

`LSB` takes the least significant bit from the multiple-bit vector expression. *This is the default for the Verilog parser*.

`LOGIC` treats the entire multiple-bit expression as a logic true/false value. For example,

```
wire [0:5] net0;
or  I00 (out, net0[0:5]);


net0[0:5] used in I00 is treated as net0[5] by
default, and it is treated as a logic true/false
value when using the LOGIC option.
```

`-READ_TRANSLATE_MSFF <ON | OFF>`

Models master-slave DFF of Liberty flip-flop cells that have `clock_on` and `clock_on_also` attributes.

`ON` translates master-slave flip-flops (MSFF) to master slave latches. With this option, two D latches are used to model the master-slave DFF. *This is the command default*.

`OFF` keeps the master slave flip-flops. With this option, two flip-flops are used to model the master-slave DFF.

`-REMOVE_CELL <cellnames*>`

Specifies a list of library cell instantiations to be excluded from the design. This option should be specified before the `READ DESIGN` command. Wildcards are accepted.

`-STANDARD_LIBERTY_FOR_BIAS_PIN_MAPPING <OFF | ON>`

Enables support for standard liberty format conversion for custom liberty attributes. Default is OFF.

`-SV_2STATE_XZ_TO_0 <ON | OFF>`

On assignment of value X or Z to a 2-state variable, convert it to 0. Default is ON.

`-SYNTHESIS_executable <syn_exe>`

Specifies the path to the Design Compiler synthesis executable. The `WRITE BLACKBOX WRAPPER -auto_substitute` command needs this to perform automatic blackbox substitution.

For example:

```
set hdl option -synthesis_executable \
/grid/DC_INSTALL_DIR/latest/bin/dc_shell
-t
```

`-UDP_IGN_VDD <module_name> <vdd_pin_name>`

In UDP modeling, power/ground ports are not supported; therefore, having this information in your UDP state table can cause unexpected results.

Use this option to specify the module and pin names of power/ground ports that are defined in the UDP. The tool will ignore the information for these ports in the UDP state table.

`-UNREACHABLE_STMT_CHECK <OFF | ON>`

When set to ON, the tool checks whether the design statement is unreachable. If it is, RTL checking will be skipped for that statement. Note that enabling this option will increase elaboration time.

Default is OFF.

`-UNSIGNED_CONVERSION_OVERFLOW <OFF | ON>`

`OFF`: *This is the default*.

`ON`: When specified, the tool will take up to 32 bits of UNSIGNED operands to signed 32-bit integer.

See example below.

`-UNSIGNED_EXPRESSION_OVERFLOW <OFF | ON>`

`OFF`: *This is the default*.

`ON`: When specified, unsigned operands of up to 32 bits can overflow (or wrap) to signed 32-bit integers.

See example below.

`-USE_LIBRARY_FIRST`

`OFF`: Design modules have highest priority during synthesis. *This is the default*. At low power native-1801 flow, a module extracted from Liberty as a macro cell still has the highest priority even the OFF value is set.

`ON`: Specifies that library cells have highest priority when being selected during synthesis.

`-USER_DEFINED_PHYSICAL_CELL <cell_name*>`

Specifies the list of user-defined physical cell names.

`-V_TO_VD <OFF | ON>`	OFF: Logic compiled with `-v` option will be categorized into the library space and logic compiled with `-vd` option will be categorized into the design space. *This is the default*. ON: Logic compiled with `-v` and `-vd` options will be categorized into the design space..
`-VARSIZE_limit <integer_value>`	
	Specifies the maximum size for a variable. By default, the maximum variable size is 4194304. For example, to increase the maximum variable size: `set hdl option -VARSIZE_limit 134217728`
`-VERILOG_INCLUDE_DIR <string>`	

Specifies the order for searching paths when opening a Verilog `include file. The argument is a string comprised of the following keywords in quotes and separated by a colon ":"

`cwd`: current working directory "."

`incdir`: directories specified by +incdir option in the verilog command file

`src`: directory that contains the file that specifies the `include directive

`sep`: Search paths added by the ADD SEarch Path command

`yd`: directories specified by -yd in the Verilog command file

`y`: directories specified by -y in the Verilog command file

When `-INCLUDE_SRC_DIR` is `OFF`, the src argument will not have any effect..

Default string is `cwd:incdir:src:sep:yd:y`.

For example, if you want to search for include files only in current working directory and all directories specified by the `ADD SEARCH PATH` command, use the command as follows:

```
set hdl option  -verilog_include_dir "cwd:sep"
```

Other directories added by +incdir, -yd, or the -y Verilog command file option are all ignored.

`-VERILOG_OUTOFBOUNDRead <PARTIAL_X | ALL_X | PARTIAL_0 | PARTIAL_1>`

Controls the interpretation of Verilog bit (or part)-select of vector typed variable/signal when index is out of the defined index range.

`PARTIAL_X` specifies that out-of-bound reading will have values from the valid index locations of the vector, and will have 'x' value from the invalid (i.e. out of bound) locations of the vector. *This is the default*.

`ALL_X` specifies that when there is out-of-bound reading, the selected portion of the vector will be treated as all 'x' values.

`PARTIAL_0` specifies that out-of-bound reading will have values from the valid index locations of the vector, and will have '0' value from the invalid (i.e. out of bound) locations of the vector.

`PARTIAL_1` specifies that out-of-bound reading will have values from the valid index locations of the vector, and will have '1' value from the invalid (i.e. out of bound) locations of the vector.

`-VERILOG_OUTOFBOUNDWrite <Noeffect | X>`

Controls the interpretation of Verilog bit (or part)-select of vector typed variable/signal when index is out of the defined index range.

`Noeffect` specifies that out-of-bound writing will have no effect. *This is the default*.

`X` specifies that when there is out-of-bound writing, the related part of the variable/signal is assigned value 'x'.

`-VERILOG_XL_LIBORDER <OFF | ON>`

When set to ON, Conformal will follow the same search order as the Cadence NC-Verilog handling of the -y/-v command file options. Default is OFF, and this will keep the legacy behavior of Conformal Verilog parsing.

`-VHDL_OUTOFBOUNDWrite <X | Noeffect>`

Controls the interpretation of VHDL bit (or part)-select of vector typed variable/signal when index is out of the defined index range.

`X` specifies that when there is out-of-bound writing, the related part of the variable/signal is assigned value 'x'. *This is the default*.

`Noeffect` specifies that out-of-bound writing will have no effect.

`-VERILOG_TRIMINDex <OFF | ON>`

`ON` controls to trim the index to necessary bits for the Verilog files. `OFF` is the command default.

Note: This option might be used to verify implementations in which indexes are intentionally trimmed.

`-VHDL_OUTOFBOUNDRead <ALL_X | PARTIAL_X | PARTIAL_0 | PARTIAL_1>`

Controls the interpretation of VHDL bit (or part)-select of vector typed variable/signal when index is out of the defined index range.

`ALL_X` specifies that when there is out-of-bound reading, the selected portion of the vector will be treated as all 'x' values. *This is the default*.

`PARTIAL_X` specifies that out-of-bound reading will have values from the valid index locations of the vector, and will have 'x' value from the invalid (i.e. out of bound) locations of the vector.

`PARTIAL_0` specifies that out-of-bound reading will have values from the valid index locations of the vector, and will have '0' value from the invalid (i.e. out of bound locations of the vector.

`PARTIAL_1` specifies that out-of-bound reading will have values from the valid index locations of the vector, and will have '1' value from the invalid (i.e. out of bound) locations of the vector.

`-VHDL_TRIMINDex <OFF | ON>`

ON controls to trim the index to necessary bits for the VHDL files. OFF is the command default.

Note: This option might be used to verify implementations in which indexes are intentionally trimmed.

-VHDL_ELAB_UNSPECIFIED_CONFIG <ON | OFF>

By default this option is ON. If this option is OFF, Conformal will NOT elaborate unspecified vhdl configuration declaration(s).

If the user wants the vhdl configuration elaborated, the user needs to specify the configuration as root configuration or specify the configuration(s) in the design.

-WAND_GATE [FALSE | TRUE]

FALSE specifies that the direction of the WAND gate will be controlled by the READ DESIGN -continuousassignment command. (This option controls the direction of all WIRE/WOR/WAND continuous assignments. By default, WAND gates are bidirectional).

TRUE specifies that the WAND gate is always unidirectional, regardless of what is specified by READ DESIGN -continuousassignment.

Default is FALSE.

-WOR_GATE [FALSE | TRUE]

FALSE specifies that the direction of the WOR gate will be controlled by the READ DESIGN -continuousassignment command. (This option controls the direction of all WIRE/WOR/WAND continuous assignments. By default, WOR gates are bidirectional).

TRUE specifies that the WOR gate is always unidirectional, regardless of what is specified by READ DESIGN -continuousassignment.

Default is FALSE.

-ZERO_REPLICATE_AS_ZERO <ON | OFF>

When set to ON, treats zero replication as 1'b0 for Verilog designs.

When set to OFF, treats zero replication as NULL in concatenation.

OFF is the default.

Examples

The following examples use the Verilog language to show how to control index out of bound handling. You can use similar VHDL command options to control the interpretations for the VHDL language designs.

In the following RTL-1 example, the index range of variable `mem` is 0 to 2.

RTL-1

```
wire a;
reg [2:0] index;
reg [2:0] mem;
always @(*) mem[index] = a;
```

If `index` is greater than `2`, it is out of the index range. Some synthesis tools might intentionally interpret the RTL the same as with the following RTL-2 example:

RTL-2

```
wire a;
reg[2:0] index;
reg[2:0] mem;
always @(*) mem[index[1:0]] = a;
```

But with simulation, RTL-1 and RTL-2 behave differently.

With the Conformal software, when running the command:

```
set hdl options -verilog_outofboundwrite x
```

then

- `index=0` : `mem[0]` is assigned to the value of `a`
- `index=1` : `mem[1]` is assigned to the value of `a`
- `index=2` : `mem[2]` is assigned to the value of `a`
- `index=3,4,5,6,7` : `mem[0]`, `mem[1]`, and `mem[2]` are assigned to the value of `1'bx.`

This interpretation assumes that out-of-bound writing will not happen, and consequently ignores the behavior difference when `index` is greater than `2`.

When running the command:

```
set hdl options -verilog_outofboundwrite noeffect
```

then

- `index=0` : `mem[0]` is assigned to the value of `a`
- `index=1` : `mem[1]` is assigned to the value of `a`
- `index=2` : `mem[2]` is assigned to the value of `a`
- `index=3,4,5,6,7` : `mem[0]`, `mem[1]`, and `mem[2]` will not be affected with their current value.

Based on this interpretation, RTL-1 and RTL-2 are considered functional nonequivalent, and consequently the implementation from RTL-2 will be nonequivalent to RTL-1.

Using the same RTL-1 and RTL-2 examples, when running the command:

```
set hdl options -verilog_trimindex on
```

The Conformal software will interpret RTL-1 as RTL-2 by ignoring `index[2]` in the expression `mem[index]` (RTL-1). With the `-verilog_trimindex on` option, RTL-1 and RTL-2 are considered equivalent.

Port Mismatch Example

The following example illustrates when the port connection widths between the module and an instantiation do not match:

```
module top (output [3:0] o, input [2:0] in,in1);
  sub s1 (.o(o[3:0]), .in(in));
endmodule

module sub  (output [2:0] o,input [2:0] in);
  assign o = in;
endmodule
```

By default, the tool leaves `top.o[3]` unconnected. If you use the `set hdl options -portmismatch extend` command, the tool applies sign extending and connects `top.o[3]` to `1'b0`.

Unsigned Conversion Overflow Example

For example:

```
oo: out integer;
constant c1: unsigned(31 downto 0) \
  := "11000000000000000000000000000000";
oo <= conv_integer(c1);
```

When `-unsigned_conversion_overflow OFF` is specified, `oo[31]` is `1'b0`. With this option ON, `oo[31]` is `1'b1`.

Unsigned Expression Overflow Example

For example,

```
in1: in natural; oo: out integer;
oo <= in1 * 2;
```

With `-unsigned_expression_overflow OFF` is specified, `oo[31]` is `1'b0`. With this option `ON`, `oo[31]` is `in1[30]`.

Including Source Directory Example

For example, given the following directory structure:

```
common/define.vh
rtl/define.vh
rtl/test.v
```

Where `rtl/test.v` contains:

```
`include "define.vh"
```

The "`READ DESIGN rtl/test.v`" command reads in `rtl/test.v` and `rtl/define.vh` by default.

When you use the "`SET HDL OPTIONS -include_src_dir off`" command, search paths are manually specified using the `ADD SEARCH PATH` or the +incdir+ option in the Verilog command files. For example:

```
SETUP> SET HDL OPTIONS -INCLUDE_SRC_DIR OFF
SETUP> ADD SEARCH PATH common
SETUP> READ DESIGN rtl/test.v
```

The tool now reads in `rtl/test.v` and `common/define.vh`.

Remove Cell Example

```
// design.v
module design(...);
  LIB_CELL1 u0 (...);
  LIB_CELL1 u1 (...);
  LIB_CELL2 u2 (...);
  LIB_CELL3 u3 (...);
endmodule

// Conformal commands
read library -liberty LIB_CELLS.lib // read in LIB_CELL1, LIB_CELL2, LIB_CELL3
set hdl options -remove_cell lib_cell1
read design design.v
```

The command `set hdl options -remove_cell lib_cell1` will remove the cell instances `u0` and `u1` from the module `design`, and the resulting design after removing the cells u0 and u1 will be equivalent to the following:

```
module design(...);
LIB_CELL2 u2 (...);
LIB_CELL3 u3 (...);
endmodule
```

Related Commands

ELABORATE DESIGN

READ DESIGN

READ LIBRARY

WRITE BLACKBOX WRAPPER

SET HIER_COMPARE OPTION

```
SET HIer_compare Option
     [-AUTO_ANALYZE_COMPARE]
     [-GOHIER_SHARED_MEMORY]
     [-NOIGNORE_TOP_MERGE_CONSTRAINT | -IGNORE_TOP_MERGE_CONSTRAINT]
     [-TOP_USE_ALL_THREADS]
     (Setup Mode)
```

Specifies the option to control hierarchical comparison.

Tcl Command

`set_hier_compare_option`

Parameters

`-AUTO_ANALYZE_COMPARE`

Enable `analyze_compare` in `write_hier_compare_dofile`. It also turned off the `set_datapath_option -auto`.

`-GOHIER_SHARED_MEMORY`

Enables `gohier` by using shared memory. This is effective when all workers are set as localhost. This is set with `SET PARAllel Option -workers localhost localhost`.

`-NOIGNORE_TOP_MERGE_CONSTRAINT`

Do not ignore top-level merge commands in hierarchical constraint.

`-IGNORE_TOP_MERGE_CONSTRAINT`

Ignores top-level merge commands in hierarchical constraint.

`-TOP_USE_ALL_THREADS`

Allows top-module comparison to perform parallel work with the number of "worker*threads" in the command `GO HIer_compare`.

Related Commands

GO HIER COMPARE

SET PARALLEL OPTION

SET HIER_COMPARE SELECTION

```
SET HIer_compare Selection
     [-NOSMART | -SMART]
     (Setup Mode)
```

Note: This requires a Smart LEC license.

Smart LEC provides an analytic engine that can select higher quality modules for hierarchical comparisons to minimize the overhead for module comparison. Instance selection is based on the following design characteristics:

- Primitive gate counts
- RTL/netlist design
- Datapath types
- Boundary complexity
- Periphery datapaths

Tcl Command

```
set_hier_compare_selection
```

Parameters

`-NOSMART`	Do not enable smart instance selection in a hierarchical comparison. *This is default*.
`-SMART`	Enable smart instance selection in a hierarchical comparison. To enable smart instance selection in a hierarchical comparison, use the `set_hier_compare_selection -smart` command option before the `analyze_hier_compare` or `write_hier_compare_dofile` command. Note: Do not set a custom `-threshold` through `analyze_hier_compare` or `write_hier_compare_dofile` command. Smart instance selection works best with the default threshold.

Example

The following is a sample dofile for an RTL-to-synthesized gate netlist hierarchical comparison using smart instance selection.

```
// Read in the library and design files
read_design rtl.v -golden
read_design netlist.v -revised

// Specify user constraints for test/dft/etc.
// add_pin_constraint 0 scan_en -golden/revised
// add_ignored_output scan_out -golden/revised

// Enable smart instance selection before writing out
// the hierarchical compare dofile
set_hier_compare_selection -smart
write_hier_compare_dofile hier.do -balanced_extraction \
-replace -usage -constraint -noexact_pin_match -function_pin_mapping
// Run the hierarchical comparison
go_hier_compare hier.do
```

Related Commands

GO HIER COMPARE

RUN HIER_COMPARE

SET IMPLEMENTATION

```
SET IMPlementation
     <MULtiplier [-AUTO | -CSA | -WALL | -RCA | -NBW | -BKA] |
        DIVider [-RPL | -BLA | -CLA | -CLA2]
           [-OVERFLOW_TRUNCATE | -OVERFLOW_SATURATE | -OVERFLOW_DONTCARE]
           [-ALL_div | -RTL_div | -DW_div]>
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Specifies the multiplier and divider implementations in the Golden and Revised designs. Execute this command before `READ LIBRARY` and `READ DESIGN`.

The types of multipliers supported are:

- Carry Save Adder (CSA)
- Ripple Carry Adder (RCA)
- Booth Encoded-Wallace tree (WALL)
- Non-Booth Encoded-Wallace tree (NBW)
- Brent-Kung Adder (BKA)

Default Multiplier Implementation Is Automatically Determined

By default, Conformal automatically determines the multiplier implementation as follows:

If `a_width + b_width <42`, Conformal chooses NBW.

If `a_width + b_width >=42`, Conformal chooses WALL.

The types of dividers supported are:

- Ripple Borrow (RPL)
- Borrow Look-Ahead
- Carry Look-Ahead
- Carry Look-Ahead, 2-Way

Required Options

With this command, you *must* include either the `multiplier` or `divider` option; however, Conformal permits *both* options, as shown below:

```
set implementation multiplier -csa divider -bla
```

Tcl Command

```
set_implementation
```

Parameters

`MULtiplier` By default, Conformal automatically determines the multiplier implementation as follows:

If `a_width + b_width < = 52`, Conformal chooses NBW.
If `a_width + b_width > 52`, Conformal chooses WALL.

The multiplier type is one of the following:

`-AUTO`	Conformal will automatically determine the multiplier implementation. *This is the default.*
`-CSA`	Carry Save Adder
`-WALL`	Booth Encoded, Wallace tree
`-RCA`	Ripple Carry Adder
`-NBW`	Non-Booth Encoded, Wallace tree
`-BKA`	Brent-Kung Adder

`DIVider` The divider type is one of the following:

`-RPL`	Ripple Borrow, *which is the initial default*
`-BLA`	Borrow Look-Ahead
`-CLA`	Carry Look-Ahead
`-CLA2`	Carry Look-Ahead, 2-Way

The following options specify how to treat over signed division *overflow*, which is when a minimum negative value is divided by -1.

`-OVERFLOW_TRUNCATE`	Truncates results as defined by twos-complement arithmetic. *This is the default.*

`-OVERFLOW_SATURATE`	Saturates results to the largest positive value.
`-OVERFLOW_DONTCARE`	Treats overflow results as don't cares.

`DIVider` can be inferred from either `DW_div` instantation or RTL arithmetic operation (for example, '/' in Verilog). The following options specify the divider class for the following settings.

`-ALL_div`	Specifies that dividers are inferred from both RTL and DW. *This is the default.*
`-RTL_div`	Specifies that dividers are only inferred from RTL operation.
`-DW_div`	Specifies that dividers are only inferred from `DW_div` instantiation.

`-Both`	Specifies the implementation type for both the Golden and Revised designs. *This is the default.*
`-Golden`	Specifies the implementation type for the Golden design.
`-Revised`	Specifies the implementation type for the Revised design.

Related Command

READ DESIGN

SET INSTANTIATION DEPTH

```
SET INstantiation Depth
     <integer>
     (Setup Mode)
```

Use this command before the `READ DESIGN`, `READ LIBRARY`, and `ELABORATE DESIGN` commands to override the default instantiation depth limit. By default, the default instantiation depth is 100.

Tcl Command

`set_instantiation_depth`

Parameters

`<integer>`	Specifies a positive integer for the instantiation depth limit.

Examples

The following command sets the instantiation depth to 30:

```
set instantiation depth 30
```

Note: If you do not specify a positive integer, the tool issues an Error that says the specified number is too small. For example:

```
SETUP> set instantiation depth 0
// Error: The specified number 0 is too small.
```

SET LOG FILE

```
SET LOg File
     [<filename>
     [-NOBACkup]]
     [-NOPROGRESS | -PROGRESS]
     [-Replace | -Append [-BANNER | -NOBANNER]]
     (Setup / LEC Mode)
```

Writes the transcript to a specified file. The commands and any output information write to this file. As you review the file, identify commands by the keyword:

```
//Command:
```

When you want the Conformal software to stop writing to the log file, enter the command without any options.

Note: If the filename you specify already exists, you must use either the `-replace` or `-append` option. If you do not include an option, the Conformal software generates an error message that the file exists. If you receive this message, reenter the command with either a new filename or the appropriate option. If the filename is not writable, the software writes it to the `/tmp` directory.

If you are writing the transcript to a file, you might want to turn off the screen transcript display with the `SET SCREEN DISPLAY` command. (If you do not specify otherwise, the transcript prints to the screen.)

To store log files based on the software version, use the `LEC_VERSION` environment variable. For example:

```
set log file lec.$LEC_VERSION.log -replace
```

To verify the current log file setting, use the `REPORT ENVIRONMENT` command.

Tcl Command

```
set_log_file
```

Parameters

`<filename>`	Writes the transcript run to this file.

`-NOBACkup`	Do not create a backup file. **Note:** If you do not specify this option, it will create a backup file when you replace or append a file.
`-NOPROGRESS`	Do not write the percentage completion progress to the log file. *This is the default.*
`-PROGRESS`	Writes the percentage completion progress to the log file.
`-Replace`	If the specified filename already exists, overwrites the contents of that file.
`-Append`	Appends the transcript run to the end of the specified filename.
`-BANNER`	Prints banner first when appending to the logfile. *This is the default*.
`-NOBANNER`	Do not print banner when appending to the logfile. (Use only with the `-append` option.)

Related Commands

REPORT ENVIRONMENT

REPORT COMMAND PROFILE

SET COMMAND PROFILE

SET SCREEN DISPLAY

SET LOWPOWER OPTION

```
SET LOwpower Option
     (CPF Flow)
     [-GOLDEN_NETlist_style <LOGical | HYBrid | PHYsical>]
     [-REVISED_NETlist_style <LOGical | HYBrid | PHYsical>]
     [-ISOLATE_TIED_CONSTANT | -NO_ISOLATE_TIED_CONSTANT]
     [-NO_BBOX_MACRO_MODELS]
     [-REPLACE_LIB_MACRO | -NO_REPLACE_LIB_MACRO]
     (1801 Flow)
     [-NATIVE_1801]
     [-APPLY_1801_HIERARCHICAL_BBOX_SCOPE_PRIMARY_SUPPLY]
     [-AUTO_IDENTIFY_UNUSED_SUPPLIES_IN_POST_ROUTE]
     [-CONSIDER_EXTERNAL_TOP_SUPPLIES_IN_POST_ROUTE]
     [-CONSTANT_signal <NOT_LOCALIZED | LOCALIZED_Direct | LOCALIZED_Propagated> ]
     [-ENABLE_1801_HIERARCHICAL_BBOX]
     [-EXCLUDE_LOCALIZED_PATH_TYPE <NONE | TOP_PORT | MACRO_PORT |
     TOP_AND_MACRO_PORT>]
     [-GOLDEN_ANALYSIS_STYLE < PRE_SIMulation | PRE_SYNthesis
         | POST_SYNthesis | PRE_ROUTE | POST_ROUTE | POST_ROUTE_WITHOUT_SWITCH>]
     [-REVISED_ANALYSIS_STYLE    < PRE_SIMulation | PRE_SYNthesis
         | POST_SYNthesis | PRE_ROUTE | POST_ROUTE | POST_ROUTE_WITHOUT_SWITCH>]
     [-GOLDEN_DOMAIN_INTERFACE_def <1.0 | UPF | 2.0 | 1801-2009
         | 2.1 | 1801-2013>]
     [-REVISED_DOMAIN_INTERFACE_def <1.0 | UPF | 2.0 | 1801-2009
         | 2.1 | 1801-2013>]
     [-INSERT_ISO_FOR_CONSTANT_CROSSING_WITH_CONFLICTING_CLAMP |
      -NO_INSERT_ISO_FOR_CONSTANT_CROSSING_WITH_CONFLICTING_CLAMP]
     [-LSH_SAME_VOLTAGE_CROSSING <
           NEVER
         | NO_RULE_THRESHOLD
         | FORCE_SHIFT_AND_NO_RULE_THRESHOLD
         | DEFAULT_RULE_THRESHOLD
         | FORCE_SHIFT_AND_DEFAULT_RULE_THRESHOLD >]
     [-NO_ADD_VIRTUAL_SWITCH_IN_SCOPE | -ADD_VIRTUAL_SWITCH_IN_SCOPE]
     [-NO_ENABLE_TERMINAL_BOUNDARY_SUPPORT | -ENABLE_TERMINAL_BOUNDARY_SUPPORT]
     [-NO_EXCLUDE_SINGLE_PWR_AND_SINGLE_GND_AT_MISSING_CSN_MACRO |    -
     EXCLUDE_SINGLE_PWR_AND_SINGLE_GND_AT_MISSING_CSN_MACRO]
     [-NO_FILTER_FLOATING_UNDRIVEN_PORT_WITH_DIFF_SUPPLY_ONLY | -
     FILTER_FLOATING_UNDRIVEN_PORT_WITH_DIFF_SUPPLY_ONLY
     [-NO_GOLDEN_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX
         | -GOLDEN_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX]
     [-ISOLATE_ASYNC_CONTROL_NOT_IN_REQUIRED_CONDITION]
     [-NO_MATCH_STRATEGY_WITH_ONE_OF_FANOUTS_AND_NO_CROSSING_AT_OTHER_FANOUTS
         | -MATCH_STRATEGY_WITH_ONE_OF_FANOUTS_AND_NO_CROSSING_AT_OTHER_FANOUTS]
     [-NO_REVISED_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX |
           -REVISED_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX]
     [-NO_TRAVERSE_DOMAIN_BOUNDARY_FOR_ISO_ZERO_PIN_RET_MATCHING |        -
     TRAVERSE_DOMAIN_BOUNDARY_FOR_ISO_ZERO_PIN_RET_MATCHING]
     [-NO_USE_SIMSTATE_FOR_STANDBY  | -USE_SIMSTATE_FOR_STANDBY]
     [-SUPPORT_RET_STRATEGY_WITH_COMPATIBLE_CTRL_AND_RET_COND_AS_ZERO_PIN
         |-NO_SUPPORT_RET_STRATEGY_WITH_COMPATIBLE_CTRL_AND_RET_COND_AS_ZERO_PIN]
     [-INSERT_ISO_AT_FLOATING_UNDRIVEN |
        -NO_INSERT_ISO_AT_FLOATING_UNDRIVEN]
     [-TRAVERSE_AON_FOR_STRATEGY_MATCHING]
     [-TRAVERSE_BUF_AND_INV_FOR_STRATEGY_MATCHING]
     [-EXCLUDE_SIGNALS_AS_RETENTION_ELEMENTS
     [-IDENTIFY_ALIVE_POWER_DOWN_ISO_CELL <STRUCTURE | ATTRIBUTE |
     ATTRIBUTE_OR_STRUCTURE>]
     [-PST_SUPPORT_RANGE_VOLTAGE]
     [-STANDBY_DETERMINED_BY_POWER_AND_GROUND_VOLTAGE_ONLY]
     [-TRAVERSE_TERMINAL_BOUNDARY_EXCEPT_FOR_STRATEGIES |
         -NO_TRAVERSE_TERMINAL_BOUNDARY_EXCEPT_FOR_STRATEGIES]
     [-USE_ACTIVE_PHASE_FOR_ASYNC_CONTROL_CLAMP_VALUE]
     [-USE_STRATEGY_LOCATION_AS_HIERARCHY | -
     NO_USE_STRATEGY_LOCATION_AS_HIERARCHY]
     [-USE_CELL_ATTRIBUTE_FOR_ASYNC_CONTROL_CLAMP_VALUE]
     [-NOLP_CTRL_SIGNAL_COMPARED_PAIR | -LP_CTRL_SIGNAL_COMPARED_PAIR]
     [-WITNESS_LIMIT <n>]
     (Setup Mode)
```

Note: This is a Conformal Low Power command.

Enables options to execute Low Power checks. Options differ between the CPF and 1801 flows. The syntax above describes which options are available in the CPF flow and which are available with 1801.

For more information on these low power checks and cell types, see CHECK LOWPOWER CELLS.

Tcl Command

```
set_lowpower_option
```

Parameters

`-GOLDEN_NETlist_style <LOGical | HYBrid | PHYsical>`

Specifies the Golden design netlist style.

`LOGical` (*the default*) indicates that the design is a logical netlist, in that it does not have any power or ground connectivity.

`HYBrid` indicates that the design is a hybrid netlist, in that it has partial power and ground connectivity. It will be transformed to a "logical" style, where the power is defined as logical "1" and ground as logical "0", for functional comparison.

`PHYsical` indicates that the design is a physical netlist, in that it has complete power and ground connectivity. It will be transformed to a "logical" style for functional comparison.

`-REVISED_NETlist_style <PHYsical | LOGical | HYBrid>`

Specifies the Revised design netlist style.

`LOGical` (*the default*) indicates that the design is a logical netlist, in that it does not have any power or ground connectivity.

`HYBrid` indicates that the design is a hybrid netlist, in that it has partial power and ground connectivity. It will be transformed to a "logical" style, where the power is defined as logical "1" and ground as logical "0", for functional comparison.

`PHYsical` indicates that the design is a physical netlist, in that it has complete power and ground connectivity. It will be transformed to a "logical" style for functional comparison.

`-ISOLATE_TIED_CONSTANT` By default, Conformal inserts isolation on a domain crossing driven by a direct constant.

`-NO_ISOLATE_TIED_CONSTANT` Do not automatically insert isolation on domain crossings driven by a direct constant.

`-NO_BBOX_MACRO_MODELS`	By default, extracted liberty macro models and CPF macro models are blackboxed for crossing analysis by default. This option disables the auto blackboxing of macro models that are not configured through macro instance mapping.
`-REPLACE_LIB_MACRO`	When a macro model cell is read in both its Liberty through `READ LIBRARY` and its CPF model through `READ POWER INTENT`, by default, Conformal uses the CPF model and replaces Liberty for those macros.
`-NO_REPLACE_LIB_MACRO`	When a macro model cell is read in both its Liberty through `READ LIBRARY` and its CPF model through `READ POWER INTENT`, use this option to avoid the replacement of macro models extracted from Liberty.
	Without this option, Conformal will use the CPF model and replace Liberty for those macros. Thus, the CPF model written out after reading in power intent will only contain the macros used in the power intent. Users need this option when they want to generate CPF for all macro models after reading the power intent if some macros are not defined in the CPF. This helps avoid cases where Liberty gets overwritten by an empty CPF model after reading in the power intent.
(1801 Options)	**Note: These options are for the 1801 flow.**
`-NATIVE_1801`	Enables the 1801 flow.

`-APPLY_1801_HIERARCHICAL_BBOX_SCOPE_PRIMARY_SUPPLY`

Assigns the hierarchical blackbox boundary port that is missing the required port attribute to the block scope's `-include_scope` domain as its related supply.

This option is only valid when `-enable_1801_hierarchical_bbox` is set.

Warning: When `-apply_1801_hierarchical_bbox_scope_primary_supply` is set, the hierarchical boundary ports reported by the LP-Verify `1801_HIER_BBOX_ATTR_MISSING` rule as ports that are missing supply attributes will be assigned a supply equal to the primary supply of the domain of the block. This may not be consistent with the real physical implementation. It is the user's responsibility to verify the consistency with this behavior. This option is only effective when option `-enable_1801_hierarchical_bbox` is also set.

`-AUTO_IDENTIFY_UNUSED_SUPPLIES_IN_POST_ROUTE`

Setting this option allows supply nets and supply ports as well as supply nets handles that are external to the design to be specified as the driver supply of top input ports and/or receiver supply of top output ports. In addition, supplies that are internal to macro cells can be used to specify the receiver supplies of macro logic input ports and driver supply of macro output ports. These supply nets/ports/supply net handles do not have to be implemented in the design even in post-route and will not be reported as missing/not associated. This is the default.

NOTE: This option is a superset and replaces the option `CONSIDER_EXTERNAL_TOP_SUPPLIES_IN_POST_ROUTE`.

`-CONSIDER_EXTERNAL_TOP_SUPPLIES_IN_POST_ROUTE`

Setting this option allows supply nets and supply ports that are external to the design to be specified as the driver supply of top input ports and/or receiver supply of top output ports. These supply nets/ports do not have to be implemented in the design even in post-route and will not be reported as missing.

`-CONSTANT_signal`

`-CONSTANT_signal`: Specifies if a logical constant driving a domain interface port will be relocated to its receiver(s)

A constant signal driven from a supply net, tie cell output or soft/hard macro cell constant output pin or a pin with analog attribute is not a signal that could be relocated.

A terminal boundary input port is treated as a leaf port and the constant signal relocation stops at such ports.

`NOT_LOCALIZED`: The constant signal driving a domain interface port will not be relocated. This is the default for all analysis style.

`LOCALIZED_Direct`: A directly connected constant signal (1'b0/1'b1) driving a domain interface port is expected to be relocated to its receiver(s) during implementation stage(s). Such directly connected constant signal (1'b0/1'b1) will not cause crossing violations. This does not apply for post-route analysis style.

`LOCALIZED_Propagated`: A directly connected constant signal propagating through logic gates (including buffer and inverter) is expected to be relocated to its receiver(s) during implementation stage(s). Crossings related to these constant signals will not cause crossing violations. This does not apply for post-route analysis style.

`-ENABLE_1801_HIERARCHICAL_BBOX`

Enables 1801 hierarchical blackboxed block flow. This flow enables the block UPF to be applied at the bbox module. The block UPF should have well-defined attributes to perform the correct analysis at the chip-level integration. You should run CLP-Verify and clear any `1801-HIER_BBOX_*` violations before running this flow in LP-EC.

`-EXCLUDE_LOCALIZED_PATH_TYPE <NONE | TOP_PORT | MACRO_PORT | TOP_AND_MACRO_PORT>`

Specifies the path type that will be excluded from constant crossing localization. This option only has an effect if the option `-CONSTANT_signal` is set to `localized_direct` or `localized_propagated`.

`NONE`: None of the paths will be excluded from localization. *This is the default for all analysis styles*.

`TOP_PORT`: Constant crossing paths that end at a top port will not be localized.

`MACRO_PORT`: Constant crossing paths that end at a hard or soft macro input will not be localized.

`TOP_AND_MACRO_PORT`: Constant crossing paths that end at a top port, hard of soft macro input will not be localized.

`-GOLDEN_ANALYSIS_STYLE`

Specifies the analysis style of the Golden design.

Note: This option is for the 1801 flow.

`-REVISED_ANALYSIS_STYLE`

Specifies the analysis style of the Revised design.

`PRE_SIMulation`: Performs power intent syntax and quality checks with respect to the design model. This option is used to check RTL level design before synthesis. A technology library is not required if there are no library cells in the RTL, however certain checks to flag cell mapping consistency errors design power intent can only be done with technology library. *This is the default.*

`PRE_SYNthesis`: Checks the quality of the power intent specification and library `(Liberty) before gate synthesis.`

`POST_SYNthesis`: Performs power intent syntax and quality checks with respect to the design netlist and performs an analysis on the design netlist to verify that the power intent specification that applies to synthesis is implemented correctly. This option is used to check a post synthesis netlist with inserted isolation, level shifter, and state retention cells. For netlists without inserted isolation, level shifter, and state retention cells, use the `pre_synthesis` or `pre_route` analysis style.

`PRE_ROUTE`: Checks the quality of the power intent specification and library (Liberty) before place and route. PG connection and power switches related checks are performed at this stage.

`POST_ROUTE`: Performs power intent syntax and quality checks with respect to the design netlist, performs an analysis on the design netlist to verify the power intent specification that applies to physical place and route is implemented correctly, and performs an analysis of the power and ground connectivity for domain analysis. This option is used to check a post place and route netlist with inserted isolation cells, level shifter cells, state retention cells power switch cells, and power and ground connectivity.

`POST_ROUTE_WITHOUT_SWITCH`: Performs power intent syntax and quality checks with respect to the design netlist, performs an analysis on the design netlist to verify the power intent specification is implemented correctly, and performs an analysis of the power and ground connectivity for domain analysis. This option is used to check a netlist with inserted low power cells and power and ground connectivity. In the netlist the power switch cells may not have been implemented.

`-GOLDEN_DOMAIN_INTERFACE_def`

Specifies the way a power domain's interface is defined for the golden design, which is not the same in different versions of UPF/IEEE-1801.

If this option is not specified, the definition is taken from the design attribute `domain_interface_def` as set in the UPF file; if that attribute is not set, the definition depends on the lowest `upf_version` in effect during the execution of power intent commands.

In UPF (or 1.0), the interface of a power domain contains only the (low-conn) upper boundary ports of "interface instances" in the power domain.

In 1801-2009 (or 2.0), the lower boundary ports (high-conn of sub-instances that belong to a different domain) are also part of the domain interface. Since 1801-2013, the domain interface also includes macro ports whose driver/receiver supply differs from the primary supply of the domain to which the macro instance belongs.

If the power intent was written for a given version, executing it with the incorrect semantics can lead to double insertion of low power cells and other problems. For this reason, Cadence tools follow the domain interface definition most relevant to the language version of the power intent being executed. On the other hand, some third party implementation tools might not support all three semantics correctly. For interoperability reasons, it could be necessary to specify a given semantic version explicitly using the design attribute or this low power option.

`-REVISED_DOMAIN_INTERFACE_def`

Specifies the way a power domain's interface is defined for the revised design, which is not the same in different versions of UPF/IEEE-1801.

(See `-GOLDEN_DOMAIN_INTERFACE_def` for more information.)

`-INSERT_ISO_FOR_CONSTANT_CROSSING_WITH_CONFLICTING_CLAMP`

Setting this option specifies that constant crossings for which the isolation strategy's clamp value conflicts with the constant value will not be optimized even if the option `set_lowpower_option -constant_signal localized_direct` or `set_lowpower_option -constant_signal localized_propagated` is set. Hence, isolation cell will be inserted/expected on such crossings. *This is the default.*

`-NO_INSERT_ISO_FOR_CONSTANT_CROSSING_WITH_CONFLICTING_CLAMP`

Disables `-INSERT_ISO_FOR_CONSTANT_CROSSING_WITH_CONFLICTING_CLAMP` for the design.

`-LSH_SAME_VOLTAGE_CROSSING`

Specifies when a level shifter strategy can be applied to a *same voltage crossing port* (a port that has a driver and a load supplied by independent root supply drivers, but is not in any crossing between a driver and a load that can be at different voltages in some power state).

`NEVER`: Level shifter strategy does not apply to same voltage crossing ports. *This is the default*.

`NO_RULE_THRESHOLD`: Level shifter strategy can apply to same voltage crossing ports when neither `-rule` nor `-threshold` is specified.

`FORCE_SHIFT_AND_NO_RULE_THRESHOLD`: Level shifter strategy can apply to same voltage crossing ports only when it is specified with `-force_shift` and neither `-rule` nor `-threshold` is specified.

`DEFAULT_RULE_THRESHOLD`: Level shifter strategy can apply to same voltage crossing ports when neither `-rule` nor `-threshold` is specified or the values specified are equal to the default values of these options.

`FORCE_SHIFT_AND_DEFAULT_RULE_THRESHOLD`: Level shifter strategy can apply to same voltage crossing ports only when it is specified with `-force_shift` and neither `-rule` nor `-threshold` is specified or the values specified are equal to the default values of these options.

`-NO_ADD_VIRTUAL_SWITCH_IN_SCOPE`

With this option, the virtual power switch is inserted at the scope where it was defined. *This is the default*.

`-ADD_VIRTUAL_SWITCH_IN_SCOPE`

If `create_power_switch` does not specify `-domain`, this will insert the virtual power switch at the scope where the switch was defined.

If `create_power_switch` specifies `-domain`, the location and number of inserted virtual power switches will depend on the elements of the domain and the `create_power_switch` command specification:

- If there is only one element in the domain, the virtual switch will be inserted in that element
- If there are multiple elements in the domain and if the `create_power_switch` has no `-ack_port` option, there will be one virtual switch inserted in each element of the domain
- If the `create_power_switch` has the `-ack_port` option, there will be a single virtual switch inserted at the scope where the command was defined

`-NO_ENABLE_TERMINAL_BOUNDARY_SUPPORT`

Disables the terminal boundary support. *This is the default*.

`-ENABLE_TERMINAL_BOUNDARY_SUPPORT`

Enables the terminal boundary support.

`-NO_EXCLUDE_SINGLE_PWR_AND_SINGLE_GND_AT_MISSING_CSN_MACRO`

Do not exclude the macro cell with a single power and a single ground pin at 1801_SUPPLY_CSN_MISSING_FOR_MACRO check.

This is the default.

`-EXCLUDE_SINGLE_PWR_AND_SINGLE_GND_AT_MISSING_CSN_MACRO`

Exclude the macro cell with a single power and a single ground pin reported at 1801_SUPPLY_CSN_MISSING_FOR_MACRO checker. A macro cell which has a single power pin and a single ground pin without connect_supply_net setting is expected to be connected to the primary supply set of its placed domain.

SUPPLY_SET_DOMAIN_PRIMARY_CONFLICT checker excludes a macro cell with a single power pin and a single ground pin without this option since users expect connect_supply_net applied at all the macro power pins and ground pins and the specified connections may not be consistent with the primary supply set of its placed domain. With this option, SUPPLY_SET_DOMAIN_PRIMARY_CONFLICT includes the macro cell with single power pin and ground pin since such macro cell may not have connect_supply_net applied.

`-NO_FILTER_FLOATING_UNDRIVEN_PORT_WITH_DIFF_SUPPLY_ONLY`

Setting this option will not apply '`-diff_supply_only TRUE`' filter to strategy elements for which the logic driver or receiver supply set cannot be determined because the domain boundary port is undriven or floating. *This is the default*.

`-FILTER_FLOATING_UNDRIVEN_PORT_WITH_DIFF_SUPPLY_ONLY`

Setting this option will apply '`-diff_supply_only TRUE`' filter to strategy elements for which the logic driver or receiver supply set cannot be determined because the domain boundary port is undriven or floating.

`-NO_GOLDEN_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX`

Disables -`GOLDEN_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX` for the Golden design. *This is the default*.

`-GOLDEN_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX`

In the Golden design, the physical supply set assignment for blackboxed instance ports will be overwritten by the supply set specified in the power intent using '`set_port_attribute`' or '`set_related_supply_set`' commands. For blackbox inputs/inouts, the physical receiver supply will be overwritten by the '`set_port_attribute -receiver_supply`' or '`set_related_supply_net`' setting. For blackbox outputs/inouts, the physical driver supply will be overwritten by '`set_port_attribute -driver_supply`' or '`set_related_supply_net`' setting.

`-ISOLATE_ASYNC_CONTROL_NOT_IN_REQUIRED_CONDITION`

Specifies that if the asynchronous set and reset pins of zero-pin retention cells are not part of the expression in the required_condition of the library cell's retention_condition group, they will be isolated to their inactive values. This option only takes effect when `-USE_CELL_ATTRIBUTE_FOR_ASYNC_CONTROL_CLAMP_VALUE` is also set.

`-NO_MATCH_STRATEGY_WITH_ONE_OF_FANOUTS_AND_NO_CROSSING_AT_OTHER_FANOUTS`

Setting this option means that an inserted low power cell with multiple fanouts will be matched to a strategy only if every fanout is targeted by the strategy. This is default.

`-MATCH_STRATEGY_WITH_ONE_OF_FANOUTS_AND_NO_CROSSING_AT_OTHER_FANOUTS`

Setting this option allows strategy matching for low power cells that have multiple fanouts such that fanouts crossing the domain boundary must have a strategy applied to them while the fanouts that are not boundary ports must have logic receiver supply equivalent to the supply of the logic driver of the low power cell's data pin.

`-NO_REVISED_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX`

Disables - `REVISED_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX` for the Revised design. *This is the default*.

`-REVISED_OVERWRITE_PHYSICAL_SUPPLY_SET_BY_SPA_FOR_BBOX`

In the Revised design, the physical supply set assignment for blackboxed instance ports will be overwritten by the supply set specified in the power intent using '`set_port_attribute`' or '`set_related_supply_set`' commands. For blackbox inputs/inouts, the physical receiver supply will be overwritten by the '`set_port_attribute -receiver_supply`' or '`set_related_supply_net`' setting. For blackbox outputs/inouts, the physical driver supply will be overwritten by '`set_port_attribute -driver_supply`' or '`set_related_supply_net`' setting.

`-NO_TRAVERSE_DOMAIN_BOUNDARY_FOR_ISO_ZERO_PIN_RET_MATCHING`

To match the isolation cell with a zero-pin retention strategy, the isolation cell must be placed in the same domain as the zero-pin retention cell. *This is the default*.

`-TRAVERSE_DOMAIN_BOUNDARY_FOR_ISO_ZERO_PIN_RET_MATCHING`

Allow crossing domain boundaries to match isolation cell inserted for zero pin retention strategy. This option helps to match isolation cells that are inserted in a domain other than the domain of the zero-pin retention cell or to match isolation cells when there are buffers placed between isolation cell and the zero-pin retention cell and the buffers are in a domain different than the isolation/retention cell.

`-NO_USE_SIMSTATE_FOR_STANDBY`

In 1801, standby mode is supported through the simstate argument of the `add_power_state` command. With this option, the standby state will be treated as normal ON state and the standby state crossing checks will not be turned on. *This is the default*.

`-USE_SIMSTATE_FOR_STANDBY`

In 1801, standby mode is supported through the simstate argument of the `add_power_state` command. With this option, the supply set is considered to be in standby mode in a power state if all of its supply functions are FULL_ON and the simstate is one of following, CORRUPT_ON_ACTIVITY, CORRUPT_ON_CHANGE, CORRUPT_STATE_ON_ACTIVITY, CORRUPT_STATE_ON_CHANGE. The standby state crossing checks will also be turned on.

`-EXCLUDE_SIGNALS_AS_RETENTION_ELEMENTS`

1801 retention strategy applied objects include the signal names by default based on 1801 LRM usage. Use this option to exclude signals at the applied elements of a retention strategy.

`-INSERT_ISO_AT_FLOATING_UNDRIVEN`

Isolation cells inferred by elements of `set_isolation` strategy that have no logic drivers or logic receivers are inserted. *This is the default*.

`-TRAVERSE_AON_FOR_STRATEGY_MATCHING`

Setting this option allows strategy filtering and matching to traverse buffers and inverters that are implemented using always-on cells, but are inserted between the isolation or level-shifter instances and domain boundary (traverses through the buffers and inverters as long as no domain boundary pin is crossed).

`-TRAVERSE_BUF_AND_INV_FOR_STRATEGY_MATCHING`

Setting this option allows strategy filtering and matching to traverse buffers and inverters that are not implemented using always-on cells but are inserted between the isolation or level shifter instances and domain boundary (traverses through the buffers and inverters as long as no domain boundary pin is crossed).

`-NO_INSERT_ISO_AT_FLOATING_UNDRIVEN`

With this option, isolation cells inferred by elements of `set_isolation` strategy that have no logic drivers or logic receivers will not be inserted.

`-IDENTIFY_ALIVE_POWER_DOWN_ISO_CELL <STRUCTURE | ATTRIBUTE | ATTRIBUTE_OR_STRUCTURE>`

`STRUCTURE`, extracts the isolation cell or shifter cell with enable pin which isolation function is still alive while its isolation supply is off based on cell structure (output is `NOR` and `NAND` function with single primary power pin and single ground pin). *This is the default*.

`ATTRIBUTE`, extracts the isolation cell or level shifter cell with enable pin which isolation function is still alive based on Liberty or UPF attribute. When the output data pin of an isolation cell or level shifter cell with enable pin has `alive_during_partial_power_down:true` attribute, this cell is taken as an isolation cell which isolation function is still active while its related supply is off.

`ATTRIBUTE_OR_STRUCTURE`, extracts the isolation cell or level shifter cell with enable pin which isolation function is still alive when either its cell structure meets the tool identify mechanism or it has the required Liberty or UPF attribute.

`-PST_SUPPORT_RANGE_VOLTAGE`

Treat the 2-valued and 3-valued voltages specified in UPF command `add_port_state` and `add_power_state` as voltage range instead of its nominal value.

`-STANDBY_DETERMINED_BY_POWER_AND_GROUND_VOLTAGE_ONLY`

With this option, if a supply set is specified as standby in a power state with its power and ground voltages are {Vp, Vg}. Then all supply sets with equivalent power supply and ground supply will also be considered in standby mode in a power state where its power voltage and ground voltage are consistent to {Vp, Vg}.

This option has no effect if lowpower option `-USE_SIMSTATE_FOR_STANDBY` is not specified.

`-TRAVERSE_TERMINAL_BOUNDARY_EXCEPT_FOR_STRATEGIES`

Considers the terminal boundaries as transparent and all supply and power state analysis are done based on the full design. *This is the default*.

`-NO_TRAVERSE_TERMINAL_BOUNDARY_EXCEPT_FOR_STRATEGIES`

Honors the terminal boundaries and check each soft macro scope and top scope independently.

`-SUPPORT_RET_STRATEGY_WITH_COMPATIBLE_CTRL_AND_RET_COND_AS_ZERO_PIN`

When the retention strategy is specified using options `-save_signal`, `-restore_signal`, and `-retention_condition`, consider the strategy a zero-pin retention strategy for implementation when all options specify the same signal, and `-restore_signal` and `-save_signal` have opposite phases, and `-restore_signal` and `-retention_condition` have matching phases.

For example, strategy RET1 would be considered as zero-pin retention strategy:

```
set_retention RET1 -domain PD -
save_signal {ret low} -restore_signal
{ret high} -retention_condition {ret}
```

`-NO_SUPPORT_RET_STRATEGY_WITH_COMPATIBLE_CTRL_AND_RET_COND_AS_ZERO_PIN`

When the retention strategy is specified using options `-save_signal`, `-restore_signal`, and `-retention_condition`, consider the strategy a single-pin retention strategy for implementation when all options specify the same signal, and `-restore_signal` and `-save_signal` have opposite phases, and `-restore_signal` and `-retention_condition` have matching phases. *This is the default*.

For example, strategy RET1 would be considered as single-pin retention strategy, while strategy RET2 would be considered a zero-pin retention strategy:

```
set_retention RET1 -domain PD -
save_signal {ret low} -restore_signal
{ret high} -retention_condition {ret}
```

```
set_retention RET2 -domain PD -
retention_condition {ret}
```

`-USE_ACTIVE_PHASE_FOR_ASYNC_CONTROL_CLAMP_VALUE`

When selecting a map cell for a given element of the retention strategy, the active phase of asynchronous control signals in the selected map cell and the element in the design do not have to match. *This is the default*.

`-USE_CELL_ATTRIBUTE_FOR_ASYNC_CONTROL_CLAMP_VALUE`

For zero-pin retention cell insertion, determining the required clamp value for the isolation cells driving the clock and asynchronous control pins requires selecting a cell from the list of mapped cells. With this option, when selecting a map cell for a given element of the retention strategy, the active phase of asynchronous control signals in the selected map cell and the element in the design must match. This option takes effect when `-USE_ACTIVE_PHASE_FOR_ASYNC_CONTROL_CLAMP_VALUE` option is not set.

`-USE_STRATEGY_LOCATION_AS_HIERARCHY`

With this option, the `-location` in `set_isolation` or `set_level_shifter` strategy is not assumed to denote the domain.

`-NO_USE_STRATEGY_LOCATION_AS_HIERARCHY`

With this option the `-location` in `set_isolation` or `set_level_shifter` strategy is assumed to denote the domain rather than hierarchy. With location denoting the domain, the insertion of lowpower cells can be shifted from the location specified by the strategy to any connected net, as long as no domain boundary pin is crossed. *This is the default*.

`-NOLP_CTRL_SIGNAL_COMPARED_PAIR`

Does not enable the feature of using probe point compared pairs for comparing control signals of low-power strategies. *This is the default*.

`-LP_CTRL_SIGNAL_COMPARED_PAIR`

Enables the feature of using probe point compared pairs for comparing control signals of low-power strategies.

Users must use commands ANalyze LP_control Pair or ADD LP_control Pair to add compared pairs of low-power control signals.

`-WITNESS_LIMIT <N>` For certain rule checks, witnesses can have shared characteristics. The option `-witness_limit` specifies the maximum number of occurrences that can be created out of a single group of similar witnesses. The exact witness grouping criteria are rule-dependent. By default, the witness limit is 1. If a witness limit of 0 is specified, there is no limit: an occurrence is created for each witness. This option takes effect during the execution of the command that performs the rule checks. If the witness limit is changed after the checks are done, it's necessary to run the checks again.

Examples

Related Commands

ADD LOWPOWER CELLS

SET MAPPING METHOD

```
SET MApping Method
     < [-NAme    <First | Guide | Only> |-NOName]
       [-NOPhase | -Phase]
       [-NOSensitive | -Sensitive]>
     [-BBOX_NAme_match | -NOBBOX_NAme_match]
     [-DONOTMAP_pattern]
     [-IGNORE_BBOX_UNMAPPED_FEEDTHROUGH_INPUT]
     [-MAPPING_FILE | -NOMAPPING_FILE]
     [-MAP_UNREACH_IN_MAPPING_FILE]
     [-MEM]
     [-NAME_EFFORT <HIgh | MEdium | LOw>]
     [-NOALIAS | -ALIAS]
     [-NONETS | -NETS]
     [-NOPHASEMAPMODEL | -PHASEMAPMODEL]
     [-NOUNREACH | -UNREACH]
     [-REPORT_SUMMARY_SHOW_ZERO_COUNT]
     [-REPORT_UNREACH | -NOREPORT_UNREACH]
     [-SEARCH_IN_MAPPING_FILE]
     [-SKIP_GATED_CLOCK_DLAT]
     [-SKIP_SEQ_CONSTANT]
     [-TIMEOUT_minutes <number>]
     [-Z_NAMEMAP_EXCLUDE <string...>]
     (Setup / LEC Mode)
```

Specifies the mapping method, phase, case sensitivity, and handling for unreachable points and blackboxes when Conformal maps the key points. With the `-name` option, paths of the gates indicate some type of starting point to map key points. *The system default is name first.* This default lets Conformal first map key points with the same paths, then map the remaining unresolved key points with a mapping algorithm. All remaining unresolved key points become unmapped points.

Use the `REPORT ENVIRONMENT` command to display the setting of the mapping method and phase.

Tcl Command

```
set_mapping_method
```

Parameters

`-NAme`	The mapping method operates under the modes described as follows:	
	`First`	Conformal maps the key points with the paths of the gates first. Then, Conformal uses the mapping algorithm to map the rest of the key points. *This option is the system default.*
	`Guide`	Conformal maps key points with a mapping algorithm first.
	`Only`	Conformal only maps the key points based on the paths of the gates
`-NOName`	Do not map key points based on the paths of the gates. If the mapping algorithm cannot map a key point, it remains unmapped.	
`-NOPhase`	Do not map key points with an inverted phase. *This is the default.* These key points are represented with the symbol "+". Comparison results are either equivalent or nonequivalent.	
`-Phase`	Maps key points with an inverted phase. These key points are represented with the symbol "-". Comparison results are either inverted-equivalent or nonequivalent.	
`-NOSensitive`	Specifies that key point names are not case sensitive. *This is the default.* Notice that during mapping, when PIs/POs have the same names under nosensitive mapping, it will automatically turn into "`-sensitive`" if sensitive mapping can avoid the same names.	
`-Sensitive`	Specifies that key point names are case sensitive.	
`-BBOX_NAme_match`	Maps blackboxes only if *both* the module names and instance names match. *This is the default.*	
`-NOBBOX_NAme_match`	Maps blackboxes if instance names match.	

`-DONOTMAP_pattern`	Do not map key points whose names match the specified pattern (pattern must be provided as a regular Perl expression). Default value is an empty string.v
`-IGNORE_BBOX_UNMAPPED_FEEDTHROUGH_INPUT`	Ignores the unmapped feedthrough inputs of blackboxes such that unmapped situations for these inputs do not cause BBOX NEQs.
`-MAPPING_FILE`	Use the mapping file specified by `SET ANALYZE OPTION -MAPPING_FILE` during mapping. *This is the default.*
`-NOMAPPING_FILE`	Do not use the mapping file specified by `SET ANALYZE OPTION -MAPPING_FILE` during mapping.
`-MAP_UNREACH_IN_MAPPING_FILE`	Maps unreachable key points specified in the mapping file.
`-MEM`	Uses a mapping method that is compatible with memory array structures and results in faster and more accurate memory array element mapping and consequently more accurate comparison, reducing the number of false errors due to incomplete or incorrect mapping and reducing the need for renaming rules. This option should be set when the design to be verified is a memory circuit.

`-NAME_EFFORT`	Uses the specified amount of effort for key point mapping. This option eliminates the need for simple renaming rules such as:
	
	`add renaming rule R1 "reg\[%d\]" "reg(@1)" -golden`, which maps the following Golden and Revised design DFFs:
	
	`Golden: DFF A/B/C_reg[5]`
	
	`Revised: DFF A/B/C_reg(5)`
	
	This option applies to only DFFs and DLATs.

`HIgh`	Specifies a high effort for key point mapping. This option is the system default.
`MEdium`	Specifies a medium effort for key point mapping.
`LOw`	Specifies a low effort for key point mapping.

`-NOALIAS`	Do not change keypoint names into name aliases during mapping. *This is the default*.
`-ALIAS`	Change keypoint names into name aliases during mapping. Name aliases are specified using the `ADD NAME ALIAS` command.
`-NONETS`	Do not map according to net names. *This is the default.*
`-NETS`	Maps key points according to net names.
`-NOPHASEMAPMODEL`	Disables `-PHASEMAPMODEL`. *This is the default*.
`-PHASEMAPMODEL`	Uses the phase information provided by the `ADD MAPPING MODEL` command to determine the mapping phase. Use this option when there is a phase mismatch between the simulation and synthesis libraries for one or more sequential cells. If `-phase` is also enabled, this option overrides the `-phase` setting. If there is no `ADD MAPPING MODEL` information entered, then this option has no effect.
`-NOUNREACH`	Do not map unreachable key points. Unreachable key points are those that do not eventually affect the PO of the design. *This is the default.*
`-UNREACH`	Maps unreachable key points. Unreachable key points are those that do not eventually affect the PO of the design.
`-REPORT_SUMMARY_SHOW_ZERO_COUNT`	Specifies that the summary will be reported with ZERO count if there is an unmapped point of the same type either in Golden or Revised design.
`-REPORT_UNREACH`	Reports unreachable key points. *This is the default.*
`-NOREPORT_UNREACH`	Do not report unreachable key points.
`-SEARCH_IN_MAPPING_FILE`	Searches the specified gates' drivers for corresponding keypoints for mapping when reading in the mapping file.
`-SKIP_GATED_CLOCK_DLAT`	When mapping unreachable points (through `-UNREACH`), this option skips deglitching clock-gating DLATs.

`-SKIP_SEQ_CONSTANT`	When mapping unreachable points (through `-UNREACH`), this option skips flip-flops that have been remodeled as sequential constants.
`-TIMEOUT_minutes <number>`	Specifies the number of minutes for the mapping process to continue before it is interrupted. The default value is zero (0), which disables this check.
`-Z_NAMEMAP_EXCLUDE <string...>`	Specifies that Zs with name matching patterns are not mapped by name. The pattern strings can be multiple, and their style is the same as the style of renaming rule. For example, `-Z_NAMEMAP_EXCLUDE UNCONNECTED%d$ UNCONNECTED_HIER_Z%d$`.

Related Commands

ADD NAME ALIAS

ADD MAPPED POINTS

ADD RENAMING RULE

DELETE MAPPED POINTS

DELETE RENAMING RULE

MAP KEY POINTS

READ MEMORY PRIMITIVE

REPORT ENVIRONMENT

REPORT MAPPED POINTS

REPORT RENAMING RULE

REPORT UNMAPPED POINTS

REPORT UNMAPPED POINTS

SET ANALYZE OPTION

<u>SET NAMING RULE</u>

SET MOS MODEL

```
SET MOs Model
     <NMOS | PMOS>
     [<Spice_model_name>]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Specifies the MOS model names used in SPICE. You can then re-read the SPICE netlist.

When reading in SPICE netlists, the parser automatically identifies transistor model names as PMOS and NMOS types. However, if you have models that were not defined using `.MODEL` statements, the parser identifies them as `ERROR`. Instead of altering your SPICE file, you can use this command.

Note: You must run this command before reading in the SPICE netlist.

Tcl Command

```
set_mos_model
```

Parameters

`NMOS`	Defines the model name as an N-Channel device.
`PMOS`	Defines the model name as a P-Channel device.
`<Spice_model_name>`	Specifies a single or list of names for which to define models.

SET MULTIBIT OPTION

```
SET MULTIBIT Option
     [-Delimiter <string>]
     [-GROUP <string>]
     [-PREFIX <string>]
     [-PREPEND <string>]
     [-RANGE_PATTERN]
     [-RANGE_TOKEN <string>]
     [-SEPARATOR_TOKEN <string>]
     [-TEST <string>]
     [-GOLden | -REVised]
     (All Modes)
```

Splits a multibit instance name into individual instance names using a name delimiter, a group pattern, and/or a prefix. This can facilitate the successful mapping of the individual bits for the multibit cell.

Tcl Command

```
set_multibit_option
```

Parameters

`-Delimiter <string>`	Specifies the name delimiter as a Perl regular expression. The default value is "`_MB_`".
`-GROUP <string>`	Specifies the name group as a Perl regular expression. It must be in the format of "`(<`*`pattern`*`>)`". The default value is `(.*?)`.
`-PREFIX <string>`	Specifies the prefix (from synthesis). The default value is "`CDN_MBIT_`".
`-PREPEND <string>`	Specifies the prepend string before the prefix as a Perl regular expression. It must be in the format of "`^(<`*`pattern`*`>)`". The default value is "`^(.*?)`".

`-RANGE_PATTERN`	Specifies a pattern (as a regular Perl expression) to match the range portion in the multibit instance name. Pattern must be specified in the following format:
	`(<pattern>)(\d+)<string>(\d+)(<pattern>)`
	For example, if the range portion in the multibit instance is '3_0', the pattern will be specified as:
	`'(.*)(\d+)_(\d+)(.*?)'`
	The default value is an empty string.

`-RANGE_TOKEN`	Specifies the token to match the multibit naming pattern. It must be used with -SEPARATOR_TOKEN. For example, if RANGE_TOKEN is _TO_ and SEPARATOR_TOKEN is _MB_ It would match: reg_10_TO_8_MB_6_ to reg_10_,reg_9_,reg_8_, and reg_6_.
`-SEPARATOR_TOKEN`	Specifies the token to match the multibit naming pattern. It must be used with -RANGE_TOKEN.
`-TEST <string>`	Tests how the specified multibit instance name will be split into individual bit names.
`-GOLden`	Applies the settings to the Golden design. By default settings are applied to both Golden and Revised.
`-REVised`	Applies the settings to the Revised design. By default settings are applied to both Golden and Revised.

Example

For example:

- The `SET MULTIBIT OPTION` command (without any options), splits the multibit instance name `CDN_MBIT_a_reg[0]_MB_a_reg[1]` into `a_reg[0]` and `a_reg[1]`.

  While, the multibit instance name `resolved_hier_CDN_MBIT_a_reg[0]_MB_a_reg[1]` is split into `resolved_hier_a_reg[0]` and `resolved_hier_a_reg[1]` by default.

- The following tests how a pattern would be split:

  For an original name of `a_reg_MG_b_reg`

```
SETUP> set multibit option -test a_reg_MB_b_reg
```

Tool returns:

```
a_reg

b_reg
```

- The following command splits the multibit instance name `auto_vector_a_reg[0]_2ndsig_a_reg[1]`, into `a_reg[0]` and `a_reg[1]`.

```
MODE> set multibit option -prefix "auto_vector_" -delimiter "_2ndsig_"
```

- The following command splits instance name `a_reg[0][1]_a_reg[0][2]` into `a_reg[0][1]` and `a_reg[0][2]`:

```
SETUP> set multibit option -group "(.*?(_reg)+(\[\d+\])+)" -delimiter "_"
```

- The following command splits instance name `MBIT_a_reg[0]MBIT_a_reg[1]_MB_a_reg[2]_MB_a_reg[3]` into `a_reg[0]`, `a_reg[1]`, `a_reg[2]` and `a_reg[3]`:

```
SETUP> set multibit option -prefix "MBIT_" -delimiter "(?:(?:_MB_)|(?:MBIT_))"
```

Related Commands

SET MAPPING METHOD

SET MULTIPLIER IMPLEMENTATION

```
SET MUltiplier Implementation
     <AUTO | CSA | RCA | WALL | NBW | BKA | BOOTHRCA | NONBOOTHRCA>
     [-Both | -Golden | -Revised]
     [-Verbose]
     (Setup Mode)
```

Specifies the multiplier implementation in the Golden and Revised designs. Execute this command before `READ LIBRARY` and `READ DESIGN`.

The types of multipliers you can specify are:

- Carry Save Adder (CSA)
- Ripple Carry Adder (RCA)
- Booth Encoded-Wallace tree (WALL)
- Non-Booth Encoded-Wallace tree (NBW) multipliers
- Brent-Kung Adder (BKA)

The CSA multiplier implementation is the default.

Tcl Command

```
set_multiplier_implementation
```

Parameters

`AUTO`	The system default for Conformal is that it automatically determines the multiplier implementation as follows: If a_width + b_width < = 52, Conformal chooses NBW. If a_width + b_width > 52, Conformal chooses WALL.
`CSA`	Specifies that the multiplier type is a Carry Save Adder (CSA) multiplier.
`RCA`	Specifies that the multiplier type is a Ripple Carry Adder (RCA) multiplier.

`WALL`	Specifies that the multiplier type is a Booth Encoded, Wallace tree multiplier.
`NBW`	Specifies that the multiplier type is a non-Booth Encoded, Wallace tree multiplier.
`BKA`	Specifies that the multiplier type is a Brent-Kung Adder (BKA).
`BOOTHRCA`	Specifies that the multiplier type is Booth Encoded Rippler Carry Adder (BOOTHRCA).
`NONBOOTHRCA`	Specifies that the multiplier type is Non-Booth Encoded Rippler Carry Adder (NONBOOTHRCA)
`-Both`	Specifies the multiplier implementation type for both the Golden and Revised designs. *This is the default.*
`-Golden`	Specifies the multiplier implementation type for the Golden design.
`-Revised`	Specifies the multiplier implementation type for the Revised design.
`-Verbose`	Provides additional information.

Related Command

READ DESIGN

SET MULTIPLIER OPTION

```
SET MUltiplier Option
     [-NOAUTO | -AUTO]
     [-NOCDP_INFO | -CDP_INFO]
     [-Verbose]
     (Setup Mode)
```

Specifies whether Conformal will automatically analyze multipliers when switching from Setup to LEC mode. Additionally, use the `-cdp_info` option if you want Conformal to let you know when Conformal XL will be helpful.

Tcl Command

`set_multiplier_option`

Parameters

`-NOAUTO`	Do not automatically analyze multipliers when switching from Setup to LEC mode. *This is the default.*
`-AUTO`	Automatically analyzes multipliers when switching from Setup to LEC mode.
`-NOCDP_INFO`	Do not display a message when Conformal XL can enhance multiplier analysis. *This is the default.*
`-CDP_INFO`	Displays a message when Conformal XL can enhance multiplier analysis.
`-Verbose`	Provides additional information.

Related Commands

ANALYZE DATAPATH

ANALYZE MODULE

REPORT DATAPATH OPTION

REPORT MULTIPLIER OPTION

SET DATAPATH OPTION

SET FLATTEN MODEL

SET NAMING RULE

```
SET NAming Rule
     << -Hierarchical_separator <string>> |
     < -Tristate <string>> |
     < -REGister <string>> |
     < -Inverted_pin_extension <string>> |
     < -Parameter <string> [-VALUE_format <LEC | RC | GENUS | DC>]> |
     < -INSTANCE_Array <string>> |
     < -Array_delimiter <left_string> <right_string>> |
     < -Field_delimiter <left_string> <right_string>> |
     < -INTerface_delimiter <left_string> <right_string>> |
     < -INStance   [-ALL | -VLG | -VHDL] <condgen_string> <forgen_string>
     <instance_string>> |
     < -VARiable   [-ALL | -VLG | -VHDL] <condgen_string> <forgen_string>
     <variable_string>> |
     < < -NOMDPORTflatten | < -MDPORTflatten
       [ALL | LOGIC_only | RECORD_only | UNION_only] > >
       [ -NOMDPORT_BITblast | -MDPORT_BITblast] > |
     <-ENABLE_UNNAMED_blk_naming | -NOENABLE_UNNAMED_blk_naming> |
     <-NOENABLE_PROC_name | -ENABLE_PROC_name> |
     <-MAX_Para_mod_len <length_size>> |
     <-STRIngformat <DEFAULT | HEX | STR>>
     <-DATAPATH_INSTANCE <pattern>> |
     <-MOD_PARAM [USE_ANYWAY | USE_WHEN_DIFF | USE_ALL]>>
     <-INSTARRAY_range_reset | - INSTARRAY_range_reset <ASCEND | DESCEND>>
     [-Both | -Golden | -REvised]
     (Setup Mode)
```

Specifies the naming rules for an RTL or hierarchical design. Conformal uses the renaming rule to construct the name for the design module, instance, variable, or port. Execute this command before `READ LIBRARY` and `READ DESIGN` (note that `SET NAMING RULE -inverted_pin_extension` does not have this requirement).

Each string for all of the settings must be enclosed in double quotes (" "). These double quotes can be empty.

Use the `REPORT ENVIRONMENT` command to display the settings for the naming rules for the Golden and Revised designs.

Tcl Command

```
set_naming_rule
```

Parameters

`-Hierarchical_separator <string>`

A character or string that specifies the hierarchical separator. *The default is* "`/`".

Use the hierarchical separator string when matching key points between the Golden and Revised designs. The hierarchical separator setting has no effect on the way key points are reported (for example, when you use the `REPORT GATE` command).

`-Tristate <string>`	A string specifies how tri-state names should be constructed. *The default is* "`%s_tri`". Where `%s` denotes the tristate variable name. The string must contain exactly one "`%s`".
`-REGister <string>`	A string that specifies how register names should be constructed. *The default is* "`%s_reg`". Where `%s` denotes the register name. The string must contain exactly one "`%s`".

`-Inverted_pin_extension <string>`

A string that specifies the inverted pin extension. This option appends the string to the Golden or Revised pin name.

For example, the Golden design has a pin named "`a`". In the Revised design, it is named "`a_BAR`". The following command specifies the correct name mapping:

```
set naming rule -inverted_pin_extension "_BAR" -golden
```

`-Parameter <string> [-VALUE_format <LEC | RC | GENUS | DC>]`

String to use in parameterized module names. When you instantiate an existing module with new parameter values, Conformal creates a new module and names it using the following renaming rule:

`<original_module_name><<parameter_string>`

`<parameter value> *...>`

Where "`parameter_string`" is the string specified with this option. The default is "`_%s`" (the parameter name).

`[ -VALUE_format < LEC | RC | GENUS | DC > ]` is used to specify the "parameter value" format. By default the format is decided by `SET NAMING STYLE <lec | rc | genus | dc>`". You can use `-value_format` to control the format.

`-INSTANCE_Array <string>`

A string that specifies the instance array naming. The default is `"%s[%d]"`. The string must contain at least one `"%d"`. LEC will replace the first `"%s"` with the instance name and the first `"%d"` with the array index; all other `"%s"` and `"%d"` are kept as is.

`-Array_delimiter <left_string> <right_string>`

Two strings that specify the left and right array delimiter. *The default is "*`[`*" and "*`]`*" for the left and right string.*

`-Field_delimiter <left_string> <right_string>`

Two strings that specify the left and right record (for VHDL) or struct (for SystemVerilog) field delimiters. *The default is "*`[`*" and "*`]`*" for the left and right string.*

`-INTerface_delimiter <left_string> <right_string>`

Two strings that specify the left and right SystemVerilog interface item delimiter. The default is `"_"` and `""` for the left and right string.

For an example, refer to Case 2 in the Examples section.

`-INStance  [-ALL | -VLG | -VHDL] <condgen_string> <forgen_string> <instance_string>`

Specifies how to construct instance names inside `generate` constructs. Instance in this case refers to an rtl module instance instantiated as part of the design hierarchy.

There are two types of generate constructs: conditional generate constructs that include if-generate and case-generate blocks, and generate constructs that include for-generate blocks.

`condgen_string` specifies how to include if-generate and case-generate block names in the instance name.

`forgen_string` specifies how to include for-generate block names in the instance name, where:

`%L` specifies the block name.

`%s` specifies the next substitute string. It could be substituted by `condgen_string`, `forgen_string`, or `instance_string` (depends on the instance location).

`%d` specifies the block index.

`instance_string` specifies how to name instance names, where:

`%s` specifies the current instance name.

The default setting for `<condgen_string> <forgen_string> <instance_string>` is:

"`%L.%s`" "`%L[%d].%s`" "`%s`" for Verilog, Verilog2k, and SystemVerilog designs

"`%s`" "`%s_%d`" "`%s`" for VHDL designs

For an example, refer to Case 1 in the Examples section.

Note: According to the SystemVerilog LRM, unnamed generate constructs will be assigned a default generate block name. To control the auto-assigning, refer to the "`set naming rule -ENABLE_UNNAMED_blk_naming`"" command.

`-ALL` Specifies that this user-defined naming rule applies to Verilog/Verilog2k/SystemVerilog and VHDL designs. *This is the default.*

`-VLG` Specifies that this user-defined naming rule applies to only Verilog/Verilog2k/SystemVerilog designs.

`-VHDL` Specifies that this user-defined naming rule applies to only VHDL designs.

Note: The `-ALL`, `-VLG`, and `-VHDL` options must be used immediately after the `-instance` option.

`-VARiable [-ALL | -VLG | -VHDL] <condgen_string> <forgen_string> <variable_string>`

The variable naming scheme is similar to the instance naming scheme. Variable in this case refers to an rtl variable which could be modeled as a Conformal primitives such as DFF, DLAT.

`condgen_string` specifies how to include if-generate and case-generate block names in the variable name.

`forgen_string` specifies how to include for-generate block names in the variable name, where:

`%L` specifies the block name.

`%s` specifies the next substitute string. It could be substituted by`condgen_string`, `forgen_string`, or `variable_string` (depends on the variable location).

`%d` specifies the block index.

`variable_string` specifies how to specify the variable name, where:

`%s` specifies the current variable name.

The default setting for `<condgen_string> <forgen_string> <variable_string>` is:

"`%L.%s`" "`%L[%d].%s`" "`%s`" for Verilog, Verilog2k, and SystemVerilog designs

"`%s`" "`%s`" "`%s`" for VHDL designs

Note: According to the SystemVerilog LRM, unnamed generate constructs will be assigned a default generate block name. To control the auto-assigning, refer to the "`set naming rule -ENABLE_UNNAMED_blk_naming`" command.

For an example, refer to Case 2 in the Examples section.

	`-ALL` Specifies that this user-defined naming rule applies to Verilog/Verilog2k/SystemVerilog and VHDL designs. *This is the default.*
	`-VLG` Specifies that this user-defined naming rule applies to only Verilog/Verilog2k/SystemVerilog designs.
	`-VHDL` Specifies that this user-defined naming rule applies to only VHDL designs.
	Note: The `-ALL`, `-VLG`, and `-VHDL` options must be used immediately after the `-variable` option.
`-NOMDPORTflatten`	When synthesizing an HDL design, Conformal separates the ports of composite data types (such as structs, unions, records, or multi-dimension arrays) into multiple ports that are port-type compliant to Verilog 1995 (IEEE 1364-1995).
	For example, a multi-dimension array port will be separated into multiple ports and each port will be a vector. With `-MDPORTflatten` set, ports from the same composite data port will be concatenated into a single vector port.
	This is the default.
`-MDPORTflatten`	Flattens ports of composite data types (such as structs, unions, records, or multi-dimension arrays) into a single vector port. The vector range is `portSize`-1 down to 0 where `portSize` is the bit size of the original port.
	The default is ALL.
	ALL: Flattens ports of all composite data types and multi-dimension arrays into a vector port.
	`LOGIC_only`: Flattens logic multi-dimension array ports into a vector port.
	`RECORD_only`: Flattens struct and record type ports into a vector port.
	`UNION_only`: Flattens union type ports into a vector port.
`-NOMDPORT_BITblast`	Do not bit blast ports of a module. *This is the default*.
`-MDPORT_BITblast`	Bit blast the ports of composite data types or multi-dimension arrays not flattened into a vector port by `-MDPORTflatten` into multiple ports of a single bit.

`-ENABLE_UNNAMED_blk_naming`

By default, the tool renames unnamed blocks in accordance with Verilog-2005 LRM. For example:

```
module top #(parameter PARAM1 = 0, PARAM2 = 1) (
 input clk, input [1:0] datain,
 output [1:0] dataout
 );
 wire [1:0] dataout_int;
 generate
  begin
   if (PARAM1)
    assign dataout_int = 2'b0;
    else if (PARAM1 == 0)
     begin : label1
     reg [1:0] mem_mod;
     ...
     end
  end
 endgenerate
endmodule
```

The tool renames register `mem_mod` with `genblk1.label1.mem_mod_reg[0]` based on the Verilog-2005 LRM.

`-NOENABLE_UNNAMED_blk_naming`

Skip the unnamed block naming. For the given example, the register will be renamed to `label1.mem_mod_reg[0]` (without `genblk1`).

`-NOENABLE_PROC_name`	*This is the default*.
`-ENABLE_PROC_name`	When the naming style for the RTL design is set to Genus (through the command '`SET NAMING STYLE`'), the tool prefixes variable names in the VHDL design with the process name.

`-MAX_Para_mod_len <length_size>`

Specifies the maximum string length for parameterized module names. When you instantiate an existing module with new parameter values, the tool creates a new module and names it using the following renaming rule:

```
<orig_mod_name><<par_string><par value> *...>
```

The default length is 4096.

If the new module name is longer than the specified maximum length or longer than the default, the tool replaces it with a shorter, unique name. If a user-specified length is less than 32, the tool ignores this setting

`-STRIngformat` When a parameter value is a string, use this option to control whether the string interpreted as an ASCII binary number displayed in decimal or hexadecimal notation, or displayed in string format.

The parameter value is used to compose the module name when creating a new module for an instantiation of a parameterized module (`SET NAMING RULE -parameter`).

Note: This option works only when the `SET NAMING STYLE` is set to "`LEC`"

`DEFAULT`: *This is the default*. The string is converted into an ASCII binary number. For strings longer than 4 characters, the ASCII number is displayed in decimal base. Otherwise, the number is displayed in hexadecimal base. See Example section.

`HEX`: The string is converted to ASCII binary number and displayed in hexadecimal base.

`STR`: String is kept in string format.

`-DATAPATH_INSTANCE <"pattern_%d_$d">`

Specifies datapath instance mapping.

The `pattern_%d_%d` will be used to match the whole datapath instance name. The first %d represents the line number while the second means the column number. There can be 1 or more %d, but only the first 2 will be taken into account.

For example:

```
set naming rule -datapath_instance \

  "add_%d_%d" "mult_%d_%d"
set naming rule -datapath_instance \

  "DP_INCR_%d_x" -append

analyze hier_compare -eco_aware
```

The hierarchy mapping will map add_10_27 to add_12_27.

Note that datapath instance mapping can happen only when `SET NAMING RULE -DATAPATH_INSTANCE` is specified, and when `-eco_aware` is specified with `ANALYZE HIER_COMPARE`.

`-MOD_PARAM`	
	`USE_ANYWAY`: Compose parameterized module name with instance parameter data regardless whether the new value is the same as the default value or not. *This is the default setting*.
	`USE_WHEN_DIFF`: Compose parameterized module name with parameter data only if the new parameter data is different from the default value.
	`USE_ALL`: Apply all parameters even if instance statement does not specify parameter setting or the new data is the same as default values.
`-NOINSTARRAY_range_reset`	
	LEC will expand array instance into individual instance(s) and name the instance(s) based on "-INSTANCE_Array" setting.
	For the first %d in the "instance_array" setting, LEC will replace the first "%d" with original design array instance range declaration. *This is the default*.
`-INSTARRAY_range_reset < ASCEND | DESCEND >`	
	When "ASCEND" is used, LEC will replace the first "%d" with the array index start from 0 and end with array size -1.
	When "DESCEND" is used, LEC will replace the first "%d" with the array index start from array size -1 and end with 0.
`-Both`	The naming rule applies to both the Golden and Revised designs. *This is the default.*
`-Golden`	The naming rule applies to the Golden design alone.
`-REvised`	The naming rule applies to the Revised design alone.

Examples

Sample Usages:

```
set naming rule  -hierarchical_separator ":" -golden
set naming rule  -register "register_%s" -revised
set naming rule  -tristate "tristate_%s"
set naming rule  -array "<" ">" -golden
set naming rule  -instance  "%L.%s" "%L[%d].%s" "%s"
```

Interpreting Strings

The following illustrates how LEC interprets the following parameter string to generate a parameterized module name:

```
read design -sv test.v -root test -parameter -stri mystr "abc"
```

- By default, LEC generates the parameterized module name as "`test_mystr6382179`".
- When `-STRI hex` is used, LEC generate parameterized module name as "`test_mystrx616263`".
- When `-STRI str` is used, LEC generate parameterized module name as "`test_mystrabc`".

Case 1: Generating Instance Names

```
generate

if (1) begin: blkA    //if-generate

  for (j=0;j<=0;j=j+1) begin: forblkB   //for-generate
   or n1(a,b,c);
    for (k=23;k<=23;k=k+1) begin: forblkC //for-generate
      and n2(d,e,f);
    end
    begin : blkD  //if-generate
     nor n3 (g,h,i);
    end
   end
  end
endgenerate
```

For this example (**Case 1**):

- `set naming rule -instance  "%L.%s" "%L[%d].%s" "%s"`

  Renames instances `n1`, `n2` and `n3` as "`\blkA.forblkB[0].n1`", "`\blkA.forblkB[0].forblkC[23].n2`", and "`\blkA.forblkB[0].blkD.n3`", respectively. This is also the default setting for Verilog, Verilog2k, and SystemVerilog design.

  In this example, the block hierarchy for `n2` is:

  ```
  blkA(if-generate) => forblkB (for-generate) => forblkC (for-generate)
  ```

  The renaming process for n2 goes through the following steps:

  1.) For `blkA`, which is a if-generate, n2 is substituted by `condgen_string` "`%L_%s`". The result will be "`blkA.%s`".

  2.) For `forblkB`, which is a for-generate, `%s` of "`blkA_%s`" is substituted by `forgen_string` "`%L[%d].%s`". The result will be "`\blkA.forblkB[0].%s`".

3.) For `forblkC`, which is a for-generate, `%s` of "`\blkA.forblkB[0].%s`" is substituted by `forgen_string` "`%L[%d].%s`". The result will be "`\blkA.forblkB[0].forblkC[23].%s`".

4.) The `%s` of "`\blkA.forblkB[0].forblkC[23].%s`" is substituted by `instance_string` "%s". The final result will be "`\blkA.forblkB[0].forblkC[23].n2`"

- `set naming rule -instance "%s" "%s_%d" "%s"`

  Renames instances `n1`, `n2` and `n3` as `n1_0`, `n2_0_23`, and `n3_0`, respectively. This is also the default setting for VHDL design.

- `set naming rule -instance "%s" "%L[%d].%s" "%s_INS"`

  Renames instances `n1`, `n2` and `n3` as "`\forblkB[0].n1_INS` ", "`\forblkB[0].forblkC[23].n2_INS`", and "`\forblkB[0].n3_INS`", respectively.

Case 2: Generating SystemVerilog Interface Type Port/Variable Names

```
interface simple_bus;
  logic req, gnt;
  logic [7:0] addr, data;
endinterface: simple_bus

module memMod(simple_bus a, input bit clk);
  logic avail;
   always @(posedge clk) a.gnt <= a.req & avail;
endmodule
```

The tool generates port names `a_req`, `a_gnt`, `a_addr` and `a_data` for module `memMod`.

The following command renames ports `a_req`, `a_gnt`, `a_addr` and `a_data` as `\a.req`, `\a.gnt`, `\a.addr`, and `\a.data` respectively

```
set naming rule -INTerface_delimiter "." ""
```

Case 3: Generating Instance Array Names

```
input [4:0] in;
output [4:0] out;
wire  [4:0] out1;
sub s1 [4:0] (.in(out1), .out(out));
sub s2 [4:0] (.in(in), .out(out1));
```

For this example (Case 3):

- `set naming rule -instance_array "%s[%d]_%s_%s%d"`

  The tool generates the following instance names:

  `\s1[4]_%s_%s%d`

  `\s1[3]_%s_%s%d`

```
\s1[2]_%s_%s%d

\s1[1]_%s_%s%d

\s1[0]_%s_%s%d

\s2[4]_%s_%s%d

\s2[3]_%s_%s%d

\s2[2]_%s_%s%d

\s2[1]_%s_%s%d

\s2[0]_%s_%s%d
```

- `set naming rule -instance_array "_%s_%s"`

  The tool produces the following error because the rule does not contain a `%d` . The tool will generate the instance names using the default naming format:

  ```
  // Error: Illegal INSTANCE_Array naming _%s_%s.
  ```

  The tool generates the following instance names:

  ```
  \s1[4]

  \s1[3]

  \s1[2]

  \s1[1]

  \s1[0]

  \s2[4]

  \s2[3]

  \s2[2]

  \s2[1]

  \s2[0]
  ```

- `set naming rule -instance_array "%d_%s_%s_%d"`

  The tool generates the following instance names:

  ```
  \4_s1_%s_%d
  ```

`\3_s1_%s_%d`

`\2_s1_%s_%d`

`\1_s1_%s_%d`

`\0_s1_%s_%d`

`\4_s2_%s_%d`

`\3_s2_%s_%d`

`\2_s2_%s_%d`

`\1_s2_%s_%d`

`\0_s2_%s_%d`

- `set naming rule -instance_array "_%d_%d"`

  The tool generates the following instance names:

  `\_4_%d`

  `\_3_%d`

  `\_2_%d`

  `\_1_%d`

  `\_0_%d`

  `\_4_%d_1`

  `\_3_%d_1`

  `\_2_%d_1`

  `\_1_%d_1`

  `\_0_%d_1`

For design instance:

```
  DUT my_array_inst[5:4] ();


case 1 : set naming rule -NOINSTARRAY_range_reset
LEC will generate instance with name
  my_array_inst[5] and my_array_inst[4]


case 2 : set naming rule -INSTARRAY_range_reset  DESCEND
LEC will generate instance with name
```

```
  my_array_inst[1] and my_array_inst[0]
my_array_inst[1] corresponds to original my_array_inst[5].


case 3 : set naming rule -INSTARRAY_range_reset ASCEND
LEC will generate instance with name
  my_array_inst[0] and my_array_inst[1]
my_array_inst[0] corresponds to original my_array_inst[5].
```

Related Commands

ADD MAPPED POINTS

ADD RENAMING RULE

DELETE MAPPED POINTS

DELETE RENAMING RULE

MAP KEY POINTS

READ DESIGN

REPORT ENVIRONMENT

REPORT MAPPED POINTS

REPORT RENAMING RULE

REPORT UNMAPPED POINTS

SET NAMING STYLE

TEST RENAMING RULE

WRITE HIER_COMPARE DOFILE

SET NAMING STYLE

```
SET NAming Style
     <LEC | RC | GENUS | DC>
     [-Both | -Golden | -REVised]
     (Setup Mode)
```

Specifies the naming style for an RTL design. Execute this command before `READ DESIGN`.

This command handles standard naming (where the design uses only the default Genus synthesis attributes); it does not handle naming caused by non-default Genus synthesis attributes (which include, but are not limited to, attributes such as `inst_prefix` or `hdl_generate_index_style`). To handle those cases, you must also use the `ADD RENAMING RULE` or `SET NAMING RULE` commands.

Note: If you use the dofile generated by the Genus command **write_do_lec**, it already contains commands that handle standard renaming.

Tcl Command

```
set_naming_style
```

Parameters

`LEC`	Applies the LEC naming style to the design. *This is the default*.
`RC`	Applies the RTL Compiler naming style to the design.
`GENUS`	Applies the Genus naming style to the design.
`DC`	Applies the Design Compiler naming style to the design.
`-Both`	The naming style applies to both the Golden and Revised designs. *This is the default.*
`-Golden`	The naming style applies to the Golden design alone.
`-REvised`	The naming style applies to the Revised design alone.

Examples

The following example illustrates the difference between the naming styles.

The following design file contains one arithmetic operator(`*`) on line 7.

```
1 entity top is
2   port(A, B: in integer range 7 downto 0;
3           C: out integer range 64 downto 0);
4 end top;
5 architecture rtl of top is
6 begin
7   C <= A*B;
8 end rtl;
```

Using the LEC naming style, the instance name of the arithmetic operator will be `mult_7`.

Using the Genus naming style, the instance name of the arithmetic operator will be `mul_7_9`.

Using the DC naming style, the instance name of the arithmetic operator will be `mult_7`.

Related Commands

READ DESIGN

SET PARALLEL OPTION

```
SET PARAllel Option
     [-INFO]
     [-KEEP_DIR]
     [-KILL_COMMAND_LINE <string>]
     [-LICENSE <license_list>]
     [-LOG_FILE <filename>]
     [-MAX_Remote <integer>]
     [-MAX_THREADS <integer>]
     [-RELEASE_LICense | -NORELEASE_LICense]
     [-STARTING_PORT <integer>]
     [-SUBMIT_COMMAND_LINE <string>]
     [-THREADS [<integer>[,<integer>]][-RESERVE_LICENSE][-RETRY | -QUEUE]]
     [-TMPDIR <string>]
     [-WORKERS <localhost | batch | host >...]
     [-CLUSTER <LSF | SGE> ]
     [-BATCH_COMMAND <command>]
     [-RSH_COMMAND <command>]
     (LEC / Setup Mode)
```

Sets the parameters for parallel processing. Some options are used for multithreaded processing (recommended form of parallel processing), some are used for LSF-based parallel processing, and `-KEEP_DIR` is used for both.

The following options are used for multithreaded processing only:

- `-THREADS`
- `-MAX_THREADS`
- `-INFO`
- `-LOG_FILE`
- `-STARTING_PORT`
- `-LICENSE`
- `-TMPDIR`
- `-RESERVE_LICENSE`

For more information, see Multithreaded Processing in the *Conformal Equivalence Checking User Guide*.

The following options are used for LSF based parallel processing only:

- `-MAX_Remote`

- `-SUBMIT_COMMAND_LINE`
- `-KILL_COMMAND_LINE`

Tcl Command

`set_parallel_option`

Parameters

`-INFO`	Reports multithreaded processing related information. This includes host information (host name, number of processors, port, maximum number of licenses used in multithreaded processing, and machine average loads) and overhead information (spawned processes latency and multithreaded comparison overhead). This also reports the location of temporary directory and multithreaded processing log file, if applicable.
`-KEEP_DIR`	Specifies that if a temporary directory is created during parallel processing, this directory is preserved when the software exits. By default, this directory is deleted automatically. For more information, see Temporary Files and Directories in the *Conformal Equivalence Checking User Guide*.
`-KILL_COMMAND_LINE <string>`	
	Specifies the kill command interface. The default value is `bkill <jobid>`. The keyword `<jobid>` is determined by the software.
`-LICENSE <license_list>`	

Specifies the license, or a list of licenses, to use for multithreaded processing. The license specified in this list must be one of `XL`, `CCDXL`, `LP`, `LPGXL`, `GXL`, `ECO`, or `ECOGXL`.

For multiple licenses, the software checks out all required licenses from the first one on the list. If the first license is not available, it checks for the second license availability on the list, and so on.

License lists must be encapsulated in quotes. For example:

```
set parallel option -license "xl lp eco"
```

If this option is not specified, the following is the default:

```
XL <license_used_to_invoke_LEC> CCDXL LP LPGXL
ECO ECOGXL GXL
```

`-LOG_FILE <filename>`

Specifies the log file name of multithreaded processing.

`-MAX_Remote <integer>`

Specifies the maximum number of remotes used for performing the tasks. The number of available CPUs is considered as that number of remotes. However, this is just a recommendation. The Conformal software will check the number of available licenses and might launch remote jobs less than the number specified.

`-MAX_THREADS <integer>`

Specifies the maximum number of threads allowed in multithreaded processing.

`-RELEASE_LICense` Specifies that licenses checked out will be checked in after multithreaded processing is finished. *This is the default*.

`-NORELEASE_LICense` Specifies that licenses checked out will be held for later multithreaded processing. The licenses checked out will not be released until you use the `SET PARALLEL OPTION -release_license` command or exit the software.

`-STARTING_PORT <integer>`

Specifies the starting port for which the software will search for an available port for multithreaded processing.

`-SUBMIT_COMMAND_LINE <string>`

Specifies the submit command interface.

The default value is:

```
bsub -o <logdir>/<jobnum>_LSF.log
<submit_options> <command>
```

The keywords `<logdir>`, `<jobnum>`, and `<command>` are determined by the software.

You can specify `<submit_options>` with the `RUN PARALLEL COMPARE` command.

`-THREADS <integer>[,<integer>]`

Specifies the minimum and maximum number of threads, separated by comma. If only one number entered, this specifies both the minimum and maximum number of threads. For example '`-threads 2`' specifies two threads; '`-threads 2,4`' specifies a minimum of two threads, and a maximum of four threads.

`-RESERVE_LICENSE`
Immediately checks out and reserves the number of licenses required for multithreaded processing (if you specified a range of threads, the tool looks for the minimum number of required threads). The tool errors out if it cannot check out the required number of licenses. The licenses are reserved until released using the `-RELEASE_LICENSE` option.

Notes: This option requires that multithreading be enabled (through the `-THREADS` option).

`-RETRY`
Used with `-RESERVE_LICENSE`, repeat the license check until all licenses required to support the minimum number of threads are checked out. Press Ctrl_C to exit this procedure.

`-QUEUE`
Use with `-RESERVE_LICENSE`. Puts the license request in queue. This request will give the license when it is available. Press Ctrl_C to exit this procedure.

`-TMPDIR <string>`

Use this option to designate a location for the temporary directory that is created during multithreaded processing. If this option is not specified, the temporary directory is created under the current working directory.

Note: The `ANALYZE MODULE` command also uses the temporary directory created by the `SET PARALLEL PROCESSING` command, unless you designate a separate location using the `ANALYZE MODULE -tmpdir` command.

`-WORKERS <localhost | batch | host>`

Specifies the number of workers to use for the comparison and whether to launch them to the local machine or to a cluster of remote machines.

`-CLUSTER <LSF | SGE>`

Specifies to run the comparison on a cluster of remote machines and specifies the cluster type. LSF is for load sharing facility clusters and OpenLava, and SGE is for Sun grid engine clusters. LSF is the default.

Note: This requires a Smart LEC license.

`-BATCH_COMMAND <command> or -RSH_COMMAND <command>`

Specifies the command to submit jobs to the cluster.

Note: This requires a Smart LEC license.

Examples

The following examples are for multithreaded processing only:

- The following command sets the minimum and maximum number of threads to 3. As a result, the subsequent `COMPARE` command will use 3 threads:

```
set_parallel_option -threads 3
compare
```

- The following command specifies that LP licenses will be used during parallel processing. Then the `COMPARE -threads 3` command checks out two additional LP licenses for parallel comparison. By default, the software checks out two additional XL licenses.

```
set_parallel_option -license lp
```

- The following command specifies the list of licenses to use for parallel processing. If you have one LP and two ECO licenses, you can run parallel processing for up to four threads:

```
set_parallel_option -license "lp eco"
compare -threads 4
```

- If you have XL, LP, and ECO licenses, but do not know the exact usage of these licenses, you can use all available licenses to run parallel comparison. The following command first attempts to check out three XL licenses. If only one XL license is available, the software checks it out and attempts to check out two LP licenses. If this succeeds, the software runs parallel comparison using four threads with one XL license and two LP licenses. If only one LP license is available, the software checks it out and attempts to check out one ECO license. If this succeeds, the software runs parallel comparison with one XL, one LP, and one ECO licenses.

```
set_parallel_option -license "xl lp eco"
compare -threads 4
```

- The following command keeps the temporary directory created during parallel processing:

```
set_parallel_option -keep_dir
```

- The following command searches for an available port starting from port number 50100

```
set_parallel_option -starting_port 50100
```

The following examples are for LSF parallel processing only:

- The following command specifies that the maximum number of remote jobs submitted to the server farm is 8:

```
set_parallel_option -max_remote 8
```

- The following command specifies that the command used to submit jobs is `mybsub` and the command used to kill jobs is `mybkill`. By the default, the commands used to submit and kill jobs are `bsub` and `bkill`:

```
set_parallel_option -submit_command_line mybsub -kill_command mybkill
```

- The following command sets the minimum number of threads to 2 and maximum to 4. In this way, the software strives to use the maximum number of threads. If this does not succeed, it automatically reduces the number of threads according to the number of available licenses instead of existing with an error, unless it cannot run with the minimum number of threads:

```
set_parallel_option -threads 2,4
```

- The following commands specify that licenses will be held during mutlithreaded processing in hierarchical comparison to avoid frequent license check out and check in:

```
set_parallel_option -norelease_license
set_parallel_option -threads 4
dofile hier_compare.do
```

- Resourcing Workers on the Current Machine
  To run the comparison using your local machine, use the `-workers` option with the keyword "`localhost`" and specify it for however many workers you want to assign. The example below specifies that the parallel hierarchical comparison will use 4 threads and 4 workers on the current machine:

```
set_parallel option -threads 4
set_parallel option -workers localhost localhost localhost localhost
```

- Resourcing Workers from a Cluster
  To run the comparison on a cluster of remote machines, use the `-workers` option with the keyword "`batch`" and specify it for however many workers you want to assign to the cluster, specify the cluster type using the `-cluster` option, and the batch command to submit jobs using the -batch option. For example:

```
set_parallel_option -threads 4
set_parallel_option -workers batch batch batch batch
set_parallel_option -cluster LSF
set_parallel_option -batch_command \
\"/farm/bin/bsub -q super -R 'OSNAME==Linux && SFIARCH==EM64T\"
```

  The above specifies that the parallel hierarchical comparison will use 4 threads and 4 workers from an LSF cluster.

  You can mix `localhost` and `batch` in worker setting. For example:

```
set_parallel_option -workers batch batch localhost localhost batch localhost
```

- Viewing Current Resource Settings
  To report the current resource settings, use the `set_parallel_option` command without any options.

```
// Command: set_parallel_option
Current parallel processing options: ('*' indicates non-default value.)
================================================================================
Keep directory : NO
Multithreaded processing options:
--------------------------------------------------------------------------------
Number of threads : 0,0
Starting port : 50100
License list : SL4
Hold license : NO
Distributed parallization options:
--------------------------------------------------------------------------------
Workers : localhost localhost localhost localhost
Cluster type : LSF
Batch command : * bsub -q lnx64
Rsh commmand : rsh
```

- To check default setting, use the `set_parallel_option` command without any options. LEC will print this:

```
// Command: set_parallel_option
Current parallel processing options: ('*' indicates non-default value.)
```

```
================================================================================
Keep directory : NO
Multithreaded processing options:
--------------------------------------------------------------------------------
Number of threads : 0,0
Starting port : 50100
License list : SL4
Hold license : NO
Distributed parallization options:
--------------------------------------------------------------------------------
Workers : localhost localhost localhost localhost
Cluster type : LSF
Batch command : * (null)
Rsh commmand : rsh
```

Related Commands

SET DATAPATH OPTION

WRITE HIER_COMPARE DOFILE

ANALYZE DATAPATH

SET PARAMETER

```
SET PArameter
     [ <-MODule <moduleName>|-INTERface <interfaceName>>
     [-Parameter  [-INT |-STR|-ENUM] <paramName> <paramValue>]]
     [-Golden |-Revised |-Both]
     (Setup Mode)
```

Use this command before the `READ DESIGN`, `READ LIBRARY`, and `ELABORATE DESIGN` commands to override default parameter values assigned by source files. If a command has more than one parameter definition, Conformal uses the last parameter.

To specify multiple modules, interfaces, or parameters, you must use the corresponding option for each module, interface, or parameter that you want to set. For example:

```
set parameter -mod m1 -p param1 value1 -mod m2 -p param2 value2 \
  -mod m2 -p param3 value3 -p int param4 value4
```

Tcl Command

```
set_parameter
```

Parameters

`-MODule <moduleName>`	Applies the parameter setting to the specified module.
`-INTERface <interfaceName>`	Applies this parameter setting to the specified interface.
`-Parameter [-INT |-STR|-ENUM] <paramName> <paramValue>`	Assigns module/interface parameters or replaces existing design/interface parameters. When using "`-Parameter -INT <paramName> <paramValue>`", `<paramValue>` is converted to an integer value, which can be a positive integer (1), negative integer (-1), an integer value recognized as a string ("1"/"-1"), or a Verilog style integer ("16'h0001"). When using a Verilog style integer, the value must be specified between double- quotes (" ").

	When using `-Parameter -STR <paramName> <paramValue>`, the `<paramValue>` will be saved as a string. When using `-Parameter -ENUM <paramName> <paramValue>` command, the `<paramValue>` is converted to a VHDL/SystemVerilog enumeration literal. For example, the following command sets the parameter `P4` to VHDL/SystemVerilog enumeration literal `GREEN`: `set parameter -MOD m1 -P -enum P4 GREEN` Note: Any value that is not recognized as an unsigned decimal integer value is interpreted as a string value. Note: If `-INT` or `-STR` is not specified, then the parameter value will be interpreted as an integer if it is not between double-quotes (" "), and as a string if it is between double-quotes. Therefore, if you want to specify a Verilog format value, it must be between double-quotes and used with the `-INT` option.
`-Both`	Use this parameter setting for both the Golden and Revised designs.
`-Golden`	Applies to the Golden design.
`-Revised`	Applies to the Revised design.

Related Commands

ELABORATE DESIGN

READ DESIGN

SET PATTERN MATCH

```
SET PATtern Match
     [-Pattern <pattern_name> -MOdule <module_name ...>
       | -MAPping_file <mapping_file>]
     [-Golden | -Revised]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Sets the pattern matching modules.

Tcl Command

```
set_pattern_match
```

Parameters

`-Pattern <pattern_name>`	Specifies that the name of the pattern (that was read in with the `READ PATTERN` command) that will apply to the module(s).
`-MOdule <module_name ... >`	Specifies the module(s) to apply to the pattern.
`-MAPping_file` <mapping_file>	Specifies the file that contains the pairs of patterns and modules for pattern match. The format of the mapping file is `<pattern_name> <module_name>`.
`-Golden`	Sets the pattern match on the Golden design.
`-Revised`	Sets the pattern match on the Revised design.

Related Commands

READ PATTERN

SET ABSTRACT MODEL

SET PROJECT NAME

```
SET PRoject Name
     <project name>
     [-DIRectory <directory name>]
     [-REAd_only]
     (Setup Mode)
```

This command specifies the LEC project that will collect and consolidate various information for multiple LEC runs.

Tcl Command

```
set_project_name
```

Parameters

`project_name`	Specifies a project name. The project name cannot contain special characters (such as a space or asterisk). The tool creates a directory called *`<project_name>`*`.cfm`, where the tool will store all the information from all the LEC runs.
`-DIRectory <directory name>`	Specifies a working directory for the project. The tool will create the *`<project_name>`*`.cfm` directory in this directory. If a directory is not specified, the tool uses the current directory.
`-REAd_only`	Specifies read-only mode. When this is set, a run directory is not created for the current run and only informational queries are allowed for existing runs.

Related Commands

ANALYZE PROJECT

DELETE PROJECT

SET PROJECT OPTIONS

SET PROJECT PROPERTY

SET PROJECT OPTIONS

```
SET PRoject Options
     [-MAX_MEM <integer>]
     [-MAX_RUNS <integer>]
     [-BLOCK_DB_OUTPUT]
     [-DISABLE_RTL_RULE_REPORTS | -NODISABLE_RTL_RULE_REPORT]
     [-RESTORE_PROJECT]
     (Setup Mode)
```

This command specifies the LEC project options.

Tcl Command

```
set_project_options
```

Parameters

`-MAX_MEM <integer>`	Specifies the maximum memory in MB for the project. If this limit is exceeded, old run directories are automatically deleted. This default value is 10240MB.
`-MAX_RUNS <integer>`	Specifies the maximum number of run directories for the project. If this limit is exceeded, old run directories are automatically deleted. The default value is 100.
`-BLOCK_DB_OUTPUT`	Does not write large and time-consuming information into LEC project DB.
`-DISABLE_RTL_RULE_REPORT`	Disable saving of RTL rule report information in the project directory. By default, RTL rule report information is saved in the project directory.
`-NODISABLE_RTL_RULE_REPORT`	To re-enable saving of rtl rule report information, re-invoke the command with `-NODISABLE_RTL_RULE_REPORT` option.
`-RESTORE_PROJECT`	Specifies restoring the project in subsequent restarted LEC sessions from checkpoint files.

Related Commands

DELETE PROJECT

SET PROJECT NAME

SET PROJECT PROPERTY

SET PROJECT PROPERTY

```
SET PRoject Property
     [-Run <integer>]
     <keyword> <value>
```
(Setup/LEC Mode)

This command adds user-specified properties to current run or user specified previous run.

Tcl Command

```
set_project_property
```

Parameters

`-Run <integer>`	Specifies the run to which to add the property. The default is current run.
`<keyword>`	Specifies the name of the property to be added.
`<value>`	Specifies the value of the property to be added.

Example

The following command specifies that the current run is the "`reference`" run, and sets its property to 1. By doing so, this run is not subject to automatic removal.

```
set project property reference 1
```

For example, the `SET PROJECT OPTIONS` command can specify the maximum runs and maximum memory usage for the project. When either is exceeded, the tool automatically removes the old run directories. If you set a run's project property to 1, it is not subject to this type of automatic removal.

Related Commands

DELETE PROJECT

SET PROJECT NAME

SET PROJECT OPTIONS

SET PROJECT PROPERTY

SET RETIMING OPTION

```
SET REtiming Option
     [-NOAUTO | -AUTO]
     [-NORETIMED_MODULE | -RETIMED_MODULE]
     (Setup Mode)
```

Note: This requires a Conformal XL license.

Specifies whether the Conformal software automatically performs retiming analysis for designs when switching from Setup to LEC mode. According to the structure of the designs, the software automatically determines whether to use forward or backward pipeline retiming, or general retiming. If sequential correspondence information is available, the software uses general retiming except for cases where all the registers are on the PO or PI side for the Golden or Revised design, in which case the software uses forward or backward pipeline retiming. The analysis results enable the software to automatically resolve retiming designs.

Tip

Use the `ADD MODULE ATTRIBUTE` command to attach the `PIPELINE_Retime` attribute to a module.

Tcl Command

```
set_retiming_option
```

Parameters

`-NOAUTO`	Do not automatically analyze retiming when switching from Setup to LEC mode. *This is the default.*
`-AUTO`	Automatically analyzes retiming when switching from Setup to LEC mode.
`-NORETIMED_MODULE`	Performs retiming analysis on all modules with or without the `PIPELINE_Retime` attribute. *This is the default.*
`-RETIMED_MODULE`	Performs retiming analysis only on modules with the `PIPELINE_Retime` attribute.

Related Commands

ADD MODULE ATTRIBUTE

ANALYZE RETIMING

SET ROOT MODULE

```
SET ROot Module
     <module_name>
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Specifies the name of the root module for the Golden and Revised designs. The system default specifies that when the design is read, Conformal automatically assigns the root module. Thus, the `SET ROOT MODULE` command overrides the automatic assignment.

Use the `REPORT ENVIRONMENT` command to display the settings for the root module for the Golden and Revised designs.

Tcl Command

```
set_root_module
```

Parameters

`<module_name>`	This module is the root. This assignment overrides the automatic root module assignment Conformal makes when you use the `READ DESIGN` command.
`-Golden`	Assigns the root module name for the Golden design. *This is the default.*
`-Revised`	Assigns the root module name for the Revised design.
`-Both`	Assigns the root module name for both the Golden and Revised designs.

Related Commands

READ DESIGN

REPORT ENVIRONMENT

SET RTL TYPE

```
SET RTl Type
     <-V1995|-VERILOG2K|-SYStemverilog> <file_extension* ...>
     (Setup Mode)
```

Specifies the Verilog language standard for the file extensions. When specifying `-MIXvlog` with the `READ DESIGN` command, the file extension will be used to determine the Verilog language standard to parse the Verilog file. The default file extensions for Verilog standards are as follows:

- Verilog 1995: .v95, .v95p
- Verilog 2K: .v, .vp, .v2k
- SystemVerilog: .sv, .svp

Tcl Command

```
set_rtl_type
```

Parameters

`-V1995`	Specifies a Verilog-1995 design.
`-VERILOG2k`	Specifies a Verilog2k design.
`-SYStemverilog`	Specifies a SystemVerilog design.
`<file_extension>`	Specifies the file extension for specified Verilog standard. The file extension must include "." and is case-insensitive.

Examples

The following example demonstrates how to specify Verilog design standard for file extensions.

```
SET RTL TYPE -V1995 .v95 .v95p .vg .gv
SET RTL TYPE -VERILOG2K .v .vp .v2k
SET RTL TYPE -SYStemverilog .sv .svp .vs
```

Related Commands

READ DESIGN

SET RULE FILTER

```
SET RUle Filter
     <-ROOT_HIER_only>
     [-Golden | -Revised]
     (Setup Mode)
```

Filters out rules that occur in modules outside the root hierarchy. It is a means to remove unnecessary rule reporting and focus only on the root module's hierarchy.

Use the `REPORT RULE CHECK` command with the `-summary` option to display all of the rules and their settings and occurrences.

Tcl Command

```
set_rule_filter
```

Parameters

`-ROOT_HIER_only`	Filters out rules that occur in modules outside the root hierarchy.
`-Golden`	Applies the filter to the Golden design and library. *This is the default.*
`-Revised`	Applies the filter to the Revised design and library.

Related Commands

READ DESIGN

READ LIBRARY

REPORT RULE CHECK

SET RULE HANDLING

```
SET RUle Handling
     <[<rule_name* ...> [-Warning | -Error [-CONTinue] | -Ignore | -Note]]
       [<-EXCLude | -INCLude> <-MODule [-SUBModule] | -DESIGN_FILE | -LIB_FILE>
     <name* ...>]>
     [-LIMit <max>]
     [-SUMMARY_only]
     [-ON_ERROR <STOP | CONTinue <[BBOX | NO_BBOX]>>]
     [-ENVVAR_filepath <ON|OFf>]
     [ |-Design | -Library]
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Specifies the rule handling when reading in the designs and libraries or exclude the specified module, design file, library file, or rule from rule checking. Most rules are either warnings or notes. Execute this command before `READ LIBRARY` and `READ DESIGN`.

Note: Multiple `SET RULE HANDLING` commands can be specified, and the effects are cumulative.

Use the `REPORT RULE CHECK` command with the `-summary` option to display all of the rules and their settings and occurrences.

See the *Conformal Equivalence Checking User Guide* for rule definitions and sample cases.

Note: The wildcard (*) represents any zero or more characters in rule names.

Tcl Command

```
set_rule_handling
```

Parameters

`<rule_name* ...>`	Changes rule handling for the specified rules. This accepts wildcards.	
	`-Warning`	The rule handling will be a warning message. *This is the default.*

	`-Error [-CONTinue]`	
		The rule handling will be an error message. The `-continue` option can be used on RTL-related rules or tasks to indicate that the program continue to run instead of erroring out.
	`-Ignore`	The rule handling will be ignore. See REPORT RULE CHECK to see how this severity level affects reporting.
	`-Note`	The rule handling will be a note.
`-EXCLude | -INCLude`		
	Removes the unwanted rules. If neither of these options are specified, all rules will be checked by default.	
	`-EXCLude` removes some rules in the specified list.	
	`-INCLude` removes all rules not in the specified list.	
	`-MODule`	Removes the unwanted rules for the specified module.
	`-SUBModule`	Also removes the unwanted rules for submodules and decendant modules of the specified module. Note that if a submodule or decendant module is also part of other module hierarchy, the rule violations will still be reported for that module hierarchy.
	`-DESign_file`	Removes the unwanted rules for the specified design file.
	`-LIB_file`	Removes the unwanted rules for the specified library file.
	`<name* ...>`	Specifies the name of the module, design file, or library file.
`-LIMit <`*`max`*`>`	Limits the number of occurrences of the specified rules.	
	`<`*`max`*`>` is the limit after which no more occurrences will be recorded. Changing this limit will only have effect the next time the specified rules are checked.	
`-SUMMARY_only`	For rules with severity WARNING or lower, report the rule check occurrences only at the end of elaboration	

`-ON_ERROR`	Specifies what to do if the rule handling results in an error. `STOP`: Report the first error and then error out. `CONTINUE`: Try to continue to parse, but do not error out, with `BBOX`: Issue an error, and blackbox the module. `NO_BBOX`: Issue an error, do not blackbox the module.
`-ENVVAR_filepath`	If the filepath contains environment variable components, use the environment variable when reporting the filepath. This option affects ’report rule check’, ’write rule check’, and ’read rule check’ commands. ON: Turn on the option OFf: Turn off the option
`-Design`	Applies the rule handling to only the designs. If you do not specify `-design` or `-library`, Conformal applies the rule handling to both designs and libraries.
`-Library`	Applies the rule handling to only the libraries. If you do not specify `-design` or `-library`, Conformal applies the rule handling to both designs and libraries.
`-Both`	Applies the rule handling to both the Golden and Revised designs and libraries. *This is the default.*
`-Golden`	Applies the rule handling to the Golden design and library.
`-Revised`	Applies the rule handling to the Revised design and library.

Examples

The following command sets the severity level of RTL7.16 from error to warning for the Golden design:

```
set rule handling RTL7.16 -warning -design -golden
```

The following command sets the severity level of RTL7.16 from error to warning for all the Golden design and removes RTL7.16 from the rule check for the design file top.v

```
set rule handling RTL7.16 -warning -exclude -design_file top.v -design -golden
```

The following command sets the severity level of RTL7.16 from error to warning and only checks the specified rule RTL7.16 for the design file top.v.

```
set rule handling RTL7.16 -warning -include -design_file top.v -design -golden
```

Related Commands

READ DESIGN

READ LIBRARY

REPORT RULE CHECK

SET RUNTIME LIMIT

```
SET RUntime Limit
     [-MODule [module_name] <time>]
     [-HIER_compare <time> <filename>]
     [-QUICK_COMPARE | -NOQUICK_COMPARE]
     [-COMmand <command_name> <-EACH <time> | -TOTAL <time>>]
     (Setup Mode)
```

Set the elapse time limit for the specified module, `hierarchical_compare` and commands. The unit of the time is second..

Tcl Command

`set_runtime_limit`

Parameters

`-MODule`	Limits the runtime of LEC in the specified module. If the module name is not specified, the runtime limit will be applied in the root module.
`-HIER_compare`	Distributes the total runtime limit of all the modules. The file with the runtime limit setting, writes out after the command `write_hier_compare`.
`-QUICK_COMPARE`	When the runtime limit is reached, LEC shows the comparison result according to the current mapping and modeling result. *This is the default*.
`-COMmand`	Sets the runtime limit for the specified command. It supports `MAP_KEY_POINTS`, `ANALYZE_DATAPATH`, `ANALYZE_ABORT`, `ANALYZE_SETUP`, and `COMPARE`.
`-EACH`	The runtime limit is applied to each specified command.
`-TOTAL`	The total runtime of the specified command will be limited when the command is used multiple times.

Related Commands

DELETE RUNTIME LIMIT

REPORT RUNTIME LIMIT

SET SCREEN DISPLAY

```
SET SCreen Display
     <ON | OFf>
     [-PROGRESS | -NOPROGRESS]
     (Setup / LEC Mode)
```

Specifies whether the transcript output is displayed on the terminal screen.

Tip

If the screen display is set to off, use the `SET LOG FILE` command to save the transcript to a file.

Use the `REPORT ENVIRONMENT` command to display the setting for the screen display. *By default, screen display is on.*

Tcl Command

```
set_screen_display
```

Parameters

`ON`	Displays the transcript on the terminal screen. *This option is the system default.*
`OFf`	Do not display the transcript on the terminal screen.
`-PROGRESS`	Displays the percentage of completion on the terminal screen. *This option is the system default.*
`-NOPROGRESS`	Do not display the percentage of completion.

Related Commands

REPORT ENVIRONMENT

SET LOG FILE

SET SMARTLEC OPTION

```
SET SMartlec Option
     [-AUTO]
     (Setup / LEC Mode)
```

When you use the `-AUTO` option, it automatically enables the recommended Smart LEC features.

Tcl Command

```
set_smartlec_option
```

Parameters

`-AUTO`	Enable the recommended Smart LEC features.

Related Command

SET SPICE OPTION

```
SET SPIce OPTion
     [-BBOX | -NOBBox]
     [-BUlk | -NOBUlk]
     [-NOADDGLOBALPINs | -ADDGLOBALPINs]
     [-NOKEep_InstanceX | -KEep_InstanceX]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Specifies options for reading the SPICE netlist design (when running the `read design -spice` command).

Tcl Command

`set_spice_option`

Parameters

`-BBOX`	Specifies that `SUBCKT` contains no transistors and will be treated as a blackbox. *This is the default.*
`-NOBBox`	Specifies that `SUBCKT` contains no transistors and will be removed along with all of its instantiations.
`-BULK`	Identifies nets connected to PMOS bulk terminals as power and nets connected to NMOS bulk terminals as ground. *This is the default.*
`-NOBULK`	By default, Conformal GXL identifies nets connected to PMOS bulk terminals as power and nets connected to NMOS bulk terminals as ground. However, this option removes that assumption and you will need to specify power and ground pins/nets by using `*.GLOBAL <name>:P` for power nets and `*.GLOBAL <name>:G` for ground nets or use Conformal commands to add constraints, tied signals (pins), or net attributes.
`-NOADDGLOBALPINs`	Specifies that no extra ports for `GLOBAL` signals will be created for `SUBCKT`. *This is the default.*
`-ADDGLOBALPINs`	Specifies that extra ports for `GLOBAL` signals will be created for `SUBCKT`.

`-NOKEep_InstanceX`	Specifies that the first character 'X' of the name of instance will not be retained. *This is the default.*
`-KEep_InstanceX`	Specifies that the first character 'X' of the name of instance will not be retained.

Examples

Sample Dofile:

```
set spice option -nobulk -nobbox -addglobalpins -keep_instanceX
read design -spice library.spi -golden
abstract logic
```

Related Commands

ABSTRACT LOGIC

READ DESIGN

SET STATETABLE

```
SET STATEtable
     <ON | OFf>
     (Setup Mode)
```

Controls the global setting of the Synopsys Liberty state table support.

Note: This command must be used before `READ DESIGN` and `READ LIBRARY`.

Note: Using the `READ DESIGN` and `READ LIBRARY` command's `-STATEtable` option supersedes these settings (it also supersedes the global setting).

Tcl Command

`set_statetable`

Parameters

`ON`	Enables support for Synopsys Liberty state tables. *This option is the system default.*
`OFf`	Disables support for Synopsys Liberty state tables.

Related Commands

READ DESIGN `-statetable`

READ LIBRARY `-statetable`

SET SYNTHESIS_OFF_COMMAND

```
SET SYNTHESIS_OFF_Command
     <string>
     (Setup Mode)
```

Specifies the pragma that is used to indicate the beginning of non-synthesizable constructs in the source code or in the generated generic netlist.

Note: If you do not run the `SET_ATTR INPUT_PRAGMA_KEYWORD` command prior to running this command, the default is `translate_off`.

Note: If this command is run multiple times, only the last value is used.

Tcl Command

`set_synthesis_off_command`

Parameters

`<string>`	Specifies the action. *Default*: `translate_off` `synthesis_off`

Examples

Sample Dofile:

```
set_attr input_pragma_keyword rtl
set synthesis_off_command turn_off
set synthesis_on_command turn_on
```

After running these three commands, the Conformal and VHDL parsers will recognize the pragmas in the VHDL and Verilog Source files.

In a VHDL file, the code between `-- rtl turn_off` and `-- rtl turn_on` will not be synthesized.

In a Verilog file, the code between `// rtl turn_off` and `// rtl turn_on` will not be synthesized.

Related Commands

SET_ATTR INPUT_PRAGMA_KEYWORD

SET SYNTHESIS_ON_COMMAND

SET SYNTHESIS_ON_COMMAND

```
SET SYNTHESIS_ON_Command
     <string>
     (Setup Mode)
```

Specifies the pragma that is used to indicate the end of non synthesizeable constructs in the source code or in the generated generic netlist.

Note: If you do not run the `SET_ATTR INPUT_PRAGMA_KEYWORD` command prior to running this command, the default is `translate_on`.

Note: If this command is run multiple times, only the last value is used.

Tcl Command

`set_synthesis_on_command`

Parameters

`<string>`	Specifies the action. *Default*: `translate_on` `synthesis_on`

Examples

Sample Dofile:

```
set_attr input_pragma_keyword rtl
set synthesis_off_command turn_off
set synthesis_on_command turn_on
```

After running these three commands, the Conformal and VHDL parsers will recognize the pragmas in the VHDL and Verilog Source files.

In a VHDL file, the code between `-- rtl turn_off` and `-- rtl turn_on` will not be synthesized.

In a Verilog file, the code between `// rtl turn_off` and `// rtl turn_on` will not be synthesized.

Related Commands

SET_ATTR INPUT_PRAGMA_KEYWORD

SET SYNTHESIS_OFF_COMMAND

SET SYSTEM MODE

```
SET SYstem Mode
     <Setup | LEc [-Map | -Nomap] [-PRESERVE]>
     (Setup / LEC )
```

Switches system modes between the Setup mode and the LEC mode.

- Setup mode: While in this mode, you read in the design and set all of the necessary constraints and environment variables.
- LEC mode: While in this mode, Conformal does the comparison and diagnosis.

When you exit the Setup mode, Conformal attempts to map all key points in the Golden and Revised designs. A summary is given for the mapped points in the Golden and Revised designs. An additional summary is given if Conformal identifies any unmapped key points.

Use the `REPORT ENVIRONMENT` command to display the current system mode.

Tcl Command

```
set_system_mode
```

Parameters

`Setup`	Switches the system mode to Setup.
`LEc`	Switches the system mode to LEC.
`-Map`	Maps key points when entering the LEC system mode. *This is the default.*
`-Nomap`	*Do not* map key points when entering the LEC system mode.
`-PRESERVE`	Preserves the netlist modeling. You can use this option for netlists created with the `REPORT TESTCASE` command only.

Related Command

REPORT ENVIRONMENT

REPORT TESTCASE

SET UDP PIN

```
SET UDp Pin
     <udp_name> <pin_name <0 | 1> ...>
     [-Golden | -Revised | -Both]
     (Setup Mode)
```

Sets the pin inputs of the user-defined primitive (UDP) to constant values, which are propagated into the UDP. Some inactive entries will be removed.

The constant set to inputs simplifies the state table of the UDP, which will sometimes eliminate some ambiguity in the UDP description.

Note: This must be used before running the `READ DESIGN` command.

Tcl Command

```
set_udp_pin
```

Parameters

`<udp_name>`	Specifies the name of the UDP.
`<pin_name> <0 | 1>`	Specifies the name of the pin and its input value. Choose `0` or `1`.
`-Golden`	Sets the UDP inputs in the Golden design. *This is the default.*
`-Revised`	Sets the UDP inputs in the Revised design.
`-Both`	Sets the UDP inputs in both the Golden and Revised designs.

Example

In this example, pins `in1` and `in2` of the UDP named `udp_1` are set to `0` and `1`, respectively:

```
set udp pin udp_1 in1 0 in2 1
```

Related Commands

READ DESIGN

READ LIBRARY

SET UNDEFINED CELL

```
SET UNDEfined Cell
     <Error | Black_box>
     [-ASCEND | -NOASCEND]
     [-AUTO_Assign | -NOAUTO_Assign]
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Specifies how Conformal handles undefined cells it encounters when reading in the Golden and Revised designs. *The system default is to give an error message if there are any undefined cells.*

Use the `REPORT ENVIRONMENT` command to display the settings for the undefined cells handling for the Golden and Revised designs. Execute this command before `READ LIBRARY` and `READ DESIGN`.

Note regarding the direction of blackbox pins: Because blackboxes are undefined, the direction of their pins is automatically inferred from connectivities in the blackbox instance's parent module. For example, if the net associated with pin `d[31]` of the blackbox instance has a known driver (such as, the net is assigned by a continuous assignment or the net is connected to an instance's output pin), the pin `d[31]` is a blackbox input pin. If no drivers are found for the pin `d[31]`, the pin `d[31]` is a blackbox output pin.

Tcl Command

```
set_undefined_cell
```

Parameters

`Error`	When reading the designs, undefined cells trigger an error message. *This is the default.*
`Black_box`	When reading the designs, regards undefined cells as blackboxes. This option takes Verilog parameter values to construct the module name for the created blackbox module.
`-ASCEND`	Arranges bits of bus pins in ascending order; that is, in1 (0 to 7). *This is the default.*
`-NOASCEND`	Arranges bits of bus pins in descending order; that is, out1 (7 down to 0).

`-AUTO_Assign`	Automatically determines and assign directions to all blackbox pins. *This is the default.* **Note:** If Conformal cannot determine the direction of a pin as `input` or `output`, it assigns I/O direction.
`-NOAUTO_Assign`	Assigns I/O direction to all blackbox pins. **Note:** You must manually reassign all pin directions according to the design.
`-Both`	The specified handling for undefined cells applies in both the Golden and Revised designs. *This is the default.*
`-Golden`	The specified handling for undefined cells applies in only the Golden design.
`-Revised`	The specified handling for undefined cells applies in only the Revised design.

Related Commands

ADD BLACK BOX

ADD NOTRANSLATE MODULES

READ DESIGN

READ LIBRARY

REPORT ENVIRONMENT

SET UNDEFINED PORT

```
SET UNDEfined Port
     <Error | Ignore>
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Specifies how Conformal handles undefined ports it encounters when reading in the Golden and Revised libraries and designs. *The system default is to report an error message if there are any undefined ports referenced by the module instance.*

Use the `REPORT ENVIRONMENT` command to display the settings for the undefined ports handling for the Golden and Revised designs. Execute this command before `READ LIBRARY` and `READ DESIGN.`

Tcl Command

```
set_undefined_port
```

Parameters

`Error`	Displays an error message for undefined ports.
`Ignore`	Ignores undefined ports.
`-Both`	Applies the specified handling for undefined ports in both the Golden and Revised designs. *This is the default.*
`-Golden`	Applies the specified handling for undefined ports to the Golden design.
`-Revised`	Applies the specified handling for undefined ports to the Revised design.

Related Commands

READ DESIGN

READ LIBRARY

REPORT ENVIRONMENT

SET UNDRIVEN SIGNAL

```
SET UNDRiven Signal
     < Z | 0 | 1 | X >
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Specifies globally how Conformal treats undriven signals in the Golden and Revised designs. *The system default specifies that undriven signals are classified as high-impedance (always driven by Z) in the Golden and Revised designs.*

Use the `REPORT ENVIRONMENT` command to display the settings for the undriven signals for both the Golden and Revised designs. This command has to be specified before `SET SYSTEM MODE LEC` command if running flat comparison or `WRITE HIER COMPARE` command if running hierarchical comparison.

Tcl Command

```
set_undriven_signal
```

Parameters

`Z`	Specifies undriven signals as high impedance (always driven by Z). *This option is the system default.*
`0`	Specifies undriven signals as Logic 0.
`1`	Specifies undriven signals as Logic 1.
`X`	Specifies undriven signals as X assignments.
`-Both`	Applies the state of the undriven signal to both the Golden and Revised designs. *This is the default.*
`-Golden`	Applies the state of the undriven signal to the Golden design.
`-Revised`	Applies the state of the undriven signal to the Revised design.

Related Command

REPORT ENVIRONMENT

SET VERIFICATION INFORMATION

```
SET VErification Information
     <verification_information_directory>
     [-RECord <ALL>|
      <HIER_COMpare | ANALYZE_ABOrt | ANALYZE_DATapath | MAPping> ...]
     [-NORECord <ALL>|
      <HIER_COMpare | ANALYZE_ABOrt | ANALYZE_DATapath | MAPping> ...]
     [-SEtup <ALL>|
      <HIER_COMpare | ANALYZE_ABOrt | ANALYZE_DATapath | PHAse | MAPping |
       SEQ_CONST | SEQ_MERGE> ...]
     [-NOSEtup <ALL> |
      <HIER_COMpare | ANALYZE_ABOrt | ANALYZE_DATapath | PHAse | MAPping |
       SEQ_CONST|SEQ_MERGE> ...]
     (Setup/LEC Mode)
```

Enables the usage of verification information in the current run and uses it to help optimize verification. Verification information from the current run can only be written out (`WRITE VERIFICATION INFORMATION`) when `SET VERIFICATION INFORMATION` is enabled.

Tcl Command

```
set_verification_information
```

Parameters

`verification_information_directory`	
	Name of the directory from which to read the verification information. Unless any categories are enabled/disabled specifically (by the `-setup` and `-nosetup` options), Conformal uses the `hier_compare`, `analyze_abort`, datapath analysis, and `phase` categories of information by default (described below).
`-RECord`	Specifies the categories of second run information to be recorded for future LEC run.
`-NORECord`	Disables the categories of second run information to be recorded.
`-SEtup`	Specifies the categories of verification information to use to help setup the current LEC run.

`-NOSETup`	Disables the usage of verification information to help setup the current LEC run. You can disable all information or specific categories of information. `ALL`: Enable or disable all categories of information. `HIER_COMpare`: Hierarchical comparison information from previously captured verification information to avoid dynamic flattening. This feature is described further in a web interface document titled *Second Run Optimization in Hierarchical Compare.* `ANALYZE_ABOrt`: Abort analysis information from previously captured verification information to speed up the analyze abort process. This feature is described further in a web interface document titled *Second Run Optimization for Analyze Abort.* `ANALYZE_DATapath`: Information from second run abort analysis. This feature is described further in a web interface document titled *Datapath Analysis Second Run Optimization*. `PHAse`: Information about sequential phase inversion from the Genus implementation information for phase mapping. This feature is described further in a web interface document titled *Verification of RTL Compiler Synthesis with Sequential Phase Inversion.* `MAPping`: Mapping information from previously captured verification information to speed up mapping process. `SEQ_CONST`: Sequential constant information from previously captured verification information to speed up sequential constant modeling process. `SEQ_MERGE`: Sequential merge information from previously captured verification information to speed up sequential merge modeling process.

Related Command

READ IMPLEMENTATION INFORMATION

WRITE VERIFICATION INFORMATION

SET WATCHER

```
SET Watcher
     [ON | OFF]
     [-DIRectory <directory>]
     [-RECORD <JSON>]
     [-LIMIT <resource name> <maximum value>]
     (Setup Mode / LEC Mode)
```

This command launches LEC Watcher in the background for monitoring host status.

Tcl Command

`set_watcher`

Parameters

`ON`	Launches LEC Watcher in the background
`OFF`	Terminates LEC Watcher
`DIRectory <directory>`	Specifies the working directory of LEC Watcher. The default directory is`.lec_watcher_<pid>`.
`RECORD <JSON>`	Enables LEC Watcher to record historical data of LEC and the host. LEC Watcher dumps the recorded data to the given JSON file under its working directory.
`LIMIT <resource name> <maximum value>`	Enables LEC Watcher to keep monitoring the specified resources used by LEC and terminates LEC if the resources exceeds the limits.

SET WEB INTERFACE

```
SET WEB_interface
     [ON | OFF]
     [-ANYHOST]
     [-BROWser]
     [-DIRLEVEL <integer>]
     [-DOConly | -NODOCOnly]
     [-FILEPermission | -NOFILEPermission]
     [-PORT <integer>]
```
(Setup / LEC Mode)

This command starts a web server so that you can access Conformal documentation and file directories in a web browser. Only the latest versions of Chrome and Firefox are supported.

Tcl Command

`set_web_interface`

Parameters

`ON`	Starts the web interface.
`OFF`	Stops the web interface.
`-ANYHOST`	Allows other machines to access the web interface. Without this option, only the local host can access it.
`-ANYUser`	Anyone who has this web URL can view the local file.
`-BROWser`	Displays the web interface in a stand-alone browser.
`-DIRLevel <integer>`	Specifies how many directory levels the web interface can access. Where level 0 means unlimited. Level 1 is the current directory. Default is 6.
`-DOConly`	Launches the web interface with only the Conformal documentation viewer. *This is the default.*
`-NODOCOnly`	Launches the full web interfacee viewer, which contains Conformal documentation and file directories.

`-FILEPermission`	Specifies that the web interface can only access files with world read permissions. The read permissions of the file must be enabled for user, group, and others. This is default.
`-NOFILEPermission`	Specifies that the web interface can access files without checking world readable permissions.
`-PORT <integer>`	Specifies a different port number.

Example

The following describes how to view the web interface with file directories and documentation:

1. Start the web server for web viewing:

```
SETUP> set web_interface ON
//Web Interface URL is http://host.xyz.com:8090 (http://1.2.3.4:8090)
// A modern browser supporting HTML5 is required.
// File browsing is enabled. To limit access to documentation only, use the -DOCOnly option.
```

2. The tool returns a URL of the web interface. Copy this URL into your web browser.

The following illustrates how to view the web interface with just the documentation:

```
SETUP> set web_interface ON -DOCOnly
// A modern browser supporting HTML5 is required.
// Browsing is limited to documentation only.
```

Note: The port number is unique for each LEC run on a machine. If you try to start a server for a port that is already in use, the tool returns a message similar to the following:

```
Failed to bind to port 8090: Address already in use
// Error: Web interface server cannot be created.
```

In this case, use the -PORT option and specify a new port number:

```
SETUP> set web_interface ON  -port 8091
// Web Interface URL is http://hostname:8091
```

SET WIRE RESOLUTION

```
SET WIre Resolution
     <And | Or | Wire>
     [-Both | -Golden | -Revised]
     (Setup Mode)
```

Specifies how Conformal treats the output behavior of multi-driven nets. *The system default for both the Golden and Revised designs specifies that multi-driven nets are treated as a wire-AND behavior.*

When you use the `wire` option and Conformal encounters a multi-driven net (that is, bus contention) in a design, Conformal models this multi-driven net as `TIE-X`.

Note: This `TIE-X` is a pseudo input, *not* a "don't care".

Use the `REPORT ENVIRONMENT` command to display the settings of the wire resolution for the Golden and Revised designs.

Tcl Command

```
set_wire_resolution
```

Parameters

`And`	Assigns a wire-AND behavior to multi-driven nets. *This option is the system default.*
`Or`	Assigns a wire-OR behavior to multi-driven nets.
`Wire`	Assigns a TIEX behavior to multi-driven nets.
`-Both`	Applies the specified behavior of multi-driven nets to both the Golden and Revised designs. *This is the default.*
`-Golden`	Applies the specified behavior of multi-driven nets to the Golden design.
`-Revised`	Applies the specified behavior of multi-driven nets to the Revised design.

Related Command

REPORT ENVIRONMENT

SET X CONVERSION

```
SET X COnversion
     <DC | E | 0 | 1>
     [-BOth | -GOlden | -REvised]
     (Setup Mode)
```

Specifies how Conformal handles X assignments when modeling the design. It takes effect when changing from the Setup mode to the LEC mode.

The system defaults specify that X assignments are treated as *"Don't Cares"* for the Golden and "*Error (E) Gates*" for the Revised design. If the X assignment space of the Revised design is within the X assignment space of the Golden design, then the E gate is marked as an extra unmapped point (redundant gate) after comparison.

The Revised X Handling feature enhances RTL-to-RTL comparisons. It ensures that the Revised Xs are in the Golden Don't Care space. To turn off this feature, use `set x conversion dc -revised`. This feature has been available since version 4.3.

However, use this option only if you are certain that the X assignment space of the Revised design is within the X assignment space of the Golden design; otherwise, potential errors might be masked.

Use the `REPORT ENVIRONMENT` command to display the settings of the X assignment for the Golden and Revised designs.

Tcl Command

```
set_x_conversion
```

Parameters

`DC`	Assigns "Don't Care" handling to X.
`E`	Assigns "Error Gate" handling to X. E is a pseudo input.
`0`	Assigns Zero logic handling to X: `1'b0`.
`1`	Assigns One logic handling to X: `1'b1`.
`-BOth`	Applies the specified behavior of X assignments to both the Golden and Revised designs. *This is the default.*
`-GOlden`	Applies the specified behavior of X assignments to the Golden design.

`-REvised`	Applies the specified behavior of X assignments to the Revised design.

Related Command

REPORT ENVIRONMENT

SET XC

```
SET XC
```
(*Setup Mode*)

Note: This requires a Conformal GXL license.

Analyzes switch and primitive drive strength to achieve the most accurate logic function result. This technique can be applied to complex custom macros such as RAM and ROM and is essential to accurate verification of circuits with complex layer switch nets.

SETENV

```
SETENV
    <variable> <value>
    (Setup / LEC Mode)
```

Assigns a value to an environment variable name.

Tcl Command

`setenv`

Parameters

`<variable>`	Adds the specified variable to the environment.
`<value>`	Assigns the specified value to the variable.

Examples

```
setenv LM_LICENSE_FILE 5280@host
```

where `host` is the license server host name.

Related Command

PRINTENV

SETUP NAME

```
SETup NAme
     [-GOLden | -REVised]
     [-CHANGE <instance_name> <new_instance_name>
        [-TYPE <CELL|INSTANCE|PORT>]]
     [-UNGROUP <instance_name>]
     [-MAP_MODULE_NAME <instance_path> <new_module_name>]
     [-ALIAS_MODULE_NAME <module_name> <new_module_name>]
     [-MODULE <module_name>]
     [-UNIQUIFY]
     [-BUILD [-CASESENSITIVE]]
     [-BUILD_PIN]
     [-CLEAR]
     [-SOURCE <file>]
     [-COMMIT]
     [-QUERY <hierarchical_instance_name>]
     [-PROFILE]
     [-LINT]
     [-RENAME <string> <string>]
     [-APPLY_MULTIBIT_RULE -ONTO <modules*>]
     [-DUPLICATION_PATTERN <string> [<string>]]
     [-NOKEEP]
     [-VERBOSE [-VERBOSE]]
     (Setup / LEC Mode)
```

`SETUP NAME` creates a deck to apply the naming guidance for equivalence checking. The usage starts from `-BUILD` and ends with `-COMMIT`.

`BUILD` and `BUILD_PIN` create the naming DB from HRC DB. `MAP_MODULE_NAME`, `ALIAS_MODULE_NAME`, and `UNIQUIFY` add module reference when applying the naming conversion.

`CHANGE` and `UNGROUP` would modify the naming DB. The converted name can utilize command `SETUP_REGISTER_OPTIMIZATION`, and for mapping after `-COMMIT`.

For the name of `instance_name` and `instance_path`, the command searches the reference by using case-sensitive style in default and treat `/|[].:_` as the same delimiter. For example, `a[3], a_3_` would point to the same thing..

Tcl Command

`setup_name`

Parameters

`-GOLden`	Applies settings to Golden design.
`-REVised`	Applies settings to Revised design.
`-CHANGE`	Given the pin/cell/instance name under the module specified by `-MODULE`, LEC adds the new name that can refer to the specified instance. If `-TYPE` is not given, the default type is cell which includes the register and instance.
`-TYPE <CELL|PIN}|INSTANCE>`	The type used in `-CHANGE`. `CELL` includes the register and the instance.
`-UNGROUP <instance_name>`	The cell/pin/instance names under the instance_name specified of `-UNGROUP` would can be reference by its parent module. For example, if user specify `-UNGROUP` ’ins’ `-MODULE` ’mod’, and there is pin ’pp’ under ’ins’, the ’ins/pp’ can be referenced by ’mod’.
`-MAP_MODULE_NAME <instance_path> <new_module_name>`	The `instance_path` specified under the module of `-MODULE` can be referred by `new_module_name` when the option is used.
`-ALIAS_MODULE_NAME <module_name> <new_module_name>`	The `module_name` specified can be referred by the `new_module_name` after the option issues.
`-UNIQUIFY`	After the option is issued, the module in Golden can be referred by the name of the module in Revised if they have the same name.
`-MODULE <module_name>`	Specifies the module_name to search the `instance_path` or `instance_name` in different options.
`-BUILD`	Build the naming DB for the `setup_name`.
`-CASESENSITIVE`	Setup the DB by using the case sensitive style.

`-BUILD_PIN`	In default, cell and instance name would be changed and referred, but pin name does not setup due to reducing memory size. With `-BUILD_PIN`, we can apply the change name to the pins. The `BUILD_PIN` only creates on module where they have the same instance hierarchy from top.
`-CLEAR`	Clear the DB.
`-SOURCE <Tcl_command_files>`	Quickly apply the `setup_name` command and `setup_register_optimization` in a file. The file can contain the line-by-line commands.
`-COMMIT`	Create the mapping information and sequential optimization to enable guidance setup flow in LEC mode. After `-COMMIT`, the naming DB will be clear.
`-QUERY <instance_path>`	Query if the `instance_path` exists and shows its module name.
`-PROFILE`	Dumps all historical names in the module specified from `-MODULE`.
`-VERBOSE`	Dumps the verbose information.
`-LINT`	Report the mapping quality. If `-verbose -verbose` apply, it will output the detailed information about not-mapped.
`-RENAME <string> <string>`	Apply the name changes to the instance/cell/pin names under the specified -MODULE by the renaming rule. The first string is the matched pattern and second string is the substituted pattern as the same in `add_renaming_rule`.
`-APPLY_MULTIBIT_RULE -ONTO <modules*>]`	Apply the name changes to the cell names by decomposing multibit naming. The rule is from the command set_multibit_option. To use this option, please specify the option -MODULE to assign the scope of application and -ONTO to the library module you want to apply for.
`-DUPLICATION_PATTERN <string> [<string>]`	

	Annotate duplication flops from the pattern, and LEC would apply the annotation in compare stage. There are 2 forms: 1 argument and 2 arguments. If you specify 2 arguments, LEC would matches by duplicated cells by first pattern and find the representative by the second pattern. The schema of first and second pattern is the same to the matched and substitution pattern in add_renaming_rule. For example, you can specify "abc_%d" "abc" and this options will annotate abc_1, abc_2 as the duplication flops of abc. if you specify 1 arguments form, LEC automatically creates the match and substitution pattern by "(.*)_<string>" "@1". One argument form is often to used when you want to annotate the flops with name abc_rep1 abc_rep2... you can apply "_rep%d" as the argument.
`-NOKEEP`	This is the option followed by -CHANGE. Tool would drop the original name for following applications.

Related Command

SETUP REGISTER OPTIMIZATION

READ GUIDANCE INFORMATION

SETUP REGISTER OPTIMIZATION

```
SETup REgister Optimization
     [-GOLden | -REVised]
     [-VERBOSE]
     [-SEQ_MERGE <hierarchical_instance_name> <hierarchical_instance_name>]
     [-SEQ_CONSTANT <hierarchical_instance_name>]
     [-INVERT <hierarchical_instance_name>]
     [-MODULE <module_name>
     (Setup / LEC Mode)
```

Annotate the sequential optimization information by using the name build in SETUP_NAME.

Tcl Command

```
setup_register_optimization
```

Parameters

`-GOLden`	Applies settings to Golden design.
`-REVised`	Applies settings to Revised design.
`-VERBOSE`	Dump the verbose information.
`-SEQ_MERGE <hierarchical_name> <hierarchical_name>`	Merges the second register to the first register by using the `hierarchical_name` specified. If `-MODULE` is applied together with this option, it will search the hierarchical name from the module.
`SEQ_CONSTANT <hierarchical_name>`	Guild LEC to do sequential constant optimization to the register specified by the `hierarchical_name`. If `-module` is applied, it searches the flop from the module.
`-INVERT <hierarchical_name>`	Annotate the flop has the inverted phase to the original design. If one of the mapped registers are annotated as `-INVERT`, LEC performs inverted map in mapping. If both mapped registers are annotated as `-INVERT`, LEC does positive-phase mapping.

Related Command

SETUP NAME

READ GUIDANCE INFORMATION

SUBSTITUTE BLACKBOX WRAPPER

```
SUBStitute BLackbox Wrapper
     <pattern_list>
     [-GOLden | -REVised]
     (Setup Mode)
```

Searches for each blackbox instance in your design whose module name matches those in a specified pattern list, and replaces them with new, fully-defined modules. Use this module in conjunction with the `WRITE BLACKBOX WRAPPER` command.

If you enabled the automatic blackbox substitution feature using the `WRITE BLACKBOX WRAPPER -auto_substitute` command, you do not need to use this command as the models are automatically replaced.

Tcl Command

`substitute_blackbox_wrapper`

Parameters

`<pattern_list>`	Searches for blackbox instances whose module name matches the specified pattern(s). This option accepts the * wildcard.
`-Golden`	Search and replace only in the Golden design.
`-Revised`	Search and replace only in the Revised design.

Example

The following is a set of sample commands that show this and related commands in context.

Sample module:

```
<<< des.v>>>
module design(clk, rst, cs, wr, rd_addr, wr_addr, din, dout);
input clk, rst, cs, wr;
input [2:0] rd_addr, wr_addr;
input [4:0] din;
output[4:0] dout;

DW_ram_r_w_s_dff #(5, 8, 0) ram (.clk(clk), .rst_n(rst), .cs_n(cs),
.wr_n(wr), .rd_addr(rd_addr), .wr_addr(wr_addr), .data_out(dout), .data_in(din) );
endmodule
```

1. Specify that Conformal treat undefined cells as blackboxes.

   ```
   > set undefined cell black_box
   ```

2. Read in the design, which contains our sample module.

   ```
   > read design des.v
   ```

3. Write a wrapper file `dir/_DW_ram_r_w_s_dff_5_8_0.v` for blackbox module `DW_ram_r_w_s_dff_5_8_0`.

   ```
   > write blackbox wrapper DW* -directory dir
   > break
   ```

   Note: This command also generates synthesis script template file `dir/RUN/synth.tcl`.

4. Use the `dir/RUN/synth.tcl` script with your own synthesis tool to generate `dir/RUN/_DW_ram_r_w_s_dff_5_8_0.g.v`.

5. Read the newly created `dir/RUN/_DW_ram_r_w_s_dff_5_8_0.g.v` file into the design.

   ```
   > read design -append dir/RUN/*.g.v
   ```

6. Substitute the old module of blackbox instance `ram` with the new module `_DW_ram_r_w_s_dff_5_8_0_DW_ram_r_w_s_dff_5_8_0_0(dir/RUN/_DW_ram_r_w_s_dff_5_8_0.g.v:229).`

   ```
   > substitute blackbox wrapper DW*
   ```

Related Command

WRITE BLACKBOX WRAPPER

SYSTEM

```
SYStem
     <string_command>
     (Setup / LEC Mode)
```

Enables any command your UNIX operating system recognizes.

Note: In GUI mode, the Conformal software prints the return in the transcript window.

You can substitute the exclamation mark (!) for the word `System`, as shown in the example below.

Tcl Command

```
system
```

Parameters

`<string_command>`	Any valid UNIX command.

Example

```
system ls
system pwd
!pwd
```

TCLMODE

```
TCLMODE
        (Setup / LEC Mode)
```

Switches from Conformal command entry mode (VPX mode) to Tcl command entry mode. VPX mode is the default command mode.

There are two types of Tcl mode commands:

- Native Tcl commands
- Conformal Tcl commands

Note: When issuing commands in Tcl mode, you must type them in lowercase. In Tcl mode, you must use a backslash to specify the following special characters: `{ } $ [ ] "`.

TCL

For more information about native Tcl commands, refer to the public Tcl manual, which is widely available online. Conformal Tcl commands are discussed in detail in the *Conformal Equivalence Checking User Guide*.

Tip

To start the Conformal software in Tcl mode without executing any initialization script, run the following command at a UNIX system prompt:

```
UNIX% lec -tclmode
```

Tip

In the Tcl command entry mode, you can save report data to a file using the redirection command. For example, the following command saves the gate report data to a file named `gate.out`:

```
TCL_SETUP> report_gate -type dff > gate.out
```

Related Command

VPXMODE

TEST RENAMING RULE

```
TESt REnaming Rule
     <-Design [-NOprint | -Print <Single | Pair | Group>]
       |<string>
       |-GATE_id <gate_id>>
     [-All | -NEw_rule <string> <string>]
     [-File <filename> [-REPlace]]
     [-MAp [-TYpe <PI | E | Z | DFf | DLat | CUt | BBox | PO>...
       |-NOTYpe < PI  | E | Z | DFf | DLat | CUt | BBox | PO>...]
       |-MOdule |-PIn [-BBox <module_name>]]
     [-RULE_USAGE | -NORULE_USAGE]
     [-SORTNAme]
     [-Both | -Golden | -Revised]
     [-VErbose]
     (LEC Mode)
```

Displays the results of the key point matching based on the user-specified renaming rules. Use this command to obtain a quick summary of how well the key points will be mapped based on the specified renaming rules. Test new and existing rules.

Tcl Command

```
test_renaming_rule
```

Parameters

`-Design`	Tests the renaming rules on the entire design. This argument instructs Conformal to display a summary of the Golden and Revised *pairs and groups* and Golden and Revised *single un-grouped* key points for the entire design.	
`-NOprint`	Do not display the key point pairs, un-grouped single key points, or grouped key points. *This is the default.*	
`-Print`	Displays key points as follows:	
	`Single`	Un-grouped single key points
	`Pair`	Key point pairs
	`Group`	Grouped key points
`string`	A string that the renaming rule uses as an example.	
`-GATE_id <gate_id>`	Tests renaming rules on the specified gate.	

`-All`	Tests all renaming rules within the given defaults. *This is the default.*
`-NEw_rule <string> <string>`	A new renaming rule. The first string is the pattern to be matched; the second string is the substitution pattern.
`-File <filename>`	Writes the results to the specified file.
`-REPlace`	Replaces the above file, if it exists.
`-MAp`	Tests the renaming rules on key points that will be mapped. *This is the default.*
`-TYpe`	Tests renaming rules for all key points of the specified type. *This is the default.*
`-NOTYpe`	Tests renaming rules for all key points except the specified types.

The available types are as follows:

`PI`	Primary Inputs
`E`	TIE-E gates
`Z`	TIE-Z gates
`DFF`	D flip-flops
`DLAT`	D-latches
`CUT`	Artificial gates for breaking combinational feedback loops
`BBOX`	Blackboxes
`PO`	Primary Outputs

`-MOdule`	Tests the renaming rules on the modules in the design.
`-PIn`	Tests the renaming rules on pin names of blackboxes.
`-BBox <module_name>`	Tests the pin renaming rule on the specified blackbox module. *The default is to test all blackboxes.*
`-RULE_USAGE`	Displays the number of matches for the specified string. *This option is turned on by default.*
`-NORULE_USAGE`	Do not display the number of matches for the specified renaming rule.

`-SORTNAme`	When this option is used with the `-print` option, the results appear in alphabetical order by name.
`-Both`	Tests the renaming rules on both the Golden and Revised designs. *This is the default.*
`-Golden`	Tests the renaming rules on the Golden design.
`-Revised`	Tests the renaming rules on the Revised design.
`-VErbose`	Displays both the original key point name and the renamed key point.

Understanding the Report Output

The following is sample output for the `TEST RENAMING RULE` command.

```
================================================================================
Test name mapping results
--------------------------------------------------------------------------------
Golden-Revised pair: 147
PI:                 65
PO:                 5
DFF:                32
DLAT:               1
BBOX:               1
Z:                  43
Golden-Revised group:       1 (22 key points)
Golden single:              33
Reachable:                  32
Unreachable:                1
Revised single:             2
Reachable:                  1
Unreachable:                1
================================================================================
Renaming rule usages:
================================================================================
Matches  Name       Rename pattern
--------------------------------------------------------------------------------
32  rule1      df_reg1_reg\[%d\]
0   rule2      df_reg\[%d\]
================================================================================
```

The `TEST RENAMING RULE` command does a trial of the user-specified renaming rules on all of the keypoints and displays the potential results of the `MAP KEY POINTS` command and the number of key points the renaming rules should be able to successfully rename.

For this example, there are 147 matching key points; this indicates that the `MAP KEY POINTS` command *should* be able to map 147 key points.

For the other areas (Golden-Revised group, Golden-single, and Revised-single), use additional renaming rules to try and convert them into mappable key points (Golden-Revised pair).

Test name mapping results

This section of the report displays the key points that the `MAP KEY POINTS` command should be able to map given the current renaming rules. Note: In some cases, the mapping process determines that some key points are unreachable, resulting in a mismatch between the number reported by `TEST RENAMING RULE` and `REPORT KEY POINTS`. Where:

- "Golden-Revised pair" refers to the total number of key points that name-match based on the user-specified renaming rules. The key points are then listed by type (PI, PO, DFF, DLAT, and BBOX).
- "Golden-Revised group" refers to the number of groupings that are similar based on functional mapping, but do not have exact matching names. This number is followed by the total number of key points from all groups. To view the groupings, use the following command:

  ```
  test renaming rule -design -print group -file <group.log> -replace
  ```
- "Golden-single" and "Revised-single" refers to key points that have no corresponding key points in the other design. To view the individual key points, use the following command:

  ```
  test renaming rule -design -print single -file <single.log> \   -
  replace
  ```

Renaming rule usages

This section of the report lists the renaming rules and the number of key points matched by that renaming rule. Rules are listed in the order of entry; the renaming engine applies renaming rules in the order they were entered and adds +1 to the success count when it successfully renames a key point. Therefore, if a previous renaming rule covers a latter one, the latter one will not be applied during mapping.

If a renaming rule is listed with zero matched key points, it means the renaming rule cannot find the specified search pattern (either the renaming rule was entered incorrectly or previous renaming rules changed the key point name).

Examples

- The following command displays the summary results of the renaming rules on the designs:

  ```
  test renaming rule -design -all
  ```

- The following command displays a list of all Golden/Revised un-grouped single key points:

  ```
  test renaming rule -design -print single
  ```

- The following command displays a list of all Golden/Revised grouped key points:

  ```
  test renaming rule -design -print group
  ```

- The following command displays a list of all Golden/Revised pair key points:

  ```
  test renaming rule -design -print pair
  ```

- The following command writes a list of all Golden/Revised grouped key points to a file:

  ```
  test renaming rule -design -print group -file group_list
  ```

- The following command displays the summary results of the renaming rule on the designs with a new rule:

  ```
  test renaming rule -design -new_rule <string> <string>
  ```

- The following command displays the summary results of the renaming rules applied to a sample string:

  ```
  test renaming rule <string> -all
  ```

- The following command displays the summary results of the new renaming rule applied to a sample string:

  ```
  test renaming rule <string> -new_rule <string> <string>
  ```

Related Commands

ADD RENAMING RULE

DELETE RENAMING RULE

MAP KEY POINTS

REPORT RENAMING RULE

SET NAMING RULE

UNIQUIFY

```
UNIQuify
     <module_name ...> | <-ALL [-Library | -NOLibrary][-AUTO_MATCH | -
     NOAUTO_MATCH]>
     [-Force]
     [-USE_RENaming_rules]
     [-Summary | -Verbose]
     [-Golden | -Revised]
     (Setup Mode)
```

Makes the specified module, which has multiple instances, unique. This command lets you remedy the "incompatible" instantiations warnings during hierarchical script generation. If Conformal does not make the modules unique, they are not included in the hierarchical dofile.

Using the UNIQUIFY Command

Use this command after you have matched the instance hierarchies between the Golden and Revised designs, and before any commands with pathname-based specifications.

`UNIQUIFY` renames modules only when complementing designs have matching instance names and matching instance hierarchies. If this command attempts to rename a module name, but its instance has a difference hierarchy between the Golden and Revised designs, the renaming will fail. To ensure that the hierarchy matches between the designs, it is recommended that you match the hierarchy between the Golden and Revised designs (`FLATTEN -MATCHHierarchy`) before you use `UNIQUIFY`. Note that in the ECO flow, the module and instance names should not be changed in the golden G1 netlist. Therefore, the `FLATTEN` and `UNIQUIFY` commands should only be applied to the revised G2 netlist in that particular use case.

Use `UNIQUIFY` *before* any commands with pathname-based specifications so that the effect can be propagated during hierarchical constraint extraction.

For example, the following command sequence:

```
SETUP> add primary input usub_0/net -golden
SETUP> add pin constraint 0 usub_0/net -golden
SETUP> uniquify -all -nolib
Uniquified 1 instance(s) referring to module 'sub_0' in Golden
// Warning: Added primary input(s) are deleted
```

should be:

```
SETUP> uniquify -all -nolib
Uniquified 1 instance(s) referring to module 'sub_0' in Golden
SETUP> add primary input usub_0/net -golden
SETUP> add pin constraint 0 usub_0/net -golden
```

When using hierarchical compare for abort resolution, this command allows you to include more modules in the hierarchical dofile, therefore reducing the compare complexity of helping resolve aborts.

For more information, see Hierarchical Comparison for Abort Resolution in the *Conformal Equivalence Checking User Guide*.

Tcl Command

```
uniquify
```

Parameters

`<module_name ...>`	Makes the specified module(s) unique.
`-ALL`	Makes all modules, within the given defaults, in the specified design unique.
`-Library`	Makes all modules in designs and libraries unique. *This is the default.*
`-NOLibrary`	Makes all modules in designs unique.
`-AUTO_MATCH`	Uniquify modules with automatic instance matching between designs. The automatic instance matching considers special character handling in instance pathnames. When -ALL is specified, *this is the default*.
`-NOAUTO_MATCH`	Do not consider automatic instance matching when uniquifying modules. To specify this, you must also specify -ALL.
`-Force`	Forcibly makes specified modules in the Golden or Revised design unique, even if those modules have not been made unique in the complementing design. (For example, forcibly make Golden modules unique when the Revised modules have not been made unique.)
`-USE_RENaming_rules`	Considers renaming rules for instances.
	When adding renaming rules, the software renames the Golden design instances to be same as in the Revised design, so running a subsequent `UNIQUIFY` command with this option will make the Golden modules that have matching instance names in the Revised design unique.

`-Summary`	Summarizes the outcome of making modules unique. *This is the default.*
`-Verbose`	Provides expanded information about the modules that were made unique.
`-Golden`	The specified modules are in the Golden design. *This is the default.*
`-Revised`	The specified modules are in the Revised design.

Example

The following command example creates a hierarchical dofile script named `hier.do` containing the compare script for the submodules and the root module, then runs hierarchical compare. This is can help in resolving aborts.

```
...
uniquify -all
write hier_compare dofile hier.do
run hier_compare hier do
```

Related Commands

RESOLVE

RUN HIER_COMPARE

WRITE HIER_COMPARE DOFILE

USAGE

```
USAge
     [-Benchmark]
     [ | -Elapse | -Delta]
     [-MIN_COMMAND_SECONDS <double>]
     [-NOAuto | -Auto]
     [-Remote]
     [-WATCHER [-DETAILED]
               [-HOST]
               [-PSTACK]
               [-PERIOD <period>]
               [-REPEAT <times|INF>
               [-CHECK <resource name> <minimum value required>]]
     (Setup / LEC Mode)
```

Displays the total CPU run time and current memory use since you started Conformal.

To help identify and track usage, LEC automatically calls this command for commands like `WRITE HIER_COMPARE DOFILE`, which can potentially have a long run time.

Tcl Command

```
usage
```

Parameters

`-Benchmark`	Executes a small verification benchmark and calculates the rating of the CPU based on the time used for the benchmark. **Note:** This option takes around ten seconds to complete.
`-Elapse`	Displays the elapsed time of a process. *This is the default*.
`-Delta`	Displays the difference, in seconds, between the current CPU run time and CPU run time when you last issued the `usage` command.
`-MIN_COMMAND_SECONDS <double>`	For a single command, sets the minimum CPU run time (in seconds) required to trigger the automatic print out of usage. You must set the `-auto` option to use this option. Default value is 0.1 seconds.

`-NOAuto`	Do not automatically display usage for each command.
`-Auto`	Automatically displays usage for each command. *This is the default*.
`-WATCHER`	Displays the total CPU run time and current memory use from LEC Watcher `DETAILED`: Displays the detailed reports of LEC and its subprocesses `HOST`: Displays host information `PSTACK`: Displays the current call-stack `PERIOD` <period>: Specifies the repeating period of reporting in seconds. Default value is 1 second `REPEAT` *`<times|INF>`*: Specifies how many times to report the usage. The default value is 1 `CHECK` *`<resource name> <minimum value required>`*: Checks if a specified available resource is greater than the minimum value required; otherwise it would abort LEC. The following resources can be specified: free disk Specifies free disk space available under LEC's current working directory. free-swap Specifies free swap memory available on the machine. usable-cpus Specifies the number of cpus that can be used by LEC.

VALIDATE CIRCUIT

```
VALidate CIrcuit
     [-ASM | -NOASM]
     [-BBOXSCRipt <filename>]
     [-DOfile <filename> ...]
     [-MODule <module name>]
     [-POWERPIN_TO_INput | -NO_POWERPIN_TO_INput]
     [-POWERPINs <pin_name 0> <pin_name 1> ... <pin_name N>]
     [-PRESERVE_MODEL_OPTIONs | -NO_PRESERVE_MODEL_OPTIONs]
     [-Revised | -Golden]
     (Setup Mode)
```

Note: This requires a Conformal GXL license.

Checks circuit libraries and custom blocks (when applicable), and enables equivalence checking on the full integrated circuit design. Use this command at the integrated-circuit level for RTL or Gate to final circuit. This application is for checking the consistency of pre-defined libraries during design verification.

Important

Do not use this command for validating the library itself. To validate library itself, use the `VALIDATE LIBRARY` command instead.

For both the Golden and Revised designs, refer to the same library so that any inconsistencies at the library cell level will not affect the equivalence checking on design level. After which, all the library cells under this checking will be replaced by their counterpart reference cells.

Note: You can use this to check a verified circuit. However, you need a Conformal XL license to diagnose logic abstraction and errors that relate to library comparisons.

Tcl Command

```
validate_circuit
```

Parameters

`-ASM`	Enables the Advanced State-element Modeling (ASM) algorithm. This helps to analyze loop structure to produce better modeling of state elements, such as D-Latch, DFF, and bus-keeping I/O logic. *This is the default.*

`-NOASM` Disables the Advanced State-element Modeling (ASM) algorithm.

Tip

If there are any unexpected results, you can use this option to revert back to the functionality of the 6.2 release and earlier.

`-BBOXSCRipt <filename>`

Creates a dofile with the specified name that blackboxes all validated cells for structural verification, which is more accurate than logical verification.

`-DOfile <filename> ...`

Specifies the name of the dofile that was used to verify all custom blocks so that they can be re-checked.

If a custom module has not been verified yet, blackbox it before running the `VALIDATE CIRCUIT` command. `VALIDATE CIRCUIT` can check for consistency in custom blocks, but it cannot *verify* custom blocks. Use Conformal GXL to verify custom blocks.

`-MODule <module name>`

Validates the specified module.

By default, the software validates the root module. Use this option to validate a module other than the root module.

Note: The software validates only the topmost library and custom cells (that exist in the reference side) that are used by the specified module.

`-POWERPIN_TO_INput`

Specifies that if there are power pins that are input/output pins, they will be changed to input pins before validation. *This is the default.*

`-NO_POWERPIN_TO_INput`

Specifies that power pins that are input/output pins will NOT be changed to input pins before validation.

`-POWERPINs <pin_name 0> <pin_name 1> ... <pin_name N>`

Defines names for the pin(s) that are used as extraneous power pins, which are ignored during cell verification. For multiple power pins, each pin name must be separated by a space.

You should use the `ADD PIN CONSTRAINTS` command to tie power pins to 1 and ground pins to 0.

`-PRESERVE_MODEL_OPTIONs`

Preserves any user-defined settings made prior to this command. *This is the default.*

This option disables any flattening options that `VALIDATE CIRCUIT` sets by default.

By default, the following flatten model options are set when `VALIDATE CIRCUIT` compares two modules:

`-all_seq_merge`

`-seq_constant`

See the `SET FLATTEN MODEL` command for an explanation of these options.

With `-preserve_model_options`, the comparison is done without changing any flatten model options. Any options that were set prior to running validate circuit are used instead.

`-NO_PRESERVE_MODEL_OPTIONs`

Do not disable any flattening options that `VALIDATE CIRCUIT` sets by default.

`-Revised`	Validates the Revised database. *This is the default.*
`-Golden`	Validates the Golden database.

Related Commands

READ DESIGN

READ LIBRARY

SET FLATTEN MODEL

VALIDATE LIBRARY

```
VALidate LIbrary
     [-ANALYZE_SEtup]
     [-ASM | -NOASM]
     [-NOGOLPINDir | -GOLPINDir]
     [-NOSPIce | -SPIce]
     [-POWERPIN_TO_INput | -NO_POWERPIN_TO_INput]
     [-POWERPINs <pin_name 0> <pin_name 1> ... <pin_name N>]
     [-PRESERVE_MODEL_OPTIONs | -NO_PRESERVE_MODEL_OPTIONs]
     [-SKIP_EXTRA_CELL]
     [-Revised | -Golden]
     (Setup Mode)
```

Note: This requires a Conformal XL and Conformal GXL license, where noted.

Compares all top-level cells with matching names. Conformal can auto-abstract the modules on the SPICE side before comparison using the `-spice` option. This application is for library verification during library design. This command supports any combination of Verilog, Liberty, and SPICE.

Important

To abstract SPICE modules, you must have a Conformal GXL license.

Tcl Command

```
validate_library
```

Parameters

`-ANALYZE_SEtup`	Performs setup analysis for library cell verification. This requires a Conformal GXL license.
`-ASM`	Enables the Advanced State-element Modeling (ASM) algorithm. This helps to analyze loop structure to produce better modeling of state elements, such as D-Latch, DFF, and bus-keeping I/O logic. *This is the default.*

`-NOASM`	Disables the Advanced State-element Modeling (ASM) algorithm. *Tip* If there are any unexpected results, you can use this option to revert back to the functionality of the 6.2 release and earlier.
`-NOGOLPINDir`	Do not copy the pin directions from the Golden design to the Revised design. *This is the default.*
`-GOLPINDir`	Copies the pin directions from the Golden design to the Revised design. This is for all pins within the library cells being validated.
`-NOSPIce`	Abstracts and validates the modules that are not SPICE. *This is the default.* **Note:** This option requires a Conformal GXL license.
`-SPIce`	Abstracts and validates the SPICE modules. **Note:** This option requires a Conformal GXL license.
`-POWERPIN_TO_INput`	Specifies that if there are power pins that are input/output pins, they will be changed to input pins before validation. *This is the default.*
`-NO_POWERPIN_TO_INput`	Specifies that power pins that are input/output pins will NOT be changed to input pins before validation.
`-POWERPINs <pin_name 0> <pin_name 1> ... <pin_name N>`	Defines names for the pin(s) that are used as extraneous power pins, which are ignored during cell verification. For multiple power pins, each pin name must be separated by a space. Use the `ADD PIN CONSTRAINTS` command to tie power pins to 1 and ground pins to 0.
`-PRESERVE_MODEL_OPTIONs`	Preserves any user-defined settings made prior to this command. *This is the default.* This option disables any flattening options that `VALIDATE LIBRARY` sets by default.

`-NO_PRESERVE_MODEL_OPTIONs`	
	Do not disable any flattening options that `VALIDATE LIBRARY` sets by default.
`-SKIP_EXTRA_CELL`	Skips reporting the cells that only exist in the Golden or Revised design.
`-Revised`	Validates the Revised database. *This is the default.*
`-Golden`	Validates the Golden database.

Related Command

VALIDATE CIRCUIT

VERSION

```
VERsion
```
(*Setup / LEC Mode*)

Displays the current version release number of Conformal. You can use this command after the `SET LOG FILE` command so the version becomes a part of the transcript log. In this way, you record the Conformal version that created your results. This command is also helpful when you use the `SAVE SESSION` and `RESTORE SESSION` commands, because you must use the same Conformal version when you restore a session.

Tcl Command

```
version
```

Related Commands

RESTORE SESSION

SAVE SESSION

SET LOG FILE

VPXMODE

`VPXMode`
    (*Setup / LEC Mode*)

Switches from Tcl mode to native Conformal command entry mode (VPX mode). VPX is the default command mode.

Important

When issuing this command in the Tcl command interpreter, you must type this in lowercase. For example:

```
TCL_LEC> vpxmode
```

Tip

In VPX mode, you can save report data to a file using the redirection command. For example, the following command saves the gate report data to a file named `gate.out`:

```
SETUP> report gate -type dff > gate.out
```

Tcl Command

`vpxmode`

Related Command

TCLMODE

WRITE BLACKBOX WRAPPER

```
WRIte BLackbox Wrapper
     <pattern_list>
     [-DIRectory <dirname> ]
     [-NOAUTO_substitute | -AUTO_substitute [-LATENCY <integer>]]
     [-SYNTHESIS_library <STANDARD> | <library_list>]
     [-SYNTHESIS_Resource <filename>]
     [-TARGET_library <library_list>]
     [-Golden | -Revised]
     (Setup Mode)
```

Use this command to replace blackbox models with synthesis models.

This command generates a file that contains a wrapper for each blackbox instance in your design whose module name matches those in the specified pattern list and generates the scripts necessary for performing synthesis and generating the synthesized models.

You can also enable automatic *blackbox substitution* using the `-auto_substitute` option. This option executes the synthesis executable (specified by `SET HDL OPTIONS -synthesis_executable`) and automatically substitutes the blackbox models with the synthesized models.

Tip: Instead of using the `ADD NOTRANSLATE MODULES` command to treat particular cells as blackboxes, you can use the `set undefined cell -black_box` command prior to reading in the design and library (through the `READ DESIGN` and `READ LIBRARY` commands) to treat all undefined cells as blackboxes.

Tcl Command

```
write_blackbox_wrapper
```

Parameters

`<pattern_list>`	Writes out module wrappers for blackbox instances whose module name matches the specified pattern(s). This option accepts the `*` wildcard.

`-DIRectory <dirname>`	Specifies the working directory for this command. When this command is used, the tool creates a `RUN` directory under this working directory that will contain all the files generated by this command. If you do not specify this option, the tool creates a working directory called `CFM_BBOX_DIR` under your current working directory. All the files will be stored under this directory, in a subdirectory called `RUN`. If you specify a project directory, by default, the working directory `CFM_BBOX_DIR` is created underneath the project directory instead. Each time this command is used, the tool creates a new RUN directory under the working directory (appended by `*.<integer>`, where integer is the number of times the command has been issued). For example, `RUN.3`.
`-NOAUTO_substitute`	Disables automatic blackbox substitution. *This is the default*. See sample flows below.
`-AUTO_substitute`	Enables automatic blackbox substitution. With this option, the tool automatically performs synthesis using the corresponding synthesis script and substitutes the blackbox models with the synthesis models from the synthesis run. **Note:** Before using this command, you must: specify the synthesis executable using the `SET HDL OPTION -synthesis_executable` command, read in the synthesis library for the elaborated cells using the `READ LIBRARY -append` command, and set the path to the synthesis licenses.
`-SYNTHESIS_library <STANDARD | <library_list>>`	
	Specifies the synthesis libraries to use for synthesis. You can specify a list of space separated libraries, or you can use the STANDARD option to include `gtech.db` and `dw_foundation.sldb`. If you do not specify this option, the tool uses the default set of libraries:

	`gtech.db dw01.sldb dw02.sldb dw03.sldb dw04.sldb dw05.sldb dw06.sldb dw07.sldb dw08.sldb standard.sldb dw_foundation.sldb` These libraries are used in automatic *blackbox substitution* (enabled by the `-auto_substitute` option)
`-SYNTHESIS_Resource <filename>`	
	Specifies the resource file where the DW minPower component information is used. Single filename only.
`-TARGET_library <library_list>`	
	Specifies the user's target library.
`-LATENCY <integer>`	Specifies the maximum number of seconds to wait before reading the wrapper netlist(s) and performing the blackbox substitution. This option is useful in cases where the wrapper netlist(s) might not be immediately available due to network delays.
`-Golden`	Write out blackbox wrappers for the Golden design. *This is the default.*
`-Revised`	Write out blackbox wrappers for the Revised design.

Examples

Sample module:

```
<<< gol.v>>>
module design(clk, rst, cs, wr, rd_addr, wr_addr, din, dout);
input clk, rst, cs, wr;
input [2:0] rd_addr, wr_addr;
input [4:0] din;
output[4:0] dout;

DW_ram_r_w_s_dff #(5, 8, 0) ram (.clk(clk), .rst_n(rst), .cs_n(cs),
.wr_n(wr), .rd_addr(rd_addr), .wr_addr(wr_addr), .data_out(dout), .data_in(din) );
endmodule
```

Manual Blackbox Substitution

The following illustrates the default flow for `WRITE BLACKBOX WRAPPER` (with automatic blackbox substitution disabled):

1. Specify that Conformal treat undefined cells as blackboxes.

   ```
   > set undefined cell black_box
   ```

2. Read in the Golden design, which contains our sample module.

   ```
   > read design gol.v
   ```

3. Write a wrapper file `dir/_DW_ram_r_w_s_dff_5_8_0.v` for blackbox module `DW_ram_r_w_s_dff_5_8_0`.

   ```
   > write blackbox wrapper DW* -directory dir
   > break
   ```

   Note: This command also generates synthesis script template file `dir/RUN/synth.tcl`.

4. Use the `dir/RUN/synth.tcl` script with your own synthesis tool to generate `dir/RUN/_DW_ram_r_w_s_dff_5_8_0.g.v`.

5. Read the newly created `dir/RUN/_DW_ram_r_w_s_dff_5_8_0.g.v` file into the design.

   ```
   > read design -append dir/RUN/*.g.v
   ```

6. Substitute the old module of blackbox instance `ram` with the new module `_DW_ram_r_w_s_dff_5_8_0_DW_ram_r_w_s_dff_5_8_0_0(dir/RUN/_DW_ram_r_w_s_dff_5_8_0.g.v:229).`

   ```
   > substitute blackbox wrapper DW*
   ```

Automatic Blackbox Substitution

The following illustrates the flow for `WRITE BLACKBOX WRAPPER` with automatic blackbox substitution enabled:

1. Specify that Conformal treat undefined cells as blackboxes.

   ```
   > set undefined cell black_box
   ```

2. Specify the Design Compiler executable:

   ```
   set hdl option -synthesis_executable /grid/DC_INSTALL_DIRECTORY/latest/bin/dc_shell-t
   ```

3. Read in the Golden design, which contains our sample module.

   ```
   > read design gol.v
   ```

4. Enable automatic blackbox substitution for module `DW_ram_r_w_s_dff_5_8_0` and use only the `gtech.db` and `dw_foundation.sldb` libraries when running synthesis to generate the synthesis model:

```
    write blackbox wrapper DW* -synthesis_library standard -auto_substitute
```

Related Command

SET HDL OPTIONS

SUBSTITUTE BLACKBOX WRAPPER

WRITE COMPARED POINTS

```
WRIte COmpared Points
     <filename>
     [-CLass <All | Eq | INVequivalent | NONeq | ABort | NOTcompared>]
     [-Replace]
     [-SHOW_ORIG_RTL_NAMES]
     [-TCLmode]
     [-TYpe <All | PO | DFf | DLat | Bbox | Cut>]
     (LEC Mode)
```

Writes compared points information to a file, which you can use to add a specific class and type of compared points to a compare list. This file can then be read in directly using the `DOFILE` command.

Tcl Command

`write_compared_points`

Parameters

`filename`	Specifies the filename.	
`-CLass`	Writes out the class of compared points. By default, the command writes out all classes.	
	`All`	Writes all compared point classes. *This is the default* if you do not specify the `-CLass` option.
	`Eq`	Writes compared points that are equivalent.
	`INVequivalent`	Writes the compared points that are inverted equivalent.
	`NONeq`	Writes the compared points that are nonequivalent.
	`ABort`	Writes the aborted compared points.
	`NOTcompared`	Writes the compared points that are not compared.
`-Replace`	Replaces the existing file.	

`-SHOW_ORIG_RTL_NAMES`	Writes out the DFF/DLAT compared point names without the applied `SET NAMING RULE - REGISTER <STRING>` setting.	
`-TCLmode`	Use only Tcl commands in the file.	
`-TYpe`	Writes out the type of compared points. By default, the command writes out all types.	
	`All`	Writes all compared point types. *This is the default* if you do not specify the `-TYpe` option.
	`PO`	Writes the compared points of the primary outputs.
	`DFf`	Writes the compared points of the D flip-flops.
	`DLat`	Writes the compared points of the D-latches.
	`Bbox`	Writes the compared points of the blackboxes.
	`Cut`	Writes the compared points for artificial gates that break combinational loops.

Related Commands

DOFILE

WRITE DESIGN

```
WRIte DEsign
     <filename>
     [ | -ALL | -Module <module_name> | -BBOX_only]
     [-ENABLE_BLT_IN_DW_WRITE | -DISABLE_BLT_IN_DW_WRITE]
     [-GZip]
     [-Library]
     [-REPlace]
     [-RTL]
     [-SKIP_Undefined]
     [-STRC_VIEW]
     [-TEST_VIEW]
     [-Used]
     [-Golden | -REVised]
     (Setup / LEC Mode)
```

Writes out the Golden or Revised design in Verilog format to examine how Conformal abstracts RTL descriptions into gate-level descriptions.

Tilde: Use the tilde "~" character in the file path to replace the path to the user login home directory.

For information on how this command handles files with encrypted/protected modules, refer to the "IP Protection" section of the READ DESIGN command.

Tcl Command

```
write_design
```

Parameters

`<filename>`	Writes the design to this file.
`-ALL`	Writes out all modules that are stored in the design space.
`-Module <module_name>`	Writes out the specified module that is stored in the design space.

`-BBOX_only`	Writes out only the empty module descriptions of blackboxes that are generated through the "`set_undefined_cell bbox`", "`add_notranslate_module`", or "`add_black_box`" commands, or modules with parsing/elaboration errors. By default, the command writes out the design tree of the root module, excluding modules in the library space.
`-ENABLE_BLT_IN_DW_WRITE`	
	Writes out the CFM built-in DW module declaration. *This is the default*.
`-DISABLE_BLT_IN_DW_WRITE`	
	Do not write out the CFM built-in DW module declaration. Instead, the corresponding instance statement with the original design format will be written out.
`-GZip`	Writes the design in a compressed gzip format.
`-Library`	Writes out the library information.
`-REPlace`	Replaces the existing file.
`-RTL`	Outputs word-level operator expressions such as `*`, `+`, and `-`. By default, the command writes out the design without using the word-level operators. Without this option, the command writes out the design in Verilog primitive gates.
`-SKIP_Undefined`	Do not write undefined modules to the output.
`-STRC_VIEW`	Writes out the gate-level netlist where the netlist, MUX, D-latch, D-Flip-Flop consist of logic gates, instead of the UDP table.
`-TEST_VIEW`	Writes out the abstraction result of `ABSTRACT LOGIC -test_view`. The result might include Encounter Test primitives. To read the netlist back, use the `READ DESIGN` command's `-define` option for `ET_EC_MODEL`. For example: `read design <netlist> -define ET_EC_MODEL`
`-Used`	Writes out the specified modules and all the referenced modules, including modules in the design space.
`-Golden`	Writes out the Golden design only. *This is the default.*
`-REVised`	Writes out the Revised design only.

Related Command

READ DESIGN

WRITE EXTENDED MAPPING

```
WRIte EXtended Mapping
     <filename>
     [-REPlace]
     [-USE_PINname]
     [-GZIP]
```

This command writes the extended mapping information to a file. This file can be used to accelerate setup and mapping in Conformal ECO or LEC second run.

Use the `READ EXTENDED MAPPING` command to read the file.

Tcl Command

`write_extended_mapping`

Parameters

`<filename>`	Specifies the file name.
`-REPlace`	Replaces the existing file.
`-USE_PINname`	Appends the library pin name supported by a sequential element to its instance name. If your design is RTL, pin name will not be appended. This option allows LEC to use the pin name to distinguish different registers in the same multi-bit cell. Use this option when multi-bit cells are in your design.
`-GZIP`	Writes out the file in GZIP format. Use this option to reduce the file size.

Related Commands

ANALYZE EXTENDED MAPPING

READ EXTENDED MAPPING

WRITE HIER_COMPARE DOFILE

```
WRIte HIer_compare Dofile
     <filename>
     [-All]
     [-APPEND_String <string>]
     [-BALANCED_EXTractions]
     [-Black_box | -NOBlack_box]
     [-COMPARE_String <string>]
     [-CONDitional]
     [-ECO_aware]
     [-ECOPIN_dofile <filename>]
     [-EFFort <Medium | High>]
     [-Exact_pin_match | -NOExact_pin_match]
     [-EXTRACT_ICG]
     [-FORMat < Auto | VPX | TCL > ]
     [-KEEP_TOP_level_constraints | -NOKEEP_TOP_level_constraints]
     [-IGNORE_MISMATCH_ports]
     [-LEVEL <integer>]
     [-MODULE <golden_module> <revised_module> [-HIERarchical | -FLATten]]
     [-NEQCHECK]
     [-NOConstraint | -Constraint
          [-INPUT_OUTPUT_Pin_equivalence]]
     [-NOFUNCTION_Pin_mapping | -FUNCTION_Pin_mapping]
     [-NOPIN_BINDING | -PIN_BINDING]
     [-PREPEND_String <string>]
     [-PRINT <CPU | MEMORY | ELAPSED>]
     [-Replace]
     [-RETIMED_modules]
     [-SUPPRESS <string>]
     [-Threshold <integer>]
     [-Usage]
     [-VERBOSE]
     (Setup Mode)
```

Writes out a hierarchical dofile script that verifies the two hierarchical designs starting from the lower-level modules and progressing to the top root module. All modules not output to the hierarchical dofile script will be flattened into their parent modules for comparison.

Use options to specify one of the following actions:

- Blackbox modules after comparison
- Write modules with different numbers of pins to the dofile script
- Propagate the constraints to lower-level modules and apply them to the dofile script
- Change the minimum number of module primitives considered for hierarchical comparison

Use the tilde character (~) to shorten the path of the file.

This command also generates a dofile script to compare two libraries, such as a Liberty and Verilog library. Use the `-all` option to write all library models to the dofile script for comparison.

Notes:

- The modeling options specified using the `SET FLATTEN MODEL` command can affect the flattened netlists used for constraint extraction with the `WRITE HIER_COMPARE DOFILE` command. For example, enabling `-SEQ_CONSTANT` can propagate constant data through latches and registers, thereby extracting better quality of constraints in the generated hierarchical dofile.
- Hierarchical comparison is also useful in resolving aborts. See the `UNIQUIFY` command for more information.
- Running this command can result in a long run time. To help track the CPU run time and current memory usage, LEC automatically calls the `USAGE` command when `WRITE HIER_COMPARE DOFILE` is used.

Tcl Command

```
write_hier_compare_dofile
```

Parameters

`<filename>`	Specifies the name of the dofile script that verifies design hierarchy.
`-All`	Writes all library modules, within the given defaults, to the hierarchical dofile script. Use this option for library verification.
`-APPEND_String <string>`	Appends any string of commands to the hierarchical dofile script *after* key point comparison for each module. Use the semi-colon character (`;`) to separate commands. Enclose the string of append commands with double quotes (see "Examples" section below).
`-BALANCED_EXTractions`	Extracts a balanced set of hierarchical constraints by simultaneously using flattened Golden and flattened Revised designs.

`-Black_box`	Blackboxes each module after comparison. *This is the default.*
`-NOBlack_box`	Do not blackbox each module after comparison.
`-COMPARE_String <string>`	Replaces the default compare command with a string of compare commands in the hierarchical dofile script generation for each module. Use the semi-colon character (`;`) to separate commands. Use double quotes to surround each compare command (see "Examples" section below).
`-CONDitional`	Skips blackboxing for nonequivalent submodules during the hierarchical comparison. (The end result is that Conformal flattens these submodules.) To report the flattened modules, use the `report hier_compare result -flattened` command.
`-ECO_aware`	This option requires an ECO license. Recognizes ECO-related changes. This option recognizes ports that would otherwise be ignored for non-ECO comparisons, thus facilitating the correct comparison between the Golden and Revised design.
`-ECOPIN_dofile <filename>`	This option requires an ECO license. Writes out a dofile for adding ECO pins to the Golden design as compared to the Revised design.
`-EFFort <Medium | High>`	Specifies the effort level for generating the hierarchical comparison dofile. `Medium`: Default effort level. `High`: Provides better analysis of constraint extraction, but can increase the analysis run time.
`-Exact_pin_match`	Writes only those modules with matching pin names to the hierarchical dofile script. *This is the default.*
`-NOExact_pin_match`	Do not take pin name matching into consideration when writing modules to the hierarchical dofile script.
`-EXTRACT_ICG`	Considers the hierarchical information of ICG.

`-FLATten`	Flattens all of the submodules of the specified module.	
`-FORmat`	Specifies the format (VPX or Tcl) of the commands in the generated hierarchical dofile.	
	`Auto`	Determines the format automatically, based on the mode in which the `WRITE HIERARCHICAL DOFILE` command was specified. *This is the default*.
	`VPX`	VPX format. The hierarchical dofile will contain VPX commands.
	`TCL`	Tcl format. The hierarchical dofile will contain Tcl commands.
`-Hierarchical`	Includes all of the submodules of the specified module in the hierarchical dofile script. *This is the default.*	
`-IGNORE_MISMATCH_ports`	Forces all modules with mismatched ports to be written out to a hierarchical dofile. (To include the modules' associated messages, use the `-verbose` option.) Running the hierarchical dofile that this option writes out may cause extra non-equivalent modules to be reported (with non-dynamic hierarchical verification) and/or extra run time. Note: This is a global option that affects every module. For Conformal ECO, you can use `ADD MODULE ATTRIBUTE -eco_module` to target just the affected ECO modules.	
`-KEEP_TOP_level_constraints`		

Allows application of top-level pathname-based constraints to the appropriate submodules, during hierarchical comparison. The top-level constraints are supported for the `ADD PRIMARY INPUT`, `ADD PIN CONSTRAINTS`, and `ADD INSTANCE CONSTRAINT`S commands. This is the default.

During hierarchical comparison, if top-level constraints take effect for the specific submodule, they are enclosed within the following messages:

Applying top-level pathname-based constraints ...

End of top-level pathname based constraints.

If the top-level constraints do not take effect when they should have, the following warning message is generated:

```
//Warning: Top-level constraints might
not have been fully applied.
```

Example provided in the Examples section.

`-NOKEEP_TOP_level_constraints`

Disables the application of top-level pathname-based constraints to the appropriate submodules, during hierarchical comparison.

`-INPUT_OUTPUT_Pin_equivalence`

Extracts input-output pin equivalences within a module and applies them to hierarchical dofile script. This can be used when the design has feedthroughs or feedback buffers.

`-LEVEL <integer>`	Writes all modules to the hierarchical dofile script up to the specified hierarchical level.

`-MODULE <golden_module> <revised_module>`

Writes the specified Golden module and Revised module to the hierarchical dofile script.

Renaming rules must still be applied if the module names are different.

This option writes out the specified modules, even if the ports do not match.

`-NEQCHECK`	Skips datapath analysis when an NEQ is identified before running `ANALYZE DATAPATH`.
`-NOConstraint`	Do not apply the root module constraints and equivalences to the hierarchical dofile script. *This is the default.*
`-Constraint`	Propagates root module constraints and equivalences and applies them to the hierarchical dofile script.
`-NOFUNCTION_Pin_mapping`	Do not extract the functional mapping for the submodule boundary ports with the correct phase. *This is the default*.
`-FUNCTION_Pin_mapping`	Extracts functional mapping for the submodule boundary ports with correct phase. This option is useful when the netlist has gone through clock-tree-synthesis (CTS) creating additional submodule ports with different polarity, or if there is an inverter push across submodule boundaries without modifying the port.
`-NOPIN_BINDING`	Use `add_renaming_rules` to map pins in the dofile. *This is the default*
`-PIN_BINDING`	Use `add_pin_binding` to map pins in the dofile.
`-PREPEND_String <string>`	Appends any string of commands to the hierarchical dofile script *before* key point comparison for each module. Use a semicolon (;) to separate commands. Enclose the string of prepend commands with double quotes (see "Examples" section below).
`-PRINT`	Prints out additional information about each module comparison. `CPU`: displays the CPU run time `MEMORY`: displays the memory usage `ELAPSED`: displays the elapsed time

	For example, the following command will have the CPU run time listed in its output:
	`// Command: write hier dofile file.do -print cpu`
	`Processed 1 out of 169 module pairs EQ: 1 NEQ: 0  ABORT: 0  CPU:  80 sec`

`-Replace`	Replaces the existing file.
`-RETIMED_modules`	Writes out only those modules which have the `PIPELINE_RETIME` attribute attached to them. Use the `ADD MODULE ATTRIBUTE` command to attach the `PIPELINE_RETIME` attribute to a module.
`-SUPPRESS <`*`string`*`>`	Suppresses (does not print) warning messages that match the specified string/pattern. This option accepts wildcards (*). For example, for "-suppress *LIB*", the tool does not display any messages that contain LIB.
`-Threshold <integer>`	This threshold is the minimum number of primitives within a module that will be written to the hierarchical dofile script. *The minimum default number is 50 primitives.*
`-Usage`	Executes the `USAGE` command after each comparison and at the end of the hierarchical comparison.
`-VERBOSE`	Provides additional information when writing out the hierarchical dofile script.

Examples

```
write hier_compare dofile hier.do -replace
```

```
write hier_compare dofile hier.do -append_string "usage" -prepend_string "report
unmapped points -notmapped" -replace
```

- The following illustrates a static hierarchical comparison; it reads in the two hierarchical designs, writes out the hierarchical dofile script, and compares design hierarchies:

```
read library golden.lib -verilog -golden
read design golden.v -verilog -golden
read library revised.lib -verilog -revised
read design revised.v -verilog -revised
write hier_compare dofile hier.do -replace
dofile hier.do
```

- The following illustrates a dynamic hierarchical comparison; it creates a hierarchical dofile script named `hier.do` containing the compare script for the submodules and the root module, then runs hierarchical compare.

```
...
uniquify -all
write hier_compare dofile hier.do
run hier_compare hier do
```

- The following reads in a synthesis library and simulation library, writes out all of the library models, and compares library hierarchies:

```
read design syn.lib -liberty -golden
read design simulation.v -verilog -revised
write hier_compare dofile lib_ver.do -replace -all
dofile lib_ver.do
```

- In the following command, the default compare command is replaced with two commands during each module comparison, `set compare effort low` and `compare -abort_stop 1 -noneq_stop 1`:

```
write hier_compare dofile hier.do -compare_string \
"set compare effort low; compare -abort_stop 1 -noneq_stop 1"
```

- The following example illustrates the `-KEEP_TOP_level_constraints` option. For example, your dofile contains:

```
add primary input a0/b0/scan_en -net -Golden
add pin constraint 0 a0/b0/scan_en -Golden
write hier_compare dofile hier.do -replace -constraints \
  -keep_top_level_constraints -noexact_pin_match
dofile hier.do
...
```

- During hierarchical comparison, the logfile will contain the following messages:

```
// Running Module modB and modB
// Command: set root module modB -Golden
// Command: set root module modB -Revised
// Command: set module property -instance /a0/b0 -Golden
// Command: set module property -instance /a0/b0 -Revised
// Command: report black box -NOHidden
// Command: set system mode lec Applying top-level pathname-based constraints
// Command: add primary input scan_en -Golden
// Command: add pin constraints 0 scan_en -Golden End of top-level pathname-based constraints
// Processing Golden ...
// Modeling Golden ...
...
```

Related Commands

ADD NOBLACK BOX

ANALYZE HIER_COMPARE

DELETE NOBLACK BOX

READ DESIGN

READ LIBRARY

REPORT HIER_COMPARE RESULT

REPORT NOBLACK BOX

RESET HIER_COMPARE RESULT

RUN HIER_COMPARE

SAVE HIER_COMPARE RESULT

SET NAMING RULE

UNIQUIFY

USAGE

WRITE LIBRARY

```
WRIte LIbrary
     <filename>
     [-GZip]
     [-Module <module_name>]
     [-Verilog]
     [-REPlace]
     [-Golden | -REVised]
     (Setup / LEC Mode)
```

Writes out the Golden or Revised library to a Verilog file. *The default is to write out the library as functional Verilog model descriptions.*

Use this command to examine how Conformal abstracts complex UDP library models.

Use the tilde character (~) to shorten the path of the file.

Tcl Command

`write_library`

Parameters

`<filename>`	Specifies the library filename.
`-GZip`	Writes the library in a compressed gzip format.
`-Module <module_name>`	Writes out the specified module that is stored in the library space.
`-Verilog`	Writes out the library in Verilog format. *This is the default.*
`-REPlace`	Replaces the existing file.
`-Golden`	Writes out only the Golden library. *This is the default.*
`-REVised`	Writes out only the Revised library.

Related Command

READ LIBRARY

WRITE MAPPED POINTS

```
WRIte MApped Points
     <filename>
     [-ALIAS]
     [-CLass < Full | System | User>]
     [-INPUT_OUTPUT_PIN_MAPPING]
     [-NOTYpe <PI | E | Z | DFf | DLat | CUt | BBox | PO>...]
     [-REPlace]
     [-TYpe <PI | E | Z | DFf | DLat | CUt | BBox | PO>...]
     [-SHOW_ORIG_RTL_NAMES]
     (LEC Mode)
```

Writes the mapped point information to a file. If the comparison needs to be done at a later time, you can use this command to accelerate the mapping process.

Use the `READ MAPPED POINTS` command to read the file.

Use the tilde character (~) to shorten the file's path.

Tcl Command

```
write_mapped_points
```

Parameters

`<filename>`	Specifies the filename.	
`-ALIAS`	Include the name alias data, which changes Golden keypoint names to the corresponding Revised keypoint names.	
`-CLass`	Writes out the *System*, *User*, or *Full* classes of mapped points.	
	`Full`	Both the *User* and *System* `class.` *This is the default.*
	`System`	Key points that are mapped automatically.
	`User`	*User* class: key points that are manually mapped with the `ADD MAPPED POINTS` command.
`-INPUT_OUTPUT_PIN_MAPPING`	Writes out the input and output pin mappings of blackboxes.	

`-REPlace`	Replaces the existing file.
`-SHOW_ORIG_RTL_NAMES`	Writes out the DFF/Dlat mapped point names without the applied "`SET NAMING RULE -REGISTER <STRING>`" setting.
`-TYpe`	Writes only the mapped key points of the specified type.
`-NOTYpe`	Do not write the mapped key points of the specified type.

Available types are as follows:

`PI`: Primary input

`E`: TIE-E

`Z`: TIE-Z

`DFf`: D flip-flop

`DLat`: D-latch

`CUt`: All unmapped points for artificial gates that break combinational loops

`BBox`: Blackbox

`PO`: Primary output

Related Commands

ADD MAPPED POINTS

ADD NAME ALIAS

READ MAPPED POINTS

WRITE PARTITION DOFILE

WRITE MEMORY PRIMITIVE

```
WRIte MEmory Primitive
     <filename>
     [-COMmondef <filename2>]
     [-Module <module_name*> | -All]
     [-OPTionsdef <filename1>]
     [-REPlace]
     (LEC Mode)
```

Note: This requires a Conformal GXL license.

Writes memory primitives to a file. Use this command to retrieve information for simulation.

For additional information about memory primitives, refer to the "Memory Primitive Data Sheet" located at `<install_dir>/doc/MEM_datasheet.pdf`.

Use the tilde character (~) to shorten the path of the file.

Note: The wildcard (*) represents any zero or more characters in module names.

Tcl Command

```
write_memory_primitive
```

Parameters

`<filename>`	Specifies the filename.
`-COMmondef <filename2>`	Writes the common module called `Vpx_wireOrlatOrff_data` to a separate file. This module is common to all memory primitives. You can write common memory primitive definitions to a separate file during each call to WRITE MEMORY PRIMITIVE, or you can use an existing file using the `-rep` option.
`-Module <module_name*>`	Writes only the memory primitives for the specified module. This accepts wildcards.
`-All`	Writes out memory primitives for all modules.

`-OPTionsdef <filename1>`	
	Writes parameter option definitions to a separate file. This will write the `` `define `` statements, which are options to available parameters, from the memory primitives to a separate file.
`-REPlace`	Replaces the existing file.

Related Command

READ DESIGN

WRITE PARTITION DOFILE

```
WRIte PArtition Dofile
     <filename>
     [-APPEND_String <string>]
     [-COMPARE_String <string>]
     [-Map <filename>]
     [-PREPEND_String <string>]
     [-Replace]
     [-Usage]
     (Setup / LEC Mode)
```

Writes out a partition dofile script based on the key point names specified with the `ADD PARTITION KEY_POINT` command. The number of compare iterations is based on whether the key point names have all-pattern, one-hot, or one-cold constraints.

Use the tilde character (~) to shorten the path of the file.

Tcl Command

`write_partition_dofile`

Parameters

`<filename>`	The partition dofile is written to this file.
`-APPEND_String <string>`	Appends any string of commands to the partition dofile script *after* key point comparison for each module. Use a semicolon (;) to separate commands. Enclose the string of append commands with double quotes (see "Examples" section below).
`-COMPARE_String <string>`	Replaces the default compare command with a string of compare commands in the partition dofile script generation for each module. Use semicolon (;) to separate commands. Use double quotes to surround each compare command (see "Examples" section below).

`-Map <filename>`	Uses the specified file for key point mapping. (You must use the `WRITE MAPPED POINTS` command before using this option.)
`-PREPEND_String <string>`	Appends any string of commands to the partition dofile script *before* key point comparison for each module. Use a semicolon (;) to separate commands. Enclose the string of prepend commands with double quotes (see "Examples" section below).
`-Replace`	Replaces the existing file.
`-Usage`	Executes the `USAGE` command after each comparison and at the end of the partition dofile.

Example

```
write partition dofile partition.do -replace
```

```
write partition dofile ptn.do -append_string "usage" -prepend_string "report
unmapped points -notmapped" -replace
```

In the following command, the default compare command is replaced with two commands during each module comparison, `set compare effort low` and `compare -abort_stop 1 -noneq_stop 1`:

```
write partition dofile -compare_string "set compare effort low; compare -abort_stop
1 -noneq_stop 1"
```

Related Commands

ADD PARTITION KEY_POINT

DELETE PARTITION KEY_POINT

REPORT PARTITION KEY_POINT

WRITE MAPPED POINTS

WRITE RULE CHECK

```
WRIte RUle Check
     <filename>
     [-DEsign | -LIbrary]
     [-Golden | -REVIsed]
     [-Replace]
     (Setup Mode)
```

Writes the rule violations into a rule file. Use this command the first time you run a session. For later runs, exclude the violations already flagged with the `read rule check -exclude <filename>` command.

Use the tilde character (~) to shorten the path of the file.

Tcl Command

```
write_rule_check
```

Parameters

`<filename>`	Writes rule check violations to the specified file.
`-DEsign`	Writes only design rule check violations. If you do not specify `-design` or `-library`, Conformal writes rule check violations from both designs and libraries.
`-LIbrary`	Writes only library rule check violations. If you do not specify `-design` or `-library`, Conformal writes rule check violations from both designs and libraries.
`-Golden`	Writes rule check violations from the Golden design. *This is the default.*
`-Revised`	Writes rule check violations from the Revised design.
`-REPlace`	Replaces the previously saved file.

Report Columns

This command generates a report listing the rule violations. For example:

```
Rule    Type Filename Line Col  Info Library Module Object
RTL6.1  57   test.v   20   22  16   0       test   (null)
```

where:

- Rule: Lists the rule name.
- Type: Lists the integer value of the rule type. Each rule has a unique rule name and a unique type.
- Filename: Name of the file where the violation occurs.
- Line: Line number where the violation occurs.
- Col: Column number where the violation occurs.
- Info: Set of internal flags.
- Library: When 0, the rule violation is in the design space. When 1, the rule violation is in the library space.
- Module: Name of the module where the violation occurs.
- Object: Text string used to display the message of the reported rule violation.

Examples

In the following example, the second `report rule check` will not report any rules.

```
read design g.v -golden
read design r.v -revised

write rule check rule.g -golden -replace
write rule check rule.r -revised -replace

read design g.v -golden -replace
read design r.v -revised -replace

report rule check -verbose -both

read rule check rule.g -exclude -golden
read rule check rule.r -exclude -revised

report rule check -verbose -both
```

Related Command

READ RULE CHECK

WRITE TEMPLATE

```
WRIte TEmplate
     [-Outfile <string>]
     [-show_tags]
     [-list]
     [tag_name ...]
     (Setup / LEC Mode)
```

Views and writes the Conformal template scripts with the necessary commands and attributes. To see all the available Conformal template scripts, run the command `write_template` with no option

Tcl Command

```
write_template
```

Parameters

`-Outfile <string>`	Specifies the file name to write the template script.
`-show_tags`	Lists out all possible tags.
`-list`	Displays all the available templates.
`tag_name ...`	Specifies the tag to select one template; If the tag matches with the unique template name, it will write out that template; otherwise, it will list out all the matching templates. User then can select the specified template to view or write out.

WRITE VERIFICATION INFORMATION

```
WRIte VErification Information
     [<verification_information_directory>]
     [-COMpute]
     [-GOLden <label>]
     [-REVised <label>]
     (Setup / LEC Mode)
```

Writes out verification information to the specified directory. This command is enabled by the `SET VERIFICATION INFORMATION` command.

Tcl Command

```
write_verification_information
```

Parameters

`verification_information_directory`	
	Writes the verification information to the specified directory.
	By default, this command uses the `verification_information_directory` specified by the `SET VERIFICATION INFORMATION` command.
`-COMpute`	Indicates that computation of verification information is needed.
`-GOLden <label>`	Specifies which set of information to write out for the golden side.
`-REVised <label>`	Specifies which set of information to write out for the revised side.

Example

This command can be used with `READ VERIFICATION INFORMATION` to compute a new set of information that can be used to optimize new verification.

Suppose verification information for RTL versus G1 netlist is `rtlg1.uvi`, and verification information for G1 versus G2 netlists is `g1g2.uvi`. Then, verification information that can be used for directly verifying RTL versus G2 can be computed as:

```
set verification information rtlg2.uvi
read verification information rtlg1.uvi -golden rtl -revised g1
read verification information g1g2.uvi -golden g1 -revised g2
write verification information -golden rtl -revised g2 -compute
```

A special case is to get the information for a new verification with golden and revised netlists swapped:

```
set verification information g1g2.uvi
read verification information g1g2.uvi -golden g1 -revised g2
write verification information -golden g2 -revised g1 -compute
```

Related Commands

READ SETUP INFORMATION

READ VERIFICATION INFORMATION

SET VERIFICATION INFORMATION

3

ECO Command Reference

This chapter describes the Encounter® Conformal® ECO commands. The commands are presented in alphabetical order.

ADD ECO CUTPOINT

```
ADD ECo Cutpoint
     <name>
     [-GOLden [[+|-] <gate>]*]
     [-REVised [[+|-] <gate>]*]
     (LEC Mode)
```

This command requires an ECO license.

Adds an ECO cutpoint. The cutpoint will be used by the `ANALYZE ECO` command and is intended to help reduce the complexity of the ECO analysis by reducing the size of the ECO patch.

Tcl Command

```
add_eco_cutpoint
```

Parameters

`<name>`	Specifies the name for the ECO cutpoint.
`-Golden [[+|-] <gate>`	Flattens the gates in the Golden design with either a positive or negative phase. When neither +/- is specified, the phase defaults to positive.
`-Revised`	Flattens the gates in the Revised design with either a positive or negative phase. When neither +/- is specified, the phase defaults to positive.

Example

```
add eco cutpoint CUT1 -GOLDEN + U30/g1 -REVISED - U13/g1
add eco cutpoint CUT2 -GOLDEN U31/g1 - U32/g2 -REVISED + U13/g1
analyze eco cutpoint
analyze eco patch.v
```

Related Commands

ANALYZE ECO

ANALYZE ECO CUTPOINT

DELETE ECO CUTPOINT

REPORT ECO CUTPOINT

ADD ECO PIN

```
ADD ECo Pin
     <module_name>
     <pin_name> | <bus_name <size>> ...
     [-INput | -OUTput | -IO]
     [-FORce]
     (Setup Mode)
```

This command requires an ECO license and works only on the Golden design.

Adds a pin to a module. If the Revised module has extra ports, you can use this command to add new pins to the Golden module. You should add the extra pins before running the `WRITE HIER_COMPARE DOFILE` command.

Tcl Command

add_eco_pin

Parameters

`<module_name>`	Specifies the name of the module.
`<pin_name>`	Specifies the name of the pin(s).
`<bus_name <size>>`	Specifies the name of the bus(es), where `<size>` is `[msb:lsb]`, for example, `[16:0]`. You must include the braces in the command. See the Examples section.
`-INput`	Specifies that the pin is an input pin. *This is the default.*
`-OUTput`	Specifies that the pin is a output pin.
`-IO`	Specifies that the pin is an input/output pin.
`-FORce`	Renames the signal if it conflicts with the port name. See the Examples section.

Examples

- The following command adds `wr_req` input port and `data[15:0]` input bus to module `mod_A`

```
add eco pin mod_A wr_req data[15:0] -input
```

- This example shows how the command's `-force` option affects the following design:

  ```
  module top(x,y);
    wire z;
  endmodule
  ```

  - ❑ The following command will error out because there is a conflict between the net `z` and the new port `z`:

    ```
    add eco pin -output top z
    ```

  - ❑ The following command will not error out and net `z` will be renamed `z_1`:

    ```
    add eco pin -output top z -FORce
    ```

Related Commands

ANALYZE ECO

WRITE HIER_COMPARE DOFILE

ADD SPARE CELL

```
ADD SPare Cell
     [-DEFfile <filename>]
     [-FReedcell]
     [-SParecell <cell_instance_name*> ... ]
     [-GAFiller <cell_instance_name*> ... ]
     (Setup Mode)
```

This command requires an ECO-GXL license.

Adds the spare cells or freed cells as the available cells for the `OPTIMIZE PATCH` command.

When using `OPTIMIZE PATCH` with the `-usespare` option, the spare cell count for each cell type is considered. When using `OPTIMIZE PATCH` with the `-def` option, the spare cell instances are considered for location-aware optimization.

Tcl Command

```
add_spare_cell
```

Parameters

`-DEFfile <filename>`	Specifies the DEF filename (compressed gzip format is alo supported). This searches for the spare cell in the DEF file. By default, the spare cell is searched for in current hierarchy.
`-FReedcell`	Specifies that freed cells will be used for mapping.
`-SParecell <cell_instance_name*> ...`	Specifies the spare cells to be added. This accepts wildcards.
`-GAFiller <cell_instance_name*> ...`	Specifies the gate array filler to be added. This accepts wildcards.

Related Commands

DELETE SPARE CELL

OPTIMIZE PATCH

REPORT SPARE CELL

ANALYZE ECO

```
ANAlyze ECo
     <filename>
     [-REPlace | -APPend]
     [-EFFort <HIGH | MEDIUM | ULTRA>]
     [-POST_OPTImization]
     [-CONE_SWAP]
     [-CONNECT_ICG_TESTENABLE <AUTOmatic | MANual>]
     [-CREATE_DB [R1R2 | R1G1 | R2G2]<db_filename>]
     [-HIERarchical [-MODule <module name>][-ECOPIN_dofile <file_name>]]
     [-IMPORT_DB <db_filename>+]
     [-PLACEment_driven]
     [-PRESERVE_clock | -NOADD_ICG]
     [-THREADS <integer>[,<integer>]]
     [-CPULIMIT <integer>]
     [-USE_BUF_CHAIN_Source]
     (LEC Mode)
```

This command requires an ECO license.

Analyzes the ECO change in the Revised root module comparing to the Golden root module. The logic change is written to the specified patch file, which contains the Verilog module with the port names corresponding to the nets in the Golden design.

This command also enables the creation of the ECO cutpoint database (`-create_db` option). For more information on the usage of this command, refer to the "ECO Cut Point Flow" section of the *ECO User Guide*.

Note: Only the logic cone under NONEQ points are analyzed by the command.

Tcl Command

`analyze_eco`

Parameters

`<patch_filename>`	Specifies the name of the patch file.
`-REPlace`	Replaces the existing file.
`-APPend`	Append database to the existing database file.
`-EFFort <HIGH	MEDIUM	ULTRA>`	

Specifies the analysis effort level. The command default is `HIGH`.

`HIGH`: Enables grouping and more optimization steps

`MEDIUM`: Skips potentially slow groupings and optimization steps

`ULTRA`: Enables the post-optimization engine to further optimize the patch

Note: Effort `ULTRA` is available for beta testing. Its features/functionality may change prior to final release.

`-POST_OPTImization`	Enable the post-optimization engine to reduce the patch. This option is enabled automatically when you use the effort ULTRA.
`-CONE_SWAP`	Generates the maximum patch. This ECO patch contains the entire logic cone of the NEQ point from the G2 netlist. This can help preserve paths that have already been optimized for timing.
`-CONNECT_ICG_TESTENABLE`	`<AUTOmatic	MANual>`

Specifies that when new ICGs are created, their test enable pins will be connected to the test enable driver.

`AUTOmatic`	Specifies that the ICG test enable pin driver will be identified automatically by searching for the equivalent signal in the Golden design. For more information on this option, refer to the web interface (enabled using the `SET WEB INTERFACE` command) article titled Automatic Test Enable Connection for New Integrated Clock Gating Cell (ICG) in the Patch.

`MANual`	Specifies the ICG test enable pin driver is specified from the command `SET ECo ICG_Testenable`. For more information on this option, refer to the web interface (enabled using the `SET WEB INTERFACE` command) article titled User Specified Test Enable Connection for New Integrated Clock Gating Cell (ICG) in the Patch.

`-CREATE_DB [R1R2 | R1G1 | R2G2] <db_filename>`

Enables the creation of the ECO cutpoint database.

Note: This option is available for beta testing. The options and features described may change prior to final release.

`R1R2`	Analyzes the RTL cutpoint candidates and stores them into the the specified database name (*`db_filename`*).
`R1G1`	Reads in the `R1R2` database, which is imported using the `-import_db` option, and analyzes the `R1` and `G1` cutpoint candidates. The results are stored in a new database specified by the database name (*`db_filename`*).
`R2G2`	Reads in the `R1R2` database, which is imported using the `-import_db` option, and analyzes the R2 and G2 cutpoint candidates. The results are stored in a new database specified by the database name (*`db_filename`*).
`db_filename`	Specifies the new database name.

`-HIERarchical [-MODule <module name>`

Analyzes all the nonequivalent modules reported by `COMPARE ECO HIERARCHY`.

If `COMPARE ECO HIERARCHY` has not been executed, the tool calls it implicitly.

If the patch file name contains `%s`, it will be replaced by the module name in each ECO hierarchy. Otherwise, all the patches will output to the specified patch file.

`-MODule <MODULE_NAME>`	(Available with only the `-HIERarchical` option.) Analyze only the specified nonequivalent module
`-PLACEment_driven`	Enables the placement-driven ECO engine for better timing optimization.
`-PRESERVE_clock`	Attempts to minimize the changes to the existing clock network and may not apply to newly added registers.
`-NOADD_ICG`	Avoid adding integrated clock gating (ICG) in the patch. Implement clock gating functions using MUX feedback structures. This function covers only registers that already exist in G1 and does not cover newly added registers.
`-ECOPIN_dofile <FILE_NAME>`	Creates a dofile that adds ECO pins when there are extra pins in the Revised design (as compared to Golden design) and deletes ECO pins when there are extra pins in the Golden design (as compared to Revised design).
`-IMPORT_DB <db_filename>+`	Use this option to read in the R1R2 database required for the R1G1 and R2G2 modes of the `-create_db` option. **Note:** This option is available for beta testing. The options and features described may change prior to final release.
`-THREADS`	Specifies the minimum and maximum number of threads, separated by a comma. If only one number entered, this specifies both the minimum and maximum number of threads. For example: `-threads 2` specifies two threads; `-thread 2,4` specifies a minimum of two threads and a maximum of four threads. Each thread requires one Conformal ECO license. Multi-threading is only triggered when `ANALYZE ECO -effort HIGH` (the default effort) or `ANALYZE ECO -effort ULTRA` is used. **Note:** This is available for beta testing. Its features/ functionality may change prior to final release

`-CPULIMIT <integer>`	Specifies the run time limit (in seconds) for high and ultra effort level. After the specified time limit, the effort level is lowered to medium for the remainder of the command. The default is value 36000.
`-USE_BUF_CHAIN_SOURCE`	If the driver of the patch is a buffer/inverter chain, use the source of the buffer/inverter chain as the driver for the patch. Use this when you need to improve the timing of the ECOed (G3) netlist

Examples

In the following example, the `analyze eco "patch_%s.v" -hierarchical -ecopin_dofile ecopins.do` command will analyze the nonequivalent modules reported by `COMPARE ECO HIERARCHY`. The `%s` in `patch_%s` will be replaced by the module name in each ECO hierarchy.

```
setup> set eco option -flat
setup> set system mode lec
lec> analyze hier_compare -dofile hier.do -constraints -FUNCTION_Pin_mapping -
eco_aware -replace
lec> add compare point -all
lec> compare
lec> compare eco hierarchy
lec> report eco hierarchy -hierarchy
lec> analyze eco "patch_%s.v" -hierarchical -ecopin_dofile ecopins.do
lec> set system mode setup
lec> dofile ecopins.do
lec> apply patch -auto
```

The following illustrates the usage of the `-create_db` and `-import_db` options.

```
analyze eco -create_db r1r2 r1r2.db
//Creates the R1R2 database
analyze eco -create_db r1g1 r1g1.db -import_db r1r2.db
//Creates the R2G2 database and imports the R1R2 database.
analyze eco -create_db r2g2 r2g2.db -import_db r1r2.db
report eco cutpoint -auto -file cutpoint.do -import r1g1.db r2g2.db
analyze eco cutpoint
dofile cut.do
```

Related Commands

ADD ECO PIN

ANALYZE ECO

ANALYZE HIER_COMPARE

COMPARE ECO HIERARCHY

ANALYZE ECO CUTPOINT

```
ANAlyze ECo Cutpoint
```
(*LEC Mode*)

This command requires an ECO license.

Analyzes the ECO cutpoints and their effect on the existing nonequivalent points. For more information on the usage of this command, refer to the "ECO Cut Point Flow" section of the *ECO User Guide*.

Note: This command is available for beta testing. The options and features described may change prior to final release.

Tcl Command

```
analyze_eco_cutpoint
```

Parameters

None.

Related Commands

ADD ECO CUTPOINT

DELETE ECO CUTPOINT

ANALYZE ECO

REPORT ECO CUTPOINT

APPLY PATCH

```
APPly PAtch
     [<module_under_ECO_name> <patch_module_name> | -AUTO]
     [-KEEPHierarchy | -NOKEEPHierarchy]
     [-NETnaming <format_string>]
     [-INStancenaming <format_string>]
     [-BBOXnaming <format_string>]
     [-SEQuentialnaming <format_string>]
     [ | -KEEPFREED | -TIEFREED0 | -TIEFREED1]
     [-FREESCAN]
     [-FREENONSCAN <0|1>]
     [-STITCHSCAN]
     [-SHIFTEnable <net_name> <0|1>]...
     [-Golden | -Revised]
     (Setup Mode)
```

This command requires an ECO license.

Applies the ECO change specified in the patch module, generated by the `ANALYZE ECO` command, to the module under ECO. The patched module can be written out with the `WRITE DESIGN` command.

Note: The patch generated by the `ANALYZE ECO` command can contain unmapped primitives.

Tcl Command

`apply_patch`

Parameters

`<module_under_ECO_name>`	
	Specifies the name of the module being changed for ECO.
`<patch_module_name>`	Specifies the name of the patch module containing the ECO changes.
`-AUTO`	Automatically reads in and applies all patches that were created with the `ANALYZE ECO` command in the current session.
`-KEEPHierarchy`	Specifies that the ECO changes will be put in a submodule. This is the default.

`-NOKEEPHierarchy`	
	Do not put ECO changes in a submodule.
`-NETnaming <format_string>`	
	Specifies the net naming format of the ECO nets. For example, for `eco_net_%d`, the `%d` will be an integer that makes the net name unique.
`-INStancenaming <format_string>`	
	Specifies the instance naming format of the ECO combinational cells. For example, for `eco_instance_%d`, the `%d` will be an integer that makes the instance name unique.
`-BBOXnaming <format_string>`	
	Specifies the naming format of the ECO blackboxes. For example, `eco_bbox_%s`, where `%s` is the original blackbox name.
`-SEQuentialnaming <format_string>`	
	Specifies the instance naming format of the ECO registers and latches. For example, `eco_%s`, where `%s` is the original register name.
`-KEEPFREED`	Retains all freed instances and leaves input pins connected. Freed instance will not be re-used for mapping the patch.
`-TIEFREED0`	Retains all freed instances and applies a value of '0' (tie low) to the input pins of any freed instance.
`-TIEFREED1`	Retains all freed instances and applies a value of '1' (tie high) to the input pins of any freed instance.
`-FREESCAN`	Free up the DFFs reachable only by scan chain.
`-FREENONSCAN <0|1>`	Do not free up the DFFs reachable only by scan chain, but tie the non-scan related input pins of the DFFs to a constant. The set and reset pins of the DFFs are not affected.
`-STITCHSCAN`	Reconnect the scan chain if some DFFs are removed or some submodule scan out connection are affected by ECO changes. Note: In the case where DFFs are added, the newly-added DFFs are not stitched.
`-SHIFTEnable <net_name> <0|1>`	
	Specify the scan chain is enabled when net value is 0 or 1.

`-Golden`	Applies to the Golden design. *This is the default.*
`-Revised`	Applies to the Revised design.

Example

The following command automatically applies the ECO changes in the submodules of modules under ECO:

```
apply patch -auto -keephierarchy
```

In addition to applying ECO changes, the following command frees up the DFFs that are reachable only by the scan chain and reconnects the scan chain based on the given scan chain configuration. The scan chain is enabled when `test_mode` and `scan_enable` are set.

```
apply patch -auto -keephierarchy -freescan -stitchscan -shiftenable test_mode 1 -
shiftenable scan_enable 1
```

Related Commands

OPTIMIZE PATCH

CHECK ECO SETUP

```
CHEck ECo Setup
     [-VERbose]
```
     (Setup/LEC Mode)

This command requires an ECO license.

Performs ECO-related checks that will flag potential issues related to setup and offer potential solutions early. For more information on this command and its usage, refer to the web interface article *ECO Setup Checks.* To launch the web interface, use the "`set web on`" command.

Tcl Command

```
check_eco_setup
```

Parameters

`-VERbose`	Displays detailed information regarding the setup checking.

COMPARE ECO HIERARCHY

```
COMpare ECo Hierarchy
      [-CONE_SWAP | -CONE_SWAP_FILE <file_of_swap_points>]
      [-HIGH_FANOUT_CUT_INSERTion_for_gated_clock
      | -NOHIGH_FANOUT_CUT_INSERTion_for_gated_clock]
      [-VERbose]
      [-THREADS <integer>[,<integer>]]
      (LEC Mode)
```

This command requires an ECO license.

This command is used in the flattened ECO flow. Use this command in LEC mode, and after running the `ANALYZE HIER_COMPARE` and `COMPARE` (complete flattened compare) commands. This command takes the flattened compare results, along with the boundary information, from `ANALYZE HIER_COMPARE` and breaks down the nonequivalent points to their associated submodules. The flattened ECO flow eliminates any possible false nonequivalent points that arise from the hierarchical compare process (such as boundary optimization).

Note: The results that you get from this command might be different from the results of the flattened comparison. For example, at the module level there might be only one nonequivalent primary output. With the flattened compare, this same primary output might fanout to 10 DFFs that result to 10 nonequivalent points.

Tcl Command

```
compare_eco_hierarchy
```

Parameters

`-CONE_SWAP`	Performs cone swapping for all NEQ key points.
`-CONE_SWAP_FILE`	Specifies a file that contains the key points for cone swapping.
`-HIGH_FANOUT_CUT_INSERTion_for_gated_clock`	Analyze high-fanout gated-clock cells and insert related cut points. *This is the default*.
`-NOHIGH_FANOUT_CUT_INSERTion_for_gated_clock`	Disable the high-fanout gated-clock analysis.
`-VERbose`	Displays the non-equivalent compare points for each hierarchy.

`-THREADS`	Specifies the minimum and maximum number of threads, separated by a comma. If only one number entered, this specifies both minimum and maximum number of threads. For example: `-threads 2` specifies two threads; `-threads 2,4` specifies a minimum of two threads and a maximum of four threads. Each thread requires one Conformal ECO license. **Note:** This is available for beta testing. Its features/functionality may change prior to final release.

Example

In the following example, `COMPARE ECO HIERARCHY` analyzes and breaks down the boundaries identified by the `ANALYZE HIER_COMPARE` command:

```
setup> set flatten model -enable_analyze_hier_compare
setup> set system mode lec
lec> analyze hier_compare -dofile hier.do -constraints -FUNCTION_Pin_mapping -
eco_aware -replace
lec> add compare point -all
lec> compare
lec> compare eco hierarchy
lec> report eco hierarchy -hierarchy
lec> analyze eco patch.v -hierarchical -ecopin_dofile ecopins.do
lec> set system mode setup
lec> dofile ecopins.do
lec> apply patch -auto
```

Related Commands

ANALYZE HIER_COMPARE

ANALYZE ECO

REPORT ECO HIERARCHY

DELETE ECO CUTPOINT

```
DELete ECo Cutpoint
     [-ALL | <name>*]
     (LEC Mode)
```

This command requires an ECO license.

Deletes an ECO cutpoint.

Note: This command is available for beta testing. The options and features described may change prior to final release.

Tcl Command

```
delete_eco_cutpoint
```

Parameters

`-ALL`	Delete all ECO cutpoints.
`<name>*`	Specifies the name of the ECO cutpoint to delete.

Related Commands

ADD ECO CUTPOINT

ANALYZE ECO

ANALYZE ECO CUTPOINT

DELETE ECO PIN

```
DELete ECo PIn
     <module_name>
     <pin_name> | <bus_name> ...
     (Setup Mode)
```

Deletes the pins from the module.

Note: Primary input pins of top module cannot be deleted if they are reachable. Input and output pins cannot be deleted if they drive logics.

Tcl Command

`delete_eco_pin`

Parameters

`<module_name>`	Specifies the name of the module.
`<pin_name>`	Specifies the name of the pin(s).

`<bus_name> ...`	Specifies the name of the bus(es).
	Note: You cannot delete only partial bits. For example, if there is a bus port `OUT [1:0]`, the command should be:
	`delete eco pin top OUT`
	`not`
	`delete eco pin top OUT[1:0]`

Related Commands

ADD ECO PIN

REPORT MISMATCHED PIN

DELETE SPARE CELL

```
DELete SPare Cell
     [-FReedcell <cell_name*> ... ]
     [-SParecell <cell_name*> ... ]
     [-GAfiller <cell_name*> ...]
     [-CELLtype <cell_type_name*> ....]
     [-ALL]
     (Setup Mode)
```

This command requires an ECO license.

Deletes the spare cells or freed cells that were added with the `ADD SPARE CELL` command.

Tcl Command

`delete_spare_cell`

Parameters

`-FReedcell`	Specifies the name(s) of the freed cell(s) to be deleted. This accepts wildcards.
`-SParecell`	Specifies the name(s) of the spare cell(s) to be deleted. This accepts wildcards.
`-GAfiller`	Specifies the name of the gate array cell(s) to be deleted. This accepts wildcards.
`-CELLtype`	Specifies the name(s) of library cell type(s) to be deleted. This accepts wildcards.
`-ALL`	Deletes all the cells.

Related Commands

ADD SPARE CELL

OPTIMIZE PATCH

REPORT SPARE CELL

OPTIMIZE PATCH

```
OPTimize PAtch
     -WORKdir <working_directory>
     <-LIBrary <library_file_list> | -LIBSETUPscript <tcl_script>>
     [-SYNExec <synthesis_executable>]
     [-SDC <sdc_file_list>]
     [-VERbose]
     [-NOKEEPHierarchy | -KEEPHierarchy]
     [-CLEANUP]
     [-AVOID <cell_name>*]
     [-USE <cell_name>*]
     [-NETnaming <format_string>]
     [-INStancenaming <format_string>]
     [-BBOXnaming <format_string>]
     [-SEQuentialnaming <format_string>]
     [-CAPtable <filename>]
     [-USESPARE]
     [-LEF <filename> ...]
     [-DEF <filename> ...]
     [-PLE]
     [-MAPscript <filename>]
     [-MMMC <mmmc_file>]
     [-GAlibcell <cell_name*> ...]
     [-1801 <1801_filename>]
     [-CPF <cpf_filename>]
     [-RELATIVE2CWD]
     [-NOUSETIEcell]
     [-PRELIBscript <script_name>]
     [-PRESYNscript <script_name]
     [-POSTSYNscript <script_name>]
     [-SPLIT_non_structure <AUTO|NETLIST_ONLY|STRUCT_NETLIST_ONLY>]
     [-SUPPRESS_SDCError]
     [-PREMASKPHYsical [-PLACEscript <script_file_name>]]
     [-ALLOW_FC_MF]
     [-NOFLATTEN_SMALL]
     [-ALLOWTIECELLINVersion]
     [-QRCtech <filename>]
     (Setup Mode)
```

Writes out a Genus script (`cfm_eco_rc.tcl`) in the working directory that will optimize the patches and execute the script.

After `OPTIMIZE PATCH` successfully completes, the ECO design will be in memory and can be written out. The optimized patches will be in the working directory.

Important

Before running this command, you must run the `APPLY PATCH -keephierarchy` command.

Tcl Command

`optimize_patch`

Parameters

`-WORKdir <working_directory>`

Specifies the name of the working directory for the optimized patches.

`-LIBrary <library_file_list>`

Specifies the name of the library files.

If the library name contains a relative path, the path should be relative to the directory declared in `-workdir`. Alternatively, you can use the `-LIBSETUPscript` option to provide a file that specifies the libraries.

Note: Library files may also be specified in the CPF file. If they are and the `OPTIMIZE PATCH -library` option is used, both libraries will be read into Genus.

`-LIBSETUPscript <tcl_script>`

Specifies a library file setup script. Enables you to specify libraries in a file rather than the `-library` option. If the library name contains a relative path, the path should be relative to the directory declared in `-workdir`.

`-SYNExec <rc executable>`

Specifies the path to the Genus or RTL Compiler executable. If this is not specified, the software will use the Genus command in your search path.

`-SDC <sdc_file_list>`	Specifies the name of the SDC file(s) (compressed gzip format is also supported). If the SDC filename contains a relative path, the path should be relative to the directory declared in `-workdir`. **Note:** SDC files may also be specified in the CPF file. If they are and the `OPTIMIZE PATCH -SDC` option is used, both SDC's will be read into Genus.
`-VERbose`	Outputs all Genus messages. By default, the command only outputs error messages.
`-NOKEEPHierarchy`	Do not put ECO changes in a submodule. *This is the default.*
`-KEEPHierarchy`	Specifies that the ECO changes will be put in a submodule.
`-CLEANUP`	Deletes the generated files.
`-AVOID <cell_name>*`	Avoids the specified library cells. This accepts wildcards. When this option is used in combination with the `-USESPARE` option, only the cell quantities of those cell types not specified by `-AVOID` are available for mapping.

`-USE <cell_name>*`	Uses the specified library cells. This accepts wildcards.
	
	When this option is used in combination with the `-USESPARE` option, only the cell quantities of those cell types specified by `-USE` are used for mapping.
	
	Note: The order that you specify the `-AVOID` and `-USE` options is significant. For example:
	
	`"-avoid * -use NAND2 INV2"`
	
	avoids all the library cell types except `NAND2` and `INV2`. If these options are specified in the following order:
	
	`"-use NAND2 INV2 -avoid *"`
	
	then no cell types will be available for mapping.

`-NETnaming <format_string>`

Specifies the net naming format of the ECO nets. For example, for `eco_net_%d`, the `%d` will be an integer that makes the net name unique.

`-INStancenaming <format_string>`

Specifies the instance naming format of the ECO combinational cells. For example, for `eco_instance_%d`, the `%d` will be an integer that makes the instance name unique.

`-BBOXnaming <format_string>`

Specifies the naming format of the ECO blackboxes. For example, `eco_bbox_%s`, where `%s` is the original blackbox name.

`-SEQuentialnaming <format_string>`

Specifies the instance naming format of the ECO registers and latches. For example, `eco_%s`, where `%s` is the original register name.

`-CAPtable <filename>`

Specifies the name of the CAP file. This option requires an ECO GXL license.

`-USESPARE` Attempts to implement the ECO by mapping to spare gates. This option requires an ECO GXL license.

`-LEF <filename> ...` Specifies the LEF file(s) (compressed gzip format is also supported). This calls Genus physical synthesis to perform location-aware spare-gate mapping.

This option requires an ECO GXL license.

`-DEF <filename> ...` Specifies the DEF file(s) (compressed gzip format is also supported). This calls Genus physical synthesis to perform location-aware spare-gate mapping.

This option requires an ECO GXL license.

`-PLE` Run Genus physical synthesis with LEF only. DEF is optional.

This option requires an ECO GXL license.

`-MAPscript <filename>`

	Writes out the location aware spare gate mapping result in the form of an SoC Encounter TCL script. This option should be used with the `-DEF` option. This option requires an ECO GXL license.
`-MMMC <mmmc_file>`	Specifies the name of the MMMC view definition file to be read.
`-GAlibcell <cell_name*> ...`	Specifies the gate array library cells. This option requires an ECO GXL license.
`-1801 <filename>`	Specifies power intent in the 1801 format. This option requires a Conformal ECO GXL license.
`-CPF <cpf_filename>`	Specifies the common power format (CPF) file name. **Note:** Library and SDC files may also be specified in the CPF file. If they are and the `OPTIMIZE PATCH -LIBRARY` and/or `-SDC` option are used, both will be read into Genus. The option requires a Conformal ECO GXL license.
`-RELATIVE2CWD`	Specifies that the paths provided in other options relate to the current working directory: not the Genus working directory.
`-NOUSETIEcell`	Allows Genus to leave constants as assign statements instead of using tie high or tie low cells.
`-PRELIBscript <script_name>`	Specifies the script to run before reading in the libraries.

`-PRESYNscript <script_name>`

Specifies the script to run before each patch is synthesized. The script can have two special procedures defined in it: `eco_pre_synthesis_cmd` and `eco_post_synthesis_cmd`. Both procedures accept patch name as a parameter and return no value. The procedures, if defined, are called by Genus before and after synthesizing each patch module. The most common use of the `eco_pre_synthesis_cmd` procedure is setting attributes on the patch module. For example, this script can contain the following lines:

```
# Set a root level attribute
set_attribute information_level 9 /

proc eco_pre_synthesis_cmd {patch_name} {
# Set an attribute on a patch
set_attribute dft_scan_map_mode force_all
/designs/$patch_name
}
```

`-POSTSYNscript <script_name>`

Specifies the script to run after all patches are synthesized.

`-SPLIT_non_structure <AUTO|NETLIST_ONLY|STRUCT_NETLIST_ONLY>`

`AUTO`: Split structural and non-structural modules into two files and use different Genus parameter '`-netlist`' and '`-struct_netlist`' to elaborate. *This is the default*.

`NETLIST_ONLY`: Write all modules into a file and use only '`-netlist`' Genus parameter.

`STRUCT_NETLIST_ONLY`: Write all modules into a file and use only '`-struct_netlist`' Genus parameter.

`-SUPPRESS_SDCError` Suppresses errors related to the reading in of the SDC file.

`-PREMASKPHYsical [-PLACEscript <script_file_name>]`

	This option requires an ECO GXL license.
	`-PREMASKPHYsical`: Use this option to specify that the optimization is for physical-aware premask ECO.
	Use this option with the `-DEF` and `-LEF` options.
	`-PLACEscript`: Specifies the name of the placement script to output (in Innovus Tcl script format). To create the placement script, this option must be used with the `-PREMASKPHYsical` option.
	The placement script is read in using the `WRITE ECO DEF -placescript` command.
`-ALLOW_FC_MF`	Continue the run (in the pre-mask physical flow) even if filler and metal cells exist in the design. Note: filler and metal cells are not recommended in DEF files for the pre-mask physical flow.
	This option requires an ECO GXL license.
`-NOFLATTEN_SMALL`	Do not flatten (keep the hierarchy) of patches smaller than 10k gates.
`-ALLOWTIECELLINVersion`	Allows tie cells with inverters if one of the tie hi/lo cells is not found.
`-QRCtech <filename>`	Specifies the name of the qrctech file. This option requires an ECO GXL license.
	Note: qrctech files are supported in Genus, but not in RTL Compiler (RC).

Examples

- The following script reads in the original netlist and patch, then runs `OPTIMIZE PATCH` to map and optimize the ECO changes:

```
read library typical.lib -liberty

read design top.gv patch1.v

apply patch mod1 mod1_eco -keephierarchy

optimize patch -workdir rc_work -library ../typical.lib

write design top.eco.gv -replace
```

- The following specifies the library file in the current directory:

  ```
  optimize patch -WORK WORK -lib ../typical.lib
  ```

  or

  ```
  optimize patch -WORK -relative2cwd -lib typical.lib
  ```

- The following specifies multiple SDC files:

  ```
  optimize patch -sdc {top1.sdc top2.sdc top3.sdc}
  ```

- The following specifies running Genus in legacy UI mode which is compatible with RTL compiler commands:

  ```
  optimize patch -synexec "genus -legacy_ui" -library ../
  typical.lib -workdir genus_work
  ```

Related Commands

APPLY PATCH

ADD SPARE CELL

WRITE ECO DEF

REPORT ECO CHANGES

```
REPort ECo CHanges
     [-MODule <module_name>]
     [-BUFFER <cell_name>]
     [-DETAIL_SEQ_REPort]
     [-FILE <filename> [-REPlace]]
     [-SUMmary | -SCRipt | -GENUS | <-INNovus | -ENCounter>
        [-NOSUPPORT_CELLTYPE_CHANGE | -SUPPORT_CELLTYPE_CHANGE]
        [-NOSUPPORT_INSTNAME_CHANGE | -SUPPORT_INSTNAME_CHANGE] ]
     [-TIE1 <cell_name>] [-TIE0 <cell_name>]
     [-SUPport | -FRONTIER]
     (Setup Mode)
```

This command requires an ECO license.

Reports the ECO changes. To report the ECO change with this command, the ECOs must have been applied with the `APPLY PATCH` or `OPTIMIZE DESIGN` command.

Tcl Command

```
report_eco_changes
```

Parameters

`-MODule <module_name>`	Specifies the module to report. By default, the command reports all nets and instances that have been added and deleted.
`-BUFFER <cell_name>`	Use the specified cell to replace `assign` statements.
`-DETAIL_SEQ_REPort`	Report detailed information about sequential elements. Displays a list of newly added ICG's test-enable connection information..
`-FILE <filename>`	Redirects the output file.
`-REPlace`	Replaces the file.
`-SUMmary`	Shows a summary only. *This is the default.*

`-SCRipt`	Reports ECO changes and then generates a generic script that you can modify to use in different tools. For information on the commands within this script, refer to "Generic Script Format" in the *Conformal ECO User Guide*.
`-GENUS`	Generates a script for the Genus Implementation System.
`-ENCounter`	Generates a script for the Encounter Digital Implementation System.
`-INNovus`	Generates a script for the Innovus Implementation System.
`-NOSUPPORT_CELLTYPE_CHANGE`	Not to detect in-place cell type change and disables Innovus Tcl command `dbChangeInstCell`. *This is the default*.
`-SUPPORT_CELLTYPE_CHANGE`	Detects in-place cell type change and enables Innovus Tcl command `dbChangeInstCell`.
`-NOSUPPORT_INSTNAME_CHANGE`	Not to detect in-place instance name change and disables Innovus Tcl command `changeInstName`. *This is the default*.
`-SUPPORT_INSTNAME_CHANGE`	Detects in-place instance name change and enables Innovus Tcl command `changeInstName`.
`-TIE1 <cell_name>`	Use the specified cell as the tie high cell.
`-TIE0 <cell_name>`	Use the specified cell as the tie low cell.
`-SUPport`	Reports the key points that drive logic in the patch.
`-FRONTIER`	Reports the key points that are driven by the patch logic.

Related Commands

WRITE ECO DESIGN

WRITE ECO DEF

REPORT ECO CHECK

```
REPort ECo Check
     [-ALL | <check_name* ...>]
     [-Summary | -Verbose]
     [-Help]
     [-Ignore]
     [-Note]
     [-Warning]
     [-Error]
     (Setup Mode)
```

This command requires an ECO license.

Displays the list of ECO check violations. The ECO checks are available when the required information is ready. Conformal ECO automatically executes ECO checks after the commands `set_system_mode lec`, `compare`, `compare_eco_hierarchy`, `analyze_eco` and `write_hier_compare_dofile`.

Note: All ECO check violations with a severity of "Ignore" are only reported with the -ignore option.

Tcl Command

```
report_eco_check
```

Parameters

`-ALL`	Reports all check violations. This is the default.
`<check_name*>...`	Reports the specified check violations. Wildcards are supported.
`-Summary`	Displays the summary of ECO check violations. Wildcards are supported.
`-Verbose`	Displays the details of ECO check violations.
`-Help`	List the names, descriptions and default severity of all ECO checks.
`-Ignore`	Displays the ECO check violations that have a severity level of ignore.

`-Note`	Displays the ECO check violations that have a severity level of note.
`-Warning`	Displays the ECO check violations that have a severity level of warning.
`-Error`	Displays the ECO check violations that have a severity level of error.

Related Commands

SET ECO CHECK

SET DOFILE ABORT

REPORT ECO CUTPOINT

```
REPort ECo CUtpoint
     [-CANDidates|-AUTO]
     [-IMPORT_DB <db_file_names>*]
     [-FILE <filename>]
     (LEC Mode)
```

This command requires an ECO license.

Reports on ECO cutpoints. For more information on the usage of this command, refer to the "ECO Cut Point Flow" section of the *ECO User Guide*.

This command has three functions, it can report on added cutpoints, report on cutpoint candidates, or report on suggested cutpoints.

Note: This command is available for beta testing. The options and features described may change prior to final release.

Tcl Command

```
report_eco_cutpoint
```

Parameters

`-CANDidates`	Reports on all the existing cutpoint candidates in the database(s) specified by `-import`.
`-AUTO`	Reports on all the suggested cutpoints for the dtabase(s) specified by `import`
`-IMPORT_DB <db_file_names>*`	Specifies the database file names.
`-FILE <filename>`	Redirects the output of this command to the specified file.

Related Commands

ANALYZE ECO

ANALYZE ECO CUTPOINT

REPORT ECO GATEDCLOCK

```
REPort ECo Gatedclock
     [-XOR_only]
     [-NONEQuivalent]
     [-BOTH | -Golden | -Revised]
     (LEC Mode)
```

This command requires an ECO license.

Analyzes and reports the gate structure of DFFs.

Tcl Command

```
report_eco_gatedclock
```

Parameters

`-XOR_only`	Report only the XOR clock gating information.
`-NONEquivalent`	Analyze only the fan-in cone of the nonequivalent points.
`-BOTH`	Apply to both Golden and Revised designs. This is the default.
`-Golden`	Apply to the Golden design. .
`-REvised`	Apply to the Revised design.

Related Commands

ANALYZE ECO

REPORT ECO HIERARCHY

```
REPort ECO Hierarchy
     [-HIERarchy ]
     [-VERbose]
     (LEC Mode)
```

This command requires an ECO license.

Reports the nonequivalent modules found by `COMPARE ECO HIERARCHY`.

Tcl Command

```
report_eco_hierarchy
```

Parameters

`-HIERarchy`	Reports nonequivalent modules hierarchically.
`-VERbose`	Reports nonequivalent compare points for each hierarchy.

Related Commands

ANALYZE HIER_COMPARE

COMPARE ECO HIERARCHY

ANALYZE ECO

REPORT ECO ICG_TESTENABLE

```
REPort ECo Icg_testenable
```
(*Setup/LEC Mode*)

Displays a list of all user specified test-enable signals to drive the new ICGs with the SET ECO ICG_TESTENABLE command.

Tcl Command

```
report_eco_icg_testenable
```

Related Commands

REPORT ECO ICG_TESTENABLE

REPORT ECO PATCH

`REPort ECo PAtch`
    (*Setup Mode*)

This command requires an ECO license.

Reports the ECO patch specified by the `ADD ECO PATCH` command.

Tcl Command

```
report_eco_patch
```

REPORT ECO PIN

```
REPort ECo PIn
     [-MODule <module_name>]
     [-GOLden|-REVised]
     (Setup Mode)
```

This command requires an ECO license.

Reports the ECO pins specified by the `ADD ECO PIN` command.

Tcl Command

```
report_eco_pin
```

Parameters

`-MODule`	Reports only pins for the specified module.
`-GOLden`	Reports the ECO pins for the Golden design. *This is the default*.
`-REVised`	Reports the ECO pins for the Revised design.

Related Commands

ADD ECO PIN

REPORT ECO SCAN INPUT

```
REPort ECo Scan_input
     [-FILE <filename> [-REPlace]]
     [-GOLden | -REVised]
     (LEC Mode)
```

Collects the scan input pins and reports their respective driving flops or signals.

Tcl Command

```
report_eco_scan_input
```

Parameters

`-FILE <filename>`	Redirects the output to the specified file.
`-REPlace`	Allows the command to overwrite the output file if one exists. Otherwise, the command will error out.
`-GOLden`	Specifies that the report applies only to the Golden design. *This is the default*.
`-REVised`	Specifies that the report applies only to the Revised design.

Related Commands

REPORT ECO TESTENABLE INPUT

```
REPort ECo Testenable_input
     [-FILE <report> [-REPlace]]
     [-GOLden | -REVised]
     (LEC Mode command)
```

This command collects the test-enable pins and reports their respective driving flops or signals.

Tcl Command

```
report_eco_testenable_input
```

Parameters

`-FILE <report>`	Redirects the output to the report file.
`-GOLden`	Specifies that the report applies only to the golden design. *This is the default*.
`-REVised`	Specifies that the report applies only to the revised design.
`-REPlace`	Allows the command to overwrite the output file if one exists. Otherwise, the command will error out.

Related Commands

REPORT MISMATCHED PIN

```
REPort MIsmatched Pin
     [-Golden <module_name>]
     [-Revised <module_name>]
     [-TYPE <INput | OUTput>]
     (Setup Mode)
```

This command requires an ECO license.

Reports the mismatched pins between the specified modules of Golden and Revised designs. When executed without any options, this command reports the mismatched pins between the root module of the Golden and Revised designs.

Note: If there are any renaming rules before this command, it checks the pins after renaming.

Tcl Command

```
report_mismatched_pin
```

Parameters

`-Golden <module_name>`	
	Reports the mismatched pins between the specified modules of the Golden design.
`-Revised <module_name>`	
	Reports the mismatched pins between the specified modules of the Revised design.
`-TYPE`	Reports only mismatched pins of the specified type. When specified without a type, all types are reported.

Related Commands

ADD ECO PIN

REPORT PATHLEVELS

```
REPort PATHLevels
     < <from_gate> <to_gate> | -NONEQ | -DIFF_pathlevels <input file> > [-
     OUTputfile <filename>]
     [-SOURCE]
     [-NET]
     [-CELLView]
     [-VERbose]
     [-REPlace]
     [-Golden | -Revised]
     (LEC Mode)
```

This command requires an ECO license.

Displays the longest fan-in path depth of a gate, or between two selected gates, and saves to a file. This command can also report the path level change for a specified fan-in of a gate in a different run. Gates can be specified as gate ID numbers or instance paths.

Tcl Command

`report_pathlevels`

Parameters

`<from_gate><to_gate>`	Displays the longest fan-in path between the two specified gates. Gates can be specified as gate ID numbers or instance paths
`-NONEQ`	Displays the longest path depth for each non-equivalent key point.
`-DIFF_pathlevels <filename>`	Report path level changes for the fan-in of the specified gate. The specified file must be the output of the `REPORT PATHLEVELS` command with `-outputfile` option.
`-OUTputfile <filename>`	Outputs the report to the specified file.

`-SOURCE`	Displays the file and line number location of the gate in the path.
`-NET`	Displays the corresponding net of the gate in the path.
`-CELLVIEW`	Displays the path in instance-cell view.
`-Golden`	Reports the specified path in the Golden design. This is the default.
`-Revised`	Reports the specified path in the Revised design.

Example

In the following example, `REPORT PATHLEVEL` is called after the compare and outputs the report to report.log. After the patch is applied, `REPORT PATHLEVEL` is called again to report the longest paths and path difference as compared to the last report.

```
...
compare
report pathlevels -noneq -outputfile report.log ...
set sys mode setup
apply patch
set sys mode lec
report pathlevels -diff_pathlevels report.log -output file diff.log ...
```

The following is the sample contents of report.log:

```
//CFMDepthDiff g1/reg3/U$1 3 5
Maximum logic path levels to destination: 5 Destination : (G) 21 DFF /g1/reg3/U$1
Source :(G)6PI/in2

//CFMDepthDiff y6 0 1
Maximum logic path levels to destination: 1 Destination : (G) 16 PO /y6 Source
:(G)25Z/y6

//CFMDepthDiff g1/reg4/U$1 3 5
Maximum logic path levels to destination: 5 Destination : (G) 22 DFF /g1/reg4/U$1
Source :(G)8PI/in4

//CFMDepthDiff g1/reg5/U$1 3 5
Maximum logic path levels to destination: 5 Destination : (G) 23 DFF /g1/reg5/U$1
Source :(G)9PI/in5

Total: 4 logic cones are reported.
Maximum depth is: 5
```

The following is the sample of diff.log

```
//CFMDepthDiff g1/reg3/U$1 3 5
Maximum logic path levels to destination: 8(+3) Destination : (G) 21 DFF
/g1/reg3/U$1 Source :(G)6PI/x1

//CFMDepthDiff y6 0 1
Maximum logic path levels to destination: 6(+5) Destination : (G) 16 PO /y6
```

```
Source:(G)25Z/y2

//CFMDepthDiff g1/reg4/U$1 3 5
Maximum logic path levels to destination: 3(-2) Destination : (G) 22 DFF /g1/reg4/
U$1 Source :(G)4PI/in4

//CFMDepthDiff g1/reg5/U$1 3 5
Maximum logic path levels to destination: 5(+0) Destination : (G) 23 DFF /g1/reg5/
U$1 Source :(G)9PI/in5

Summary Report :
==============================================
Path-Levels-Difference  Total
==============================================
-2                       1
0                        1
+3                       1
+5                       1
----------------------------------------------
```

Related Commands

APPLY PATCH

REPORT GATE

REPORT PATH

REPORT SCAN CHAIN

```
REPort SCan Chain
     -SE <<net_name> <0|1>> ...
     -SI <net_name>
     [-SO <net_name>]
     [-Golden | -REvised]
     (Setup Mode)
```

This command requires an ECO license.

Reports the scan-chain based on the given scan chain configuration.

Tcl Command

```
report_scan_chain
```

Parameters

-SE	Specifies that the scan chain is enabled when the given net value is 0 or 1. You can specify the value of this option more than once.
-SI	Specify the scan input net for the scan chain.
-SO	Specify the scan output net for the scan chain.
-Golden	Apply to the Golden design. This is the default.
-REvised	Apply to the Revised design.

Example

The following command reports the scan chain based on the given scan chain configuration. The scan chain is enabled when `test_mode` and `scan_enable` are set.

```
report scan chain -se test_mode 1 scan_enable 1 -si scan_input -so scan_output
```

Related Commands

APPLY PATCH

REPORT SPARE CELL

```
REPort SPare Cell
     [-FReedcell]
     [-GAfiller]
     [-SParecell]
     [-USED]
     [-UNUSED]
     [-SUMMARY]
     (Setup Mode)
```

This command requires an ECO-GXL license.

Reports the spare cells and freed cells that were added with the `ADD SPARE CELL` command.

Tcl Command

```
report_spare_cell
```

Parameters

`-FReedcell`	Reports the freed cells only.
`-GAfiller`	Report the gate array filler cells only.
`-SParecell`	Reports the spare cells only.
`-UNUSED`	Reports unused spare or freed cells.
`-USED`	Reports used spare or freed cells.
`-SUMMARY`	Specifies that instead of listing all instances, only a summary based on cell type is reported as: `<cell_name> : <number_of_cells>`

Related Commands

APPLY PATCH

ADD SPARE CELL

DELETE SPARE CELL

OPTIMIZE PATCH

SET DEF FILE

```
SET DEf File <file_name>
```

Specifies the DEF file for placement-driven ECO.

Tcl Command

```
set_def_file
```

Parameters

No parameters.

SET ECO ICG_TESTENABLE

```
SET ECo Icg_testenable
     [-NET | -PORT | -DRIVER_PIN <name>]
     [-MODULE_INST <path_name> | -MODULE <module_name>]
     [-INVERT]
     (Setup Mode)
```

When a new ICG is added to the patch as part of the ECO, CECO can automatically connect it’s test-enable pin if it is constrained in Revised design. This command allows users to specify test-enable signals to drive the new ICGs. The specified signal must be an existing signal and constrained in the Golden design. A conflicting signal specification with the same module or instance is not allowed.

Note that the specified signal must be a G1 signal or the command will error out.

Tcl Command

```
set_eco_icg_testenable
```

Parameters

`-NET`	Specifies the net defined in the specified scope.
`-PORT`	Specifies the module port defined in the specified scope.
`-DRIVER_PIN <name>`	Specifies the instance pin. The name can be either a relative path or an absolute path to the specified scope.
`-MODULE_INST <path_name> | -MODULE <module_name>`	Is used to specify the effective scope. It includes all sub-modules under the instance-path or module. -module_inst accepts a hierarchical path. The default is root. -module specifies a module.
`-INVERT`	Adds an inversion in the path between specified test-enable control to new ICG’s test-enable pin.

Related Commands

ANALYZE ECO

REPORT ECO ICG_TESTENABLE

REPORT ECO CHANGES

SET ECO CHECK

```
SET ECo Check
     <check_name*...>
     [-Warning | -Error [-CONTinue] | -Note | -Ignore]
     (Setup Mode)
```

This command requires an ECO license.

Specifies the ECO check severity and its handling. See `set_dofile_abort` to see how Conformal ECO behaves when an error message occurs. Execute this command before `set_system_mode lec`.

Tcl Command

```
set_eco_check
```

Parameters

`<check_name*>...`	Specifies the ECO check name. This accepts wildcards. Use the command `REPORT ECO CHECK -HELP` to list all the ECO checks and the default severity levels.
`-Warning`	Changes the check severity to Warning.
`-Error [-CONTinue]`	Changes the check severity to Error. With the `-continue` option, Conformal ECO continues to run instead of erroring out.
`-Note`	Changes the check severity to Note.
`-Ignore`	Changes the check severity to Ignore. See report_eco_check to see how this severity level affects reporting.

Example

The following command sets the severity level of `ECO1.1` from warning to error.

```
set eco check ECO2.2 -error
```

Related Commands

REPORT ECO CHECK

SET DOFILE ABORT

SET ECO OPTION

```
SET ECo Option
     [-LEC | -FLAT | -HIErarchical | -TOOLbox | -HIER_FLAT | -PHYSICAL_DIFF [-
     NOSUPPORT_UNREACHABLE | -SUPPORT_UNREACHABLE]] -REPlace]
     [-CHECK_XOR_GATED_CLOCK]
     [-PATCH_CONSTANT_PROPAGATION]
     [-PRESERVE_MULTIbit_flop]
     [-PRESERVE_REPlication <ON|OFF>]
     [-RENAME_PORTNAME_Bracket | -NO_RENAME_PORTNAME_Bracket]
     [-ECOPIN_BIT_blasted]
     [-INStancenaming <format_string>]
     [-BBOXnaming <format_string>]
     [-NETnaming <format_string>]]
     [-PORTnaming <format_string>]
     [-SEQuentialnaming <format_string>]
     [-SYNTHESIS_NAMING_style <Genus|NONGenus>]
     [-BACKEND_NAMING_style <INNOvus|NONINNOvus>]
     (Setup Mode)
```

This command requires an ECO license.

Sets the intention of the subsequent dofile commands.

Tcl Command

`set_eco_option`

Parameters

`[-LEC | -FLAT | -HIERarchical | -TOOLbox | -HIER_FLAT | -PHYSICAL_DIFF [-NOSUPPORT_UNREACHABLE | -SUPPORT_UNREACHABLE]] -REPlace]`

	Using any of these command options automatically adds four renaming rules with rule name prefix "`eco_rename_high_fanout_`". For more information, use command `REPORT RENAMING RULE -pin`.
`-LEC`	Specifies that the intention is to run a normal LEC comparison. *This is the default.*

`-FLAT`

Specifies that the intention is to run the Flattened ECO flow. Setting this option executes the following commands:

```
SET FLATTEN MODEL -ECO
SET FLATTEN MODEL -ENABLE_ANALYZE_HIER_COMPARE
```

It also automatically adds the following options to the ANALYZE HIER_COMPARE command (when -ECO_AWARE is specified):

```
-CONstraints
-NOEXact_pin_match
-FUNCTION_Pin_mapping
-INPUT_OUTPUT_Pin_equivalence
-THRESHOLD 0
```

`-HIErarchical`

Specifies that the intention is to run the Hierarchical ECO flow.

It will execute the following commands:

```
SET FLATTEN MODEL -ECO
```

It also automatically adds the following options to the "`WRITE HIER_COMPARE DOFILE`" command (when `-ECO_AWARE` is specified):

```
-CONstraints
-NOEXact_pin_match
-FUNCTION_Pin_mapping
-INPUT_OUTPUT_Pin_equivalence
-BALANCED_EXTRACTIONS
```

`-TOOLbox`

This option has been obsoleted.

Specifies that the intention is to run the ECO toolbox flow. For more information on this flow, refer to the web interface article titled *Conformal ECO Toolbox.*

`-HIER_FLAT` Specify the intention is to run Hierarchical Flattened ECO flow.

With this option, it automatically adds the following options to command `WRITE HIER_COMPARE_DOFILE`

```
-BALANCED_EXTRACTIONS
-CONstraints
-NOEXact_pin_match
-FUNCTION_Pin_mapping
-INPUT_OUTPUT_Pin_equivalence
-EXTRACT_ICG
-THRESHOLD 0
```

With this option, `WRITE_HIER_COMPARE_DOFILE` only write out modules that specified by command `ADD MODULE ATTRIBUTE -FLAT_ECO_MODULE`

`-PHYSICAL_DIFF` Provide a single option set_eco_option -PHYSICAL_DIFF to easily configure PD flow setting.

`-NOSUPPORT_UNREACHABLE`

The ECO analysis does not include unreachable gate changes. *This is the default*.

`-SUPPORT_UNREACHABLE`

The ECO analysis includes unreachable gate changes.

`-REPlace` Replaces the previously defined setting.

`-PATCH_CONSTANT_PROPAGATION`

Enables constant propagation. In a post-mask ECO flow, newly added gates require spare cells to implement and map the patch. In addition to the original spare cells, Conformal ECO is also able to recycle freed gates for spare cell implementation. If any cells become constant '0' or '1' due to the ECO patch creation, this option will propagate the constants to free these gates for spare cell mapping. The number of freed gates will be displayed during the `APPLY PATCH` command.

`-CHECK_XOR_GATED_CLOCK`

Enables support for XOR clock gating.

Without this option enabled, XOR gates are treated as combinational logic and can be subject to removal.

`-PRESERVE_MULTIbit_flop`

Preserves non-ECO bits in a multi-bit flop instance. When this is enabled, only ECOed bits will be replaced by the new flop in the patch.

`-PRESERVE_REPlication`

`ON`: Preserves replicated registers and replicated ports of G1 during patch generation. This is the default.

`OFF`: The replication structure of registers and ports might not be preserved in ECO.

For more information on this feature, refer to the web interface article titled *Structure Aware Patch Generation.*

`-RENAME_PORTNAME_Bracket`

For newly created ECO pins, this option will replace open and closing brackets '[' and ']' with an underscore '_'. This is the default.

`-NO_RENAME_PORTNAME_Bracket`

Disables port name bracket renaming.

`-ECOPIN_BIT_blasted`

Use bit blasted format for added bus in the ECO pin dofile generated by `-ECOPIN_dofile <filename>`.

`-HIER_FLAT` Specify the intention is to run Hierarchical Flattened ECO flow.

With this option, it automatically adds the following options to command `WRITE HIER_COMPARE_DOFILE`

-balanced_extraction

-constraints

-noexact_pin_match

-function_pIn_mapping

-input_output_pin_equivalence

-extract_icg

-threshold 0

With this option, `WRITE_HIER_COMPARE_DOFILE` only write out modules that specified by command add module attribute -`flat_eco_module`.

`-INStancenaming <format_string>`

Specifies the instance naming format of ECO combinational cells. For example, for `eco_instance_%d`, the `%d` will be an integer that makes the instance name unique. Default is `U%d`.

`-BBOXnaming <format_string>`

Specifies the naming format of the ECO blackboxes. For example, `eco_bbox_%s`, where `%s` is the original blackbox name.

`-NETnaming <format_string>`

Specifies the net naming format of ECO nets. For example, for `eco_net_%d`, the `%d` will be an integer that makes the net name unique. Default is `N%d`.

`-PORTnaming <format_string>`

Specifies the naming format for extra ports added by the `ADD ECO PIN` command. For example, for `eco_%s`, where `%s` is the original port name. Default is %s. For the default case, special characters will be replaced by an underscore "_".

`-SEQuentialnaming <format_string>`

Specifies the instance naming format of the ECO registers and latches. For example, `eco_%s`, where `%s` is the original register name. Default is `%s`.

`-SYNTHESIS_NAMING_style <Genus|NONGenus>`
`-BACKEND_NAMING_style <INNOvus|NONINNOvus>`

These options will delete or keep the renaming rules with rule name prefix `eco_rename_high_fanout_` created by CECO.

These renaming rules are added by command `set eco option [-flat | -hierarchical]`.

When `Genus` and `INNOVus` naming style are specified, the renaming rules are deleted. If not, the renaming rules are kept. *This is the default.*

Related Commands

WRITE CORRESPONDING NETS

```
WRIte COrresponding Nets
     <netnames>
     | <expression>
     | <-NETFILE <filename>>
     [-COMbination]
     [-CPUlimit <integer>]
     [-LIMit <max_number_of_nets>]
     [-OUTFILE <filename> | -GZIPOUTFILE <filename>]
     [-REPLACE]
     [-SHOWPROgress]
     [-Golden | -Revised]
     [-VERbose]
     (LEC Mode)
```

This command requires an ECO or GXL license.

Reports the equivalent nets of each specified net on the other side of design. The nets can be specified in the command or in a given file. The report will be written out to the specified file. If no file specified, it will be written to screen.

Tcl Command

`write_corresponding_nets`

Parameters

`<netnames>`	Reports the equivalent nets of each specified net on the other side of the design.
`<expression>`	Reports the equivalent nets of the specified expression on the other side of the design. Only gate IDs are supported as operands (use `REPORT GATE` to obtain gate IDs). Supported operators include bitwise operators ("~, &, | , ^") and logical operators ("!, &&, ||").
`-NETFILE <filename>`	Specifies the file that contains the list of net names. The file can contain multiple lines, but only one net name per line.

`-COMbination`	Reports an equation with the same function as the given net. The equation will be comprised of nets from the other design (limit of 6 nets). For example: `write corresponding nets -gol -comb net_a` will report an equation with the same function as `net_a`, and the equation will be composed of nets from the Revised design. To change the maximum number of nets used in the equation, use the `-limit` option.

`-CPULIMIT <integer>`	Quits the command after the specified time limit (in seconds).
	For example:
	`write corr net net1 -cpulimit 3`
	`write corr net net2 -cpulimit 2`
	`write corr net net3`
	`write corr net -netfile file1 -cpulimit 3`
	The command will stop after 3 seconds for net1, 2 seconds on net2, and no time limit for net3. For the fourth command, the command will end after 3 seconds. All the nets specified in `file1` are treated as one entity and are searched all together.

`-LIMit <max_no_of_nets>`	Specifies the maximum number of nets to use in the equation returned by the `-combination` option. Default is 6, maximum is 16. Note: The higher the number, the longer it will take to return an equation. You also run the risk of encountering an abort.
`-OUTFILE <filename>`	Specifies the file name of the output report.
`-GZIPOUTFILE <filename>`	Specifies the file name of gzipped report.
`-REPLACE`	Replaces any existing reports.

`-SHOWPROgress`	Displays the progress percentage.
`-Golden`	Specifies that the nets listed in the netfile or specified on the command line are in the Golden design, and that this command should report the equivalent nets from the Revised design.
`-Revised`	Specifies that the nets listed in the netfile or specified on the command line are in the Revised design, and that this command should report the equivalent nets from the Golden design.

Example

This command also reports inverter equivalent nets by default. In the example below, tmp[8] in the Revised design is the inverter equivalent of n44[4] in the Golden and is reported with '-'.

```
LEC> write corresponding nets n44[4] -gol
// NET 1
+  (G) n44[4]
-  (R) tmp[8]
+  (R) N26
```

Related Commands

REPORT ECO CHANGES

WRITE ECO DEF

```
WRIte ECo DEF
     [-WORKdir <working_directory>[
     [-SYNExec <synthesis_executable>]
     [-LIBrary <libraries ...>]
     [-LEF <LEF_files ..>]
     [-DEF_IN <original_DEF_filename>]
     [-DEF_OUT <new_DEF_filename>]
     [-VERILOG_IN <original_netlist_filename>]
     [-VERILOG_OUT <new_netlist_filename>]
     [-ECOscript <change_script>]
     [-PLACEscript <place_script>]
     [-PRELIBscript <script_name>]
     [-PREPROCscript <script_name>]
     [-POSTPROCscript <script_name>]
     [-VERbose]
     [-RELATIVE2CWD]
     [-ALLOW_FC_MF]
     (Setup Mode)
```

This requires an ECO-GXL license.

Generates a consolidated placement file in design exchange format (DEF) using the DEF file from RC physical. Use this command after the `OPTIMIZE PATCH` command.

Tcl Command

`write_eco_def`

Parameters

`-WORKdir <working_directory>`

Specifies the name of the working directory.

`-SYNExec <rc executable>`

Specifies the path to the synthesis executable (Genus or RTL Compiler). If this is not specified, the software will use the Genus command in your search path.

`-LIBrary <library_file_list>`

Specifies the name of the library files.

If the library name contains a relative path, the path should be relative to the directory declared in `-workdir`.

`-LEF <filename> ...`

Specifies the LEF file. This calls RC physical synthesis to perform location-aware spare-gate mapping.

`-DEF_IN <original_DEF_filename>`

Specifies the DEF file of the original netlist (before Conformal ECO optimization).

DEF is a file format that can describe the logical and physical layout of design. It represents the netlist and circuit layout.

`-DEF_OUT <new_DEF_filename>`

Specifies the filename of the output DEF file (after Conformal ECO optimization).

`-VERILOG_IN <original_netlist_filename>`

Specifies the Verilog netlist file (before Conformal ECO optimization).

`-VERILOG_OUT <new_netlist_filename>`

Specifies the filename of the new Verilog netlist (after Conformal ECO optimization).

`-ECOscript <change_script>`

Specify the script, generated by the `REPORT ECO CHANGES -script` command, to be used with Genus and other tools.

`-PLACEscript <place_script>`

Specifies the name of the placement script to input. The placement script is generated by the `OPTIMIZE PATCH -placescript` command (this script will be used by Genus and other tools).

`-PRELIBscript <script_name>`

Specifies the script to run before all settings.

`-PREPROCscript <script_name>`

Specifies the script to run before the ECO change script (`-ECOscript <`*`change_script`*`>`).

`-POSTPROCscript <script_name>`	
	Specifies the script to run after the ECO change script (`-ECOscript <`*`change_script`*`>`).
`-VERbose`	Outputs all Genus messages. By default, the command only outputs error messages.
`-RELATIVE2CWD`	Specifies that the paths provided in other options relate to the current working directory: not the Genus working directory.
`-ALLOW_FC_MF`	Allows filler cells and metal fills in the design.
	Filler cells and metal fills are not supported by default because they generally can cause problems with the placer. When you use this option, the tool will not flag these objects and this can cause problems when you try to use the DEF files in RC physical.
	The recommended solution is to remove the filler cells and metal fills from your DEF file before running Conformal ECO.

Example

The following illustrates how to create a DEF file using an input LEF and Verilog file:

```
setup> optimize patch -workdir WORK -library ../typical.lib -def ../g1.def \
-LEF ../typical.lef -sdc ../g1.sdc -placescript place.tcl -replace
setup> report eco change -script -file script.tcl -replace
setup> write eco def -workdir WORK -library ../typical.lib -LEF ../typical.lef \
-def_in ../g1.def -def_out ../g3.def -verilog_in ../g1.v -verilog_out ../g3.v
```

Related Commands

OPTIMIZE PATCH

REPORT ECO CHANGES

WRITE ECO DESIGN

```
WRIte ECo Design
     [-OVERWrite | -NEWFILE [<filename_fmt>]
       | -DIR <directory_name> [-GZIPNEWfile]| -REVIEWonly]
     [-BACKup [<filename_fmt>]]
     [-TIMEstamp [<string>]]
     [-REPORT [<filename>]]
     [-REPlace]
     (Setup Mode)
```

This command requires an ECO license.

Writes out the ECO netlist (note: does not write out non-ECO files) and attempts to reduce the number of text differences between the original netlist file and the ECO netlist file. This will reduce the number of differences reported by the UNIX `diff` command between the two files.

Limitations

- Does not work with VHDL netlists.
- Does not work if you flatten the design during the ECO process.
- Does not work if you uniquify the design during the ECO process.

Tcl Command

```
write_eco_design
```

Parameters

`-OVERWrite`	Overwrites the original files. *This is the default*.

`-NEWFILE [<filename_fmt>]`	
	Writes the design into new files. The default format for the new files is `%s.eco`, where `%s` is the original filename. If `<filename_fmt>` is not specified, the default value is `%s.eco`. The `%s` is replaced by the original file path. **Note:** If `<filename_fmt>` does not contain `%s`, the ECO design will be directly written to a specified filename without a string replacement.
`-DIR <directory_name>`	
	Specifies the directory name to where the ECO design is written.
`-GZIPNEWfile`	Write out the ECO netlist in a compressed gzip format.
`-REVIEWonly`	Reviews the changes only and does not write out the design
`-BACKup [<filename_fmt>]`	
	Creates a back up original files if using the `-OVERWrite` option. The default format for the backup files is `%s.bak`, where `%s` is the original file name.
`-TIMEstamp [<user_specified_description>]`	

This option adds a custom timestamp before any ECO action in the generated netlist. If a string is not provided, the tool adds the date at which the ECO change occurred.

```
write eco design -newfile %s.pre.G3 -replace
-timestamp ver1.1
// ECO ver1.1
//     SDFF_X2 \state_reg[0]  (.SI(n_396),
//     .SE(FE_OFN265_scan_se),


write eco design -newfile %s.pre.G3 -replace
-timestamp
// ECO 2012-11-29
//     SDFF_X2 \state_reg[0]  (.SI(n_396),
//     .SE(FE_OFN265_scan_se)
```

`-REPORT [<filename>]`

Generates a report showing the differences between the original design and the ECO design. If the filename is not provided, the report will print to the screen.

`-REPlace` Replaces the file if it already exists when using the `-NEWFILE` and `-REPORT` options.

Example

- The following is a sample script that writes out the `a.v.eco` and `b.v.eco` files after running the Conformal ECO commands:

```
read design a.v b.v -golden
// ECO process
analyze eco ...
optimize patch ...
write eco design -newfile %s.eco -replace
```

- The following script reads in the original netlist and patch, runs the `OPTIMIZE PATCH` command to map and optimize the ECO change, and writes out the ECO netlist to `top.eco.gv`.:

```
read library typical.lib -liberty
read design top.gv patch1.v
```

```
apply patch mod1 mod1_eco -keephierarchy
   optimize patch -workdir rc_work -library ../typical.lib
   write eco design -newfile top.eco.gv -replace
```

Related Commands

REPORT ECO CHANGES

WRITE PRESERVED POINTS

```
WRIte PReserved Points
     -Dofile <dofile_name>
     [-REPlace]
     (LEC Mode)
```

This generates a set of `add_probe_point` commands for collected points.

Tcl Command

```
write_preserved_points
```

Parameters

`-Dofile <dofile_name>`	Specifies the name of the dofile that writes out probed points for constrained logic.
`-REPlace`	Replaces the existing file.

Related Commands

4 Modeling Messages

This chapter lists and describes the modeling messages you encounter when the system mode changes from *Setup* to *LEC* in the Encounter® Conformal® software.

F1

Message

```
Modeled multiple-driven net(s)
```

Description

A multi-driven net has been remodeled into Boolean logic based on how the SET WIRE RESOLUTION command is set.

Example

Sample modeling message:

```
F1: Modeled multiple-driven net(s) (Occurrence: 1)
1: /n1 (and)
```

In this message:

- `/n1` is a multi-driven net
- (and) indicates that `SET WIRE RESOLUTION` is set to `AND`

The following code illustrates a circuit with a net `n1` that has multiple drivers `u0` and `u1`:

```
not u0 (n1,a);
not u1 (n1,b);
not u2 (z,n1);
```

F2

Message

```
Inserted user cut point(s)
```

Description

Conformal inserted a cut point to break a combinational loop. This operation is enabled when you use the ADD CUT POINT command.

Example

Sample modeling message:

```
F2: Inserted user cut point(s) (Occurrence: 1)
1: CUT /n1
```

In this message:

- `CUT` indicates the gate type
- `/n1` indicates the net name

You get this message when you add a cut point at net `n1` using the following command:

```
add cut point n1 -golden
```

Circuit example:

```
mux u0 (n1,n1,d,ck);
xor u1 (q,n1);
```

F3

Message

```
Inserted system cut point(s)
```

Description

A cut point has been inserted automatically to break a combinational loop.

Example

Sample modeling message:

```
F3: Inserted system cut point(s) (Occurrence: 1)
1: MUX /u0
```

In this message:

- `MUX` is the gate type
- `/u0` is the driver instance name

The following circuit example inserts CUT gate at the output of a MUX instance `u0`:

```
mux u0 (n1,n1,d,ck);
xor u1 (q,n1);
```

F3.1

Message

```
Inserted system cut point(s) for partially modeled library module pins
```

Description

A cut point has been inserted automatically to verify partially modeled library module pins.

Example

Sample modeling message:

```
F3.1: Added 2 cut points for partially modeled library module pins
in Golden (Occurrence: 1)
1: CUT /AAA4/BBB/DDD_lec_lpv_cut
```

In this message:

- `/AAA4/BBB/DDD` is the partially modeled library pin and the inserted cut point is named `/AAA4/BBB/DDD_lec_lpv_cut`

Associated Commands

```
set flatten model -library_pin_verification
```

F3.2

Message

```
Deleted redundant cut point(s)
```

Description

A redundant cut point has been deleted. Deleting the cut point does not form a combinational loop.

Example

Sample modeling message:

```
F3.2: Deleted redundant cut point (Occurrence: 1)
1: i1[1]
```

`In this message:`

- `i1[1]` is a redundant cut point and is deleted.

Associated Commands

```
analyze setup -cut
```

F3.3

Message

```
Removed cut point(s)
```

Description

A cut point has been removed.

Example

Sample modeling message:

```
F3.3: Removed cut point (Occurrence: 2)
1: i1[1]
2: i1[0]
```

In this message:

- `i1[0] and i1[1]` are cut points that have been removed.

Associated Commands

```
analyze setup -cut
```

F3.4

Message

```
Moved cut point(s) to new location
```

Description

A cut point has been moved to a new location.

Example

Sample modeling message:

```
F3.4: Moved cut point(s) to new location (Occurrence: 1)
1: MOD_B/MOD_B_A/U$6 MOD_A/U$5/U$90
```

In this message:

- A cut point is moved from the output of gate `MOD_B/MOD_B_A/U$6` to the output of gate `MOD_A/U$5/U$90`.

Associated Commands

```
analyze setup -cut
```

F3.5

Message

```
Remodeled cut point(s) to DLAT(s)
```

Description

A cut point has been remodeled to a DLAT. The DLAT's set and reset pins are tied to zero. Its clock pin is tied to one and data pin is connected to the fan-in of the cut point.

Example

Sample modeling message:

```
F3.5: Remodeled cut point(s) to DLAT(s) (Occurrence: 1)
1: DLAT hysteresis_new/g502/SMC_I0
```

In this message:

- A cut point `hysteresis_new/g502/SMC_I0` is remodeled to a DLAT.

Associated Commands

```
analyze setup -cut
```

F3.6

Message

```
Inserted balanced cut point(s)
```

Description

A cut point has been inserted automatically to balance the cut points between Golden and Revised.

Example

Sample modeling message:

```
F3.6: Added balanced cut points (Occurrence: 1)
1: NAND g7/U$1
```

In this message:

- A balanced cut point has been inserted at the output of NAND gate `g7/U$1`.

Associated Commands

```
analyze setup -cut
```

F5

Message

```
Folded DLAT(s) into DFF(s)
```

Description

D-latches (DLATs) were folded into D flip-flops (DFFs). This operation is enabled when you use the SET FLATTEN MODEL `-LATCH_FOLD` command, which specifies that two latches that are in a master-slave configuration should be converted into a single DFF gate. This operation is also affected by the `SET FLATTEN MODEL -LATCH_FOLD_MASTER` command.

Example

Sample modeling message:

```
F5: Folded DLAT(s) into DFF(s) (Occurrence: 1)
1: /l1 /l0
```

In this message, `/l1` and `/l0` are the instance names of the folded DLATs.

You would get this modeling message if you use the following command to convert two master-slave DLATs `l0` and `l1` into a DFF:

```
set flatten model -latch_fold
```

Circuit example:

```
not  u0 (ck_,ck);
DLAT l0 (n1,,1'b0,1'b0,ck_,d);
DLAT l1 (q ,,1'b0,1'b0,ck,n1);
```

F6

Message

```
Created DLAT(s) due to trireg net(s) or combinational loop(s)
```

Description

A DLAT was created due to trireg net or a series of buffers/inverters that were implemented as a bus-holder.

Example

Sample modeling message:

```
F6: Created DLAT(s) due to trireg net(s) or combinational loop(s) (Occurrence: 1)
1: DLAT /n1 due to trireg net
```

This message indicates that a `DLAT` was created on net `n1` for `trireg`, which is illustrated in the following circuit example:

```
trireg n1;

bufif0 u0 (n1,a,s0);
bufif0 u1 (n1,b,s1);
buf    u2 (z,n1);
```

F7

Message

```
Set DLAT data port(s) as ZERO due to disabled clock port(s)
```

Description

DLAT data ports were set to zero because there were disabled clock ports. This operation is enabled when you use the SET FLATTEN MODEL `-LATCH_FOLD` command.

Example

Sample modeling message:

```
F7: Set DLAT data port(s) as ZERO due to disabled clock port(s) (Occurrence: 1)
1: Set DLAT 'q_reg' data port '/d' to ZERO due to disabled clock port
```

This message indicates that data port `d` of register `q_reg` is tied to logic `ZERO` because clock port of register `q_reg` is always disabled (logic `ZERO`).

You would get this message if you issue the following command:

```
set flatten model -latch_fold
```

Circuit example:

```
and u0 (g,1'b0,ck);

always @(d or g)
begin
    if (g)
        q <= d;
    end
```

F8

Message

```
Converted DLAT(s) to BUF(s) due to transparency
```

Description

DLATs were converted into buffers because of transparency. This operation is enabled when you use the SET FLATTEN MODEL `-LATCH_TRANSPARENT` command, which specifies that DLATs should be converted into buffers if the DLAT clock ports are always enabled.

Example

Sample modeling message:

```
F8: Converted DLAT(s) to BUF(s) due to transparency (Occurrence: 1)
1: DLAT /q_reg
```

This message indicates that a buffer was converted `buf (q,d)`, because the clock of register `q_reg` is always enabled (logic ONE).

You would get this message if you issue the following command:

```
set flatten model -latch_transparent
```

Circuit example:

```
or u0 (g,1'b1,ck);
always @(d or g)
begin
    if (g)
    q <= d;
end
```

F8.1

Message

```
Converted DLAT(s) to BUF(s) due to redundancy to DFF(s)
```

Description

DLAT is converted to BUF if its frontier contains only DFF(s) and DLAT(s) and it is redundant to all of them.

Example

F8.1: Converted DLAT(s) to BUF(s) due to redundancy to DFF(s) (Occurrence: 1)
1: /q0_dlat

This message indicates that DLAT q0_dlat was converted to BUF because it was redundant to its fanout DFF(s)/DLAT(s).

F9

Message

```
Removed DLAT data dependency due to redundancy
```

Description

Indicates that DLAT data dependencies were removed because of redundancy. This operation is enabled when you use the `SET FLATTEN MODEL -SEQ_REDUNDANT` command, which specifies that redundant fanout gates should be removed from DFFs and DLATs.

Associated Commands

```
SET FLATTEN MODEL -SEQ_REDUNDANT
```

F10

Message

```
Removed redundant AND/NAND/OR/NOR fanin gate(s) for DFF/DLAT(s)
```

Description

Redundant `AND`/`NAND`/`OR`/`NOR` fan-in gates for DFFs and DLATs were removed. This operation is enabled when you use the SET FLATTEN MODEL `-SEQ_REDUNDANT` command.

Example

Sample modeling message:

```
F10: Removed redundant AND/NAND/OR/NOR fanin gate(s) for DFF/DLAT(s)
  (Occurrence: 1)
1: Removed connection from INV /u0 to AND /u1 for DFF /f0_reg
```

This message indicates that the fan-in `AND` `u1` and `INV` `u0` gates were optimized for `DFF` `f0_reg` due to redundant logic.

You would get this command if you issue the following command:

```
set flatten model -seq_redundant
```

Circuit example:

```
not u0 (rst_,rst);
and u1 (sr,set,rst_);
DFF f0_reg(q,,sr,rst,ck,d);
```

F11

Message

```
Removed redundant AND/NAND/OR/NOR fanout gate(s) for DFF/DLAT(s)
```

Description

Redundant `AND`/`NAND`/`OR`/`NOR` fan-out gates for DFFs and DLATs were removed. This operation is enabled when you use the SET FLATTEN MODEL `-SEQ_REDUNDANT` command.

Example

Sample modeling message:

```
F11: Removed redundant AND/NAND/OR/NOR fanout gate(s) for DFF/DLAT(s) (Occurrence:
1)
1: Remodeled AND /u0 for DFF /n1_reg
```

This message indicates that the fan-out `AND` gate `u0` was optimized for `DFF` `n1_reg` due to redundant logic.

You would get this command if you issue the following command:

```
set flatten model -seq_redundant
```

Circuit example:

```
always @(posedge ck or negedge rst )
  begin
       if (!rst)
          n1 <= 0;
       else
          n1 <= d;
    end
    and u0 (q,n1,rst);
```

F12

Message

```
Converted DFF(s) to DLAT(s) due to disabled clock port(s)
```

Description

DFFs were converted into DLATs due to disabled clock ports. This operation is enabled when you use the SET FLATTEN MODEL `-DFF_TO_DLAT_ZERO`, which converts a DFF to a DLAT when the clock port is zero.

Example

Sample modeling message:

```
F12: Converted DFF(s) to DLAT(s) due to disabled clock port(s) (Occurrence: 1)
1: DLAT /q_reg
```

This message indicates DFF `q_reg` was converted to a DLAT because the clock of the register is always disabled (logic ZERO).

You would get this message with the following command:

```
set flatten model -DFF_TO_DLAT_ZERO
```

Circuit example:

```
and u0 (ck1,1'b0,ck);
      always @(posedge ck1)
      begin
            q <= d;
      end
```

Note: The `-DFF_TO_DLAT_ZERO` option is enabled by default.

F13

Message

```
Converted DFF(s) to DLAT(s) due to direct feedback
```

Description

DFFs were converted to DLATs due to direct feedback. This operation is enabled when you use the SET FLATTEN MODEL `-DFF_TO_DLAT_FEEDBACK` command, which converts a DFF to a DLAT if the DFF's output feeds back directly to the DFF's input.

Example

Sample modeling message:

```
F13: Converted DFF(s) to DLAT(s) due to direct feedback
(Occurrence: 1)
1: DLAT /q_reg
```

This message indicates that DFF `q_reg` was converted to a DLAT because of direct feedback from output `q` of the register to its input `d`.

You would get this message if you issue the following command:

```
set flatten model -DFF_TO_DLAT_FEEDBACK
```

Circuit example:

```
always @(posedge ck)
   begin
     q <= q;
   end
```

Note: The `-DFF_TO_DLAT_FEEDBACK` option is enabled by default.

F14

Message

```
Remodeled gated-clock DFF(s) or DLAT(s) to mux-feedback
```

Description

Gated-clock logic for DFFs or DLATs were remodeled to MUX-feedback. This operation is enabled when you run the SET FLATTEN MODEL `-GATED_CLOCK` command, which remodels gated-clock logic of the clock port of a DFF.

Example

Sample modeling message:

```
F14: Remodeled gated-clock DFF(s) or DLAT(s) to mux-feedback (Occurrence: 1)
1: /q_reg (DFF)
```

This message indicates that the de-glitch gating clock DLAT `l0` and enable logic `u1` for register DFF `q_reg` was converted to a mux-feedback DFF.

You would get this message when issue the following command:

```
set flatten model -GATED_CLOCK
```

Circuit example:

```
not u0 (ck_,ck);
DLAT l0 (en,,1'b0,1'b0,ck_,ena);
and u1 (ck1,ck,en);
always @(posedge ck1)
begin
   q <= d;
end
```

F14.1

Message

```
Remodeled gated-clock DFF(s) or DLAT(s) without latch to mux-feedback
```

Description

Conformal remodeled gated-clock logic for DFFs or DLATs without deglitching to MUX-feedback. This operation is enabled when you use the SET FLATTEN MODEL `-GATED_CLOCK`, which remodels gated-clock logic of the clock port of a DFF. This operation might need the ADD CLOCK command to define the clock pin.

Example

Sample modeling message:

```
F14.1: Remodeled gated-clock DFF(s) or DLAT(s) without latch to mux-
feedback(Occurrence: 1)
1: /q_reg (DFF)
```

This message indicates that the gated-clocking enable logic `u1` for register DFF `q_reg` was converted to a mux-feedback DFF.

You get this message when you issue the following commands:

```
add clock 0 ck
set flatten model -GATED_CLOCK
```

Circuit example:

```
and u0 (ck1,ck,ena);
always @(posedge ck1)
begin
  q <= d;
end
```

F16

Message

```
Converted DLAT(s) to MUX(s) due to clock inversion relation
```

Description

DLATs were converted into MUXes due to clock inversion relationships.

Example

Sample modeling message:

```
F16: Converted DLAT(s) to MUX(s) due to clock inversion relation (Occurrence: 1)
1: /l0
```

This message indicates that a dual-port DLAT `l0` was converted to a MUX - `mux l0 (q,d0,d1,ck)` due to the clock inversion relationship on the dual-port DLAT clock ports.

Circuit example:

```
not  u0 (ck_,ck);
DLAT l0 (q,,1'b0,1'b0,ck_,d0,ck,d1);
```

F17

Message

```
Converted DLAT(s) to BUF(s)/INV(s) due to set/reset inversion relation
```

Description

DLATs were converted to buffers/inverters due to set/reset inversion relationships.

Example

Sample modeling message:

```
F17: Converted DLAT(s) to BUF(s) due to set/reset inversion relation
(Occurrence: 1)
1: /l0
```

This message indicates that a DLAT `l0` was converted to a buffer, `buf l0 (q,a)`, because of set/reset inversion relationships on asynchronous set/reset of DLAT.

Circuit example:

```
not  u0 (a_,a);
DLAT l0 (q,,a,a_,ck,1'b0);
```

F18

Message

```
Converted DFF/DLAT(s) to ZERO/ONE
```

Description

Conformal converted a DFF or a DLATs to a ZERO or ONE gate. This operation is enabled when you use the SET FLATTEN MODEL `-SEQ_CONSTANT` command, which converts a DFF or DLAT to a ONE or ZERO gate if the data port is a one or zero. This operation is also affected by the SET FLATTEN MODEL `-SEQ_CONSTANT_FEEDBACK` command.

Example

Sample modeling message:

```
F18: Converted DFF/DLAT(s) to ZERO/ONE (Occurrence: 1)
1: DFF /n1_reg (ZERO)
```

This message indicates that DFF `n1_reg` was converted to logic ZERO because data port is tied to logic ZERO.

You would get this message if you issue the following command:

```
set flatten model -seq_constant
```

Circuit example:

```
always @ (posedge ck)
   n1 <= 1'b0;
   assign q = a | n1;
```

F19

Message

```
Merged DFF(s) or DLAT(s) in clock cones due to functional equivalence
```

Description

Sequentially equivalent DFFs or DLATs in clock cones were merged. This operation is enabled when you use the SET FLATTEN MODEL `-SEQ_MERGE` command.

Example

Sample modeling message:

```
F19: Merged DFF(s) or DLAT(s) in clock cones (Occurrence: 1)
1: DFF /ck0_reg /ck1_reg
```

This message indicates that DFF `ck0_reg` and DFF `ck_reg` were merged into a single DFF because the two DFFs were sequentially equivalent.

You would get this message when you issue the following command:

```
set flatten model -seq_merge
```

Circuit example:

```
always @ (posedge ck)
begin
   ck0 <= a;
   ck1 <= a;
end

always @ (posedge ck0)
  q0 <= d;
always @ (posedge ck1)
  q1 <= d;
```

F19.1

Message

```
Merged DFF(s) or DLAT(s) in set/reset cones due to functional equivalence
```

Description

Sequentially equivalent DFFs or DLATs in set/reset cones were merged. This operation is enabled when you use the `SET FLATTEN MODEL -SEQ_MERGE` command

Example

Sample modeling message:

```
(F19.1) Merged 2 DFF/DLAT(s) in set/reset cones due to functional equivalence
```

This message indicates that the two DFFs or DLATs were merged because they were functionally equivalent.

You would get this message when you issue the following command:

```
set flatten model -seq_merge
```

Circuit example:

```
always @ (posedge clk)
    begin
        set0 <= a;
        set1 <= a;
    end

always @ (posedge clk or posedge set0)
    begin
        if (set0)
            q0 <= 1'b1;
        else
            q0 <= d0;
        end
always @ (posedge clk or posedge set1)
    begin
        if (set1)
            q1 <= 1'b1;
        else
            q1 <= d1;
end
```

F20

Message

```
Merged DFF(s) or DLAT(s)
```

Description

Sequentially equivalent DFFs and DLATs were merged. This operation is enabled when you use the SET FLATTEN MODEL `-ALL_SEQ_MERGE` command, which merges common groups of sequential elements into one sequential element in a logic cone of a key point.

Example

Sample modeling message:

```
F20: Merged DFF(s) or DLAT(s) (Occurrence: 1)
1: DFF /q0_reg /q1_reg
```

This message indicates that DFF `q0_reg` and DFF `q_reg` were merged into a single DFF because the two DFFs are sequentially equivalent.

You would get this message if you use the following command:

```
set flatten model -all_seq_merge
```

Circuit example:

```
always @ (posedge ck)
  begin
   q0 <= d;
   q1 <= d;
  end
```

F21

Message

```
Merged DFF(s) or DLAT(s) defined by user
```

Description

User-defined DFFs or DLATs were merged using the ADD INSTANCE EQUIVALENCES or REMODEL `-instance_eq` command.

Example

Sample modeling message:

```
F21: Merged DFF(s) or DLAT(s) defined by user (Occurrence: 1)
1: DFF /q0_reg /q1_reg
```

This message indicates that you merged two DFFs `q0_reg` and DFF `q1_reg` into a single DFF using the following command:

```
add instance equivalences /q0_reg /q1_reg -golden
```

Circuit example:

```
always @ (posedge ck)
  begin
    q0 <= d;
    q1 <= d;
end
```

F23

Message

```
Merged DFF(s) or DLAT(s) multiple ports into single port due to equivalence
```

Description

Multiple ports belonging to DFFs or DLATs into one single port due to equivalence were merged.

Example

Sample modeling message:

```
F23: Merged DFF(s) or DLAT(s) multiple ports into single port due to equivalence
(Occurrence: 1)
1: DFF /f0
```

This message indicates that a dual-port DFF `f0_reg` with equivalent data and clock port was merged into a single port DFF - DFF `f0_reg (q,,1'b0,1'b0,ck,d)`.

Circuit example:

```
buf n0 (d0,d);
buf n1 (d1,d);
DFF f0_reg (q,,1'b0,1'b0,ck,d0,ck,d1);
```

F25

Message

```
Pipeline-retimed DFF(s) to outputs
```

Description

DFFs to outputs were pipeline-retimed. This operation is enabled when you use the ADD MODULE ATTRIBUTE `-PIPELINE_RETIME` command, which checks specified modules for pipeline retiming and remodels when necessary.

Example

Sample modeling messages:

```
Report modeling message for Revised
F25: Pipeline-retimed DFF(s) to outputs (Occurrence: 1)
  1: Pipeline retimed DFF 'q_reg' to output
Report modeling message for Revised
F25: Pipeline-retimed DFF(s) to outputs (Occurrence: 2)
  1: Pipeline retimed DFF 'a1_reg' to output
  2: Pipeline retimed DFF 'b1_reg' to output
```

These messages indicate that the DFF `a1_reg` and DFF `b1_reg` performed pipeline-retiming to the outputs.

You would get these messages if you issue the following command:

```
add module attribute ckt -pipeline_retime
```

Circuit example:

Golden:

```
assign n1 = a & b;

always @(posedge ck)
   q <= n1;
```

Revised:

```
always @(posedge ck)
begin
  a1 <= a;
  b1 <= b;
end
assign q = a1 & b1;
```

F26

Message

```
Merged dual-port DLAT(s) into single port DLAT(s)
```

Description

Dual-port DLATs were merged into single-port DLATs. This operation is enabled when you use the SET FLATTEN MODEL `-LATCH_MERGE_PORT` command.

Example

Sample modeling message:

```
F26: Merged dual-port DLAT(s) into single port DLAT(s)
(Occurrence: 1)
1: Merged multi-port DLAT '/l0' into single port DLAT
```

This message indicates that dual-port DLAT "`l0`" was merged into a single port DLAT using the following command:

```
set flatten model -latch_merge_port
```

Circuit example:

```
DLAT l0 (q,,1'b0,1'b0,ck,d0,sck,d1);
```

```
Result :

     and u0 (n1,d0,ck);
     and u1 (n2,d1,sck);
     or  u2 (din,n1,n2);
     or  u3 (ck_or,ck,sck);
     DLAT l0 (q,,1'b0,1'b0,ck_or,din);
```

F27

Message

```
Converted internal input port(s) to inout port(s)
```

Description

Internal input ports were converted into inout ports because the input port does not drive any instances.

Example

Sample modeling message:

```
F27: Converted internal input port(s) to inout port(s)
(Occurrence: 1)
1: port /u0/y in module AN2
```

This message indicates that port `y` of module `AN2` does not drive any load.

Circuit example:

```
module AN2 (y,z,a,b);
input a,b,y;
output z;

and  u0 (z,a,b);
buf  u1 (y,a);

endmodule

module ckt (y,z,a,b);
input a,b;
output y,z;

AN2 u0 (y,z,a,b);

endmodule
```

F28

Message

```
Converted internal output port(s) to inout port(s)
```

Description

Internal output ports were converted to inout ports because the output port is not driven.

Example

Sample modeling message:

```
F28: Converted internal output port(s) to inout port(s)
(Occurrence: 1)
1: port /u0/a in module AN3 due to high-impedance (Z) gates
```

This message indicates that port `a` of module `AN3` is not driven by any driver.

Circuit example:

```
module AN3 (z,a,b,c);
input b,c;
output z,a;

and u0 (z,a,b,c);

endmodule

module ckt (z,a,b,c);
input a,b,c;
output z;

AN3 u0 (z,a,b,c);

endmodule
```

F30

Message

```
Ignored weak device(s) due to the existence of strong device(s)
```

Description

Any weak devices were ignored due to the existence of a stronger device in a multiple-driven net.

Example

Sample modeling message:

```
F30: Ignored weak device(s) due to the existence of strong device(s) (Occurrence: 1)
1: n1
```

This message indicates that a weak device buffer `u0` driving net `n1` was ignored due to the existence of stronger buffer device `u1` driving the same net `n1`.

Circuit example:

```
buf (weak0,weak1)u0 (n1,a);
buf    u1 (n1,b);
buf    u2 (z,n1);
```

F32

Message

```
Created Z gate(s) for floating net(s) and floating pin(s)
```

Description

Z gates were created for floating nets and floating pins.

Example

Sample modeling message:

```
F32: Created Z gate(s) for floating net(s) and floating pin(s) (Occurrence: 1)
1: c
```

This message indicates that a net `c` of instance `u0` is not driven.

Circuit example:

```
module ckt (z,a,b);
input a,b;
output z;
wire c;

and u0 (z,a,b,c);

endmodule
```

F34

Message

```
Convert X assignment(s) as don't care(s)
```

Description

X assignments were converted to "don't cares". This operation is enabled when you use the SET X CONVERSION command.

Example

Sample modeling message:

```
F34: Convert X assignment(s) as don't care(s) (Occurrence: 1)
1: Converted 'X assignment' at 'N$1' be don't care
```

This message indicates that assignment in RTL code `default: z = 1'bx` is converted to `don't cares` through the following command:

```
set x conversion DC -golden
```

Note: This option in the Golden design is enabled by default.

Circuit example:

```
always @(a or b or sel)
    begin
     case (sel)
       2'b01   : z = a;
       2'b10   : z = b;
       default : z = 1'bx;
     endcase
    end
```

Note: The following logic behavior describes a don't care condition:

```
if (s == 1) out = 1'bx;
if (s == 0) out = in;
```

F34.1

Message

```
Convert X assignment(s) as zero(s)
```

Description

X assignments were converted to zero. This operation is enabled when you use the SET X CONVERSION command.

Example

Sample modeling message:

```
F34.1: Convert X assignment(s) as zero(s) (Occurrence: 1)
1: Converted X assignment 'N$1' as 0
```

This message indicates that X assignment in RTL code `default: z = 1'bx` was converted to logic ZERO using the following command:

```
set x conversion 0
```

Circuit example:

```
always @(a or b or sel)
begin
  case (sel)
    2'b01   : z = a;
    2'b10   : z = b;
    default : z = 1'bx;
  endcase
end
```

F34.2

Message

```
Convert X assignment(s) as one(s)
```

Description

X assignments were converted to one. This operation is enabled when you use the SET X CONVERSION command.

Example

Sample modeling message:

```
F34.2: Convert X assignment(s) as one(s) (Occurrence: 1)
1: Converted X assignment 'N$1' as 1
```

This message indicates that an X assignment in the RTL code `default: z = 1'bx` was converted to logic ONE using the following command:

```
set x conversion 1
```

Circuit example:

```
always @(a or b or sel)
   begin
     case (sel)
     2'b01   : z = a;
     2'b10   : z = b;
     default : z = 1'bx;
   endcase
end
```

F34.3

Message

```
Converted 1 X assignment(s) as E(s)
```

Description

An `X` assignment was converted to an error (E) gate . This operation is enabled when you use the SET X CONVERSION command. If the `X` assignment space of the Revised design is within the `X` assignment space of the Golden design, then the `E` gate is marked as an extra unmapped point (redundant gate).

Example

Sample modeling message:

```
F34.2: Convert 1 X assignment(s) as E (Occurrence: 1)
1: Converted X assignment 'N$1' as E
```

This message indicates that an X assignment in the RTL code `default: z = 1'bx` was converted to logic E using the following command:

```
set x conversion E
```

Circuit example:

```
always @(a or b or sel)
   begin
     case (sel)
     2'b01   : z = a;
     2'b10   : z = b;
     default : z = 1'bx;
   endcase
end
```

F36

Message

```
Don't care(s) added due to $constraint(s)
```

Description

"don't cares" were added due to `$constraint`s.

Example

Sample modeling message:

```
F36: Don't care(s) added due to $constraint(s) (Occurrence: 1)
1: u0: DFF q_reg
```

This message indicates that constraint `cstr_0` for register `q_reg` is added to don't cares.

Circuit example:

```
  always @(a or b or sel)
  begin
     case (sel)
      2'b01 : z = a;
      2'b10 : z = b;
      default : z = 1'bx;
     endcase
  end
  $constraint cstr_0 ($one_hot (sel));

   always @(posedge ck)
     q <= z;
```

F36.1

Message

```
Detected always-on DC due to $constraint(s)
```

Description

Detected always-on DC gates due to $constraint(s). An always-on DC gate means that its output is always dont-care, and that indicates the condition in $constraint(s) is always falsified. Comparing the dont-care space to anything is equivalent. Therefore, it is abnormal that a design has always-on DC. LEC can conclude equivalence, even for a bad implementation. Do review the constraints to ensure correctness.

Example

Sample modeling message:

```
F36.1: Detected always-on DC(s) due to $constraint(s) (Occurrence: 1)
1: DC /g
```

In this message, the control C-pin of the DC gate "g" is always on.

F39

Message

```
Added output Z gate(s)
```

Description

Output Z gates were added. This operation is enabled by SET FLATTEN MODEL `-OUTPUT_Z`, which is enabled by default.

Example

Sample modeling message:

```
F39: Added output Z gate(s) (Occurrence: 1)
1: /z
```

This message indicates that a Z gate was added at the output `z` using the following command:

```
set flatten model -output_z
```

Circuit example:

```
module ckt (z,a,sel);
input a,sel;
output z;

bufif0 u0 (z,a,sel);

endmodule
```

F41

Message

```
Converted set/reset loop(s) to data-hold(s)
```

Description

The Conformal software converted a data-hold function that is modeled using an asynchronous set and an asynchronous reset functions, to a mux data-hold function

Example

Sample modeling message:

```
F41: Converted set/reset loop(s) to data-hold(s) (Occurrence: 1)
1: DFF 'Q_reg'
```

Circuit example:

```
assign RN =  !Q & EN;
assign SET =  Q & EN;

always @(posedge CK or posedge RN or posedge SET)
begin
  if (RN)
    Q <= 1'b0;
  else if (SET)
    Q <= 1'b1;
  else
    Q <= D;
end
```

F42

Message

```
Unfolded DFF to latches
```

Description

A DFF was unfolded into two DLATs. This operation is enabled when you use the REMODEL `-UNFOLD_DFF` command, which specifies that a DFF should be converted into two DLATs that are in a master-slave configuration.

Example

Sample modeling message:

```
F42: Unfolded DFF to latches (Occurrence: 1)
1: /q_reg
```

This message indicates you converted a single DFF into a master-slave DLATs using the following command:

```
remodel -unfold_dff
```

Circuit example:

```
always @(posedge ck)
  q <= d;
```

F43

Message

```
Added DLATs to cut loops
```

Description

DLATs were added to cut combination loops.This operation is enabled by the SET FLATTEN MODEL command.

Example

Sample modeling message:

```
F43: Added DLATs to cut loops (Occurrence: 1)
1: /u0
```

This message indicates that a DLAT was added at instance driver `u0` to cut the combinational loop using the following command:

```
set flatten model -loop_as_dlat
```

Circuit example:

```
mux u0 (n1,n1,d,ck);
xor u1 (q,n1);
```

F44

Message

```
Converted Z(s) to ZERO/ONE/don't care(s)
```

Description

One or more Z outputs were converted to ZERO, ONE, or don't care.

F45

Message

```
Adjusted DFF/DLAT of positive library cell(s)
```

Description

One or more positive library cells have a DFF or DLAT that was adjusted.

F46

Message

```
Converted DFF/DLAT(s) to BUF(s) by user
```

Description

The user-specified DFF/DLATs were converted to buffer gates. The Conformal software performs such conversion without checking whether it preserves the functionality of the netlist or not. Therefore, this operation could modify the functionality of the netlist.

Additionally, if converting a specific DFF/DLAT to buffer creates combinational loop in the netlist, the software will not convert such DFF/DLAT.

This operation is enabled by the REMODEL `-seq2buffer` command.

Example

Circuit example:

```
not u0 (a_, a)
DLAT l0 (q, ,a, a_, ck, 1'b0)
```

F47

Message

```
Inserted DC(s) due to DFF clock/data interaction
```

Description

Clock or data interaction with a DFF caused one or more DCs to be inserted.

F49

Message

```
Removed redundant DLAT(s)
```

Description

Collapses serial D-latches (even when there is logic between them) into the last latch on that clock phase. You cannot use this option with latches that have set or reset pins

This operation is enabled by the REMODEL `-red_dlat` command.

Example

F49: Removed redundant DLAT(s) (Occurrence: 1)

```
1: /q0_reg
```

The following message indicates that the DLAT `/q0_reg` is removed from netlist. `/q0_reg` is redundant with existence of `/q1_reg`:

Circuit example:

```
always @ (clk or d)
if (clk)
q0 = d;
always @ (clk or q0)
if (clk)
q1 = q0;
```

F50

Message

```
False paths(s) broken
```

Description

One or more false paths are broken.

F51

Message

```
Merged BBOXes due to common inputs
```

Description

Blackboxes have common inputs and are merged.

F52

Message

```
Removed redundant MUX fanout for DLAT(s)
```

Description

Redundant MUX fanouts for one or more DLATs were removed.

F53

Message

```
Constraints added from OVL assertion(s)
```

Description

One or more OVL assertion constraints were added.

F54

Message

```
Extracted MUX(s) from Data port of DFF(s)
```

Description

One or more MUX from one or more DFF data ports were extracted.

F55

Message

```
Disabled the sequential modeling of DFF/D-LATCH that only fan-outs to constraints
```

Description

One or more DFF/D-LATCH were excluded from sequential modeling because they only fan-out to constraints.

F56

Message

```
Ignored power and ground pins in library module
```

Description

The power and ground pins of the library module will not be compared.

Example

Sample modeling message:

```
F56: Ignored 4 power and ground pins in library module PROBE
(Occurrence: 4)

  1: PROBE: PSUB

  2: PROBE: NSUB

  3: PROBE: VSS

  4: PROBE: VDD
```

You would get this message when verifying a physical netlist. It indicates the power and ground pins defined in the LEF file will not be compared.

F57

Message

```
BBOX inout-pin(s) modeled as input(s) based on LEF pin type
```

Description

INOUT power pins in LEF definition are modeled as inputs.

Example

Sample modeling message:

```
F57: BBOX input-pin(s) modeled as input(s) based on LEF pin type
(Occurrence: 3)

  1: PROBE: VDD

  2: PROBE: NSUB

  3: PROBE: PSUB
```

You would get this message when verifying a physical netlist. It indicates the power pins defined as INOUT in the LEF file are modeled as inputs to avoid false nonequivalence.

F59

Message

```
Convert set/reset functions of DFF/DLAT(s) to D/CLK due to disabled clock port(s)
```

Description

Set/reset functions of DFFs or DLATs with disabled clock ports are converted to data/clock, where set/reset ports are tied zero and DFFs are converted into DLATs. This operation is enabled by default. You can use `set flatten model -nolatch_sr_to_d` to turn it off.

F65.1

Message

```
Tied undriven signals to zero
```

Description

An undriven signal is tied to zero. The `SET UNDRIVEN SIGNAL COMMAND` sets all undriven signals in the design.

Example

Sample modeling message:

F65.1: Tied undriven signals to zero (Occurrence: 1)

1: /w1

In this message, signal /w1 is tied to zero.

F65.2

Message

Tied undriven signals to one

Description

An undriven signal is tied to one. The `SET UNDRIVEN SIGNAL COMMAND` sets all undriven signals in the design.

Example

Sample modeling message:

F65.2: Tied undriven signals to one (Occurrence: 1)

1: /w1

In this message, signal /w1 is tied to one.

F65.3

Message

```
Tied undriven signals to X
```

Description

An undriven signal is tied to X (don't care). The `SET UNDRIVEN SIGNAL COMMAND` sets all undriven signals in the design.

Example

Sample modeling message:

F65.2: Tied undriven signals to X (Occurrence: 1)
1: /w1

In this message, signal /w1 is tied to X (don't care).

F68

Message

```
Model Liberty contention condition as X
```

Description

Model contention condition as X to the output of the library cell. For output 'o' and contention condition 'a', the modeled output would be a ? 1'bx : o.

F69

Message

```
Disabled clock for inactive port
```

Description

The clock of a DFF/DLAT is disabled (tied to zero) if the register's data cone is a self feedback.

Example

Sample modeling message:

F69: Disabled clock for inactive port (Occurrence: 1)
1: DFF /q_reg

In this message, the clock port of register '`q_reg`' was tied to zero.

F73

Message

```
Transform set-dominant DFF/DLAT(s)
```

Description

Indicates that during auto setup the tool has analyzed and transformed unbalanced set-dominant DFF/DLAT structures.

Example

Sample modeling message:

```
F73: Transform set-dominant DFF/DLAT(s) (Occurrence: 1)
  1: Remodeled OR /dff1/U$3 for DFF /dff1/dff_sub
```

5

Tcl Command Entry Mode Support

Describes the Tcl commands available in Conformal. Commands are presented in alphabetical order.

- get_gate_id on page 1078
- get_gate_type on page 1079
- get_handle_type on page 1080
- get_instances on page 1081
- get_keypoint on page 1083
- get_license_mode on page 1084
- get_map_points on page 1085
- get_module_definition on page 1086
- get_module_instances on page 1087
- get_names on page 1088
- get_nets on page 1089
- get_parent on page 1091
- get_pins on page 1092
- get_ports on page 1094
- get_primitive_type on page 1096
- get_project_name on page 1097
- get_property on page 1098
- get_relative_path on page 1100
- get_root_module on page 1101
- get_top_module on page 1102
- get_unmap_points on page 1103
- get_version_info on page 1104
- help on page 1105
- index_collection on page 1106
- ncdecrypt on page 1107
- objtype on page 1108
- query_collection on page 1109

add_to_collection

```
add_to_collection <base_collection>
     <collection | object_list >
     [-unique]
```
(TCL_SETUP/TCL_LEC mode)

Adds objects to a collection, resulting in a new collection.

Parameters

`<base_collection>`	Specifies the base collection of objects to which objects will be added. This object will be copied, resulting in a new collection, and the specified objects will be added to the new collection.
`<collection | object_list>`	
	Specifies a collection or object list to add to the new collection.
`-unique`	Removes duplicate objects from the new collection.

Example

```
TCL_LEC> set a [find_cfm -instance * -collection]
collection_0
TCL_LEC> set b [find_cfm -instance u* -collection]
collection_1
TCL_SETUP> set c [add_to_collection $a $b]
collection_2
TCL_SETUP> query_collection $c
pmu u_gsw u1 u_gsw u1
TCL_SETUP> set c [add_to_collection $a $b -unique]
collection_3
TCL_SETUP> query_collection $c
pmu u_gsw u1
```

append_to_collection

```
append_to_collection <var>
     <collection | object_list>
     [-unique]
     (TCL_SETUP/TCL_LEC mode)
```

Appends objects to a collection.

Parameters

`<var>`	Specifies a variable name. The objects matching the specified collection or object list will be added to the collection referenced by this variable.
`<collection | object_list>`	Specifies a collection or list of objects to add.
`-unique`	Remove duplicate objects from the resulting collection. By default, duplicate objects are not removed.

Example

```
TCL_SETUP> set b [find -instance u* -collection]
collection_1
TCL_SETUP> append_to_collection z $b
TCL_SETUP> query_collection $z
u_gsw u1
```

cfm_is_gui_mode

```
cfm_is_gui_mode
```
(TCL_SETUP/TCL_LEC mode)

Checks whether the tool is in command line mode (command returns 0) or GUI mode (command returns 1).

Parameters

None.

compare_collection

`compare_collection` `<collection1> <collection2>`
    `[-order_dependent]`
    (TCL_SETUP/TCL_LEC mode)

Description

Compares two collections. Returns a value of 0 if all the objects in both collections are the same; returns 1 if the values in both collections are not the same.

Parameters

`<collection1>`	Specifies the first collection.
`<collection2>`	Specifies the second collection.
`-order_dependent`	Specifies that the order of the objects in both collections must be the same for the collections to be considered the same.

Example

```
TCL_LEC> set a [find_cfm -instance u* -collection]
collection_0
TCL_LEC> set a [find_cfm -instance ur* -collection]
collection_1
TCL_LEC> compare_collection $a $b -order_dependent
1
```

Notice that here it's the variable name instead of the collection name that needs to be passed as the argument.

copy_collection

```
copy_collection <collection>
```
(TCL_SETUP/TCL_LEC mode)

Returns a copy of the specified base collection. The base collection is not changed.

Parameters

`<collection>`	Specifies the collection that will be copied.

Example

```
TCL_LEC> copy_collection $a
collection_6
TCL_LEC> query_collection $a
/ud1 /u_iso1 /ur1 /ur2 /ur3
```

echo_result

```
echo_result [-on | -off]
```
(TCL_SETUP/TCL_LEC mode)

Turns the command result printing on or off.

Parameters

None.

encrypt

```
encrypt <inputfile> <outputfile>
    (TCL_SETUP/TCL_LEC mode)
```

Encrypts a Tcl command file.

Parameters

`inputfile`	Tcl file that you want to encrypt. This file is left unchanged by this command.
`outputfile`	Is the output file that is the encrypted form of the input file.

exit

```
exit
```
(TCL_SETUP/TCL_LEC mode)

Ends the existing Conformal session and returns to the operating system using the native Tcl exit command.

Parameters

None.

find

```
find <-Module | -Instance | -Port [-Input | -Output | -Bidir]
       | -Pin [ -Input | -Output| -Bidir]
       | -Net | -Gate | -Id>
     [-Single]
     [-HIERarchical]
     <object_name>
     [-EXCLIB]
     [-INDESIGN]
     [-LEAF]
     [-Golden | -Revised | -Both]
```
(TCL_SETUP/TCL_LEC mode)

Returns a design database object handle or list of handles for a design object or list of objects. The Tcl-based `find` and `find_cfm` commands are both useful for searching design information. They can both be used to look up design objects, such as instances, ports, pins, and nets. For gate queries, however, only the `find` command can be used.

Parameters

`-Module`	The specified `object_name` is a module name.
`-Instance`	The specified `object_name` is an instance name.
`-Port`	The specified `object_name` is a port name.
	`-Input` specifies an input port.
	`-Output` specifies an output port.
	`-Bidir` specifies a bidirectional port.
`-Pin`	The specified `object_name` is a pin name.
	`-Input` specifies an input pin.
	`-Output` specifies an output pin.
	`-Bidir` specifies a bidirectional pin.
`-Net`	The specified `object_name` is a net name.
`-Gate`	Specifies a flattened gate name.

`-Id`	Specifies the ID of a flattened gate. **Note:** Conformal automatically assigns ID numbers. They can differ from one version to another. Always use ID numbers assigned by the Conformal version you are currently running.
`-Single`	Returns none or the first found object handle instead of a list.
`-Hierarchical`	Specifies that `object_name` should be matched hierarchically. See Examples.
`<object_name>`	This is the name of a specified design object. It is a name in module context or hierarchical context. **Note:** Hierarchical objects start with "`/`".
`-EXCLIB`	Specifies that the object is exclusive to the library.
`-INDESIGN`	Find only objects that are in the current hierarchy (the current module is set by the `set_current_module` command; by default, the current module is the root module).
`-LEAF`	Find instance/pin at the primitive level.
`-Golden`	Applies to the Golden design only.
`-Revised`	Applies to the Revised design only.
`-Both`	Applies to the Golden and Revised designs.

Examples

The following example shows how the return value of the command can be saved to a variable for later reference:

```
set abc1 [find -instance /u1/U2]
set abc2 [find -single -instance /u1/U2]
...
get_nets [lindex $abc1 0]
get_nets $abc2
```

The following illustrates hierarchical name matching. For example, the first example returns no matches, the second example returns all instances that match the hierarchical name pattern "`*/*`", and the third example returns the instance that matches the name pattern "`*/x1`".

```
find -instance  */*


find -instance  -hierarchical */*
/u_rst_sync/RS2_reg
```

```
/u_rst_sync/RS1_reg
/u_pm/add_29_31
/u_tm/x1
/u_tm/x2
/u_tm/Iso_iso1

find -instance  -hierarchical */x1
/u_tm/x1
```

find_cfm

```
find_cfm -<object_type>
     [<patterns> | <object_list> | -OF_OBJects <object_list>]
     [-Filter <condition>]
     [-COLlection]
     [-HIERarchical]
     [-HSC <hierarchical_separator>]
     [-LIMit <int>]
     [-REGEXP]
     [-SCOPE <pathname>]
     [-SENsitive | -NOSENsitive]
     [-Golden | -Revised | -Both]
```
(TCL_SETUP/TCL_LEC mode)

Searches for the specified object and returns a design database object handle or list of handles for a design object or list of objects. The Tcl-based `find` and `find_cfm` commands are both useful for searching design information. They can both be used to look up design objects, such as instances, ports, pins, and nets. The `find_cfm` command, however, is the only choice for low power queries as it can also query for power intent objects, Liberty library cells, and 1801 rule filters.

Parameters

`-<object_type>`	Specifies the type of objects to be returned by this command. Where `object_type` is one of the following database object types: `-conformal`, `-design`, `-instance`, `-port`, `-pin`, `-net`, `-library`, `-libcell`, and `-libpin`. Power intent object: `-piaobj` (for a list of all supported power intent object options, refer to the man page of REPORT POWER INTENT (a Conformal Low Power command).
`<patterns>`	Specifies the name of the object to be queried. The default is * (wildcard)
`<object_list>`	List of objects. When specified, the `find_cfm` command returns all the objects of type `object_type` that are members of the `object_list` and that match the optional `-filter` condition.

`-OF_OBJects` Queries objects from another query or list of objects. The following lists the supported `-<object_types>` and objects allowed in `-of_objects`:

`-design`	`instance`
`-instance`	`net pin`
`-library`	`libcell`
`-libcell`	`libpin lib instance`
`-libpin`	`libcell pin`
`-net`	`pin instance port`
`-pin`	`net instance`
`-port`	`net`

`-Filter <condition>` Allows queries based on attributes. Condition must use the following format:

`<attribute_name><operator><value>`.

Where operator can be:==, !=, >=, >, <=, <, =~, and !~. Multiple conditions can be combined with the logical operators !, &&, ||, and parentheses ().

To report all valid attributes for an object, use `REPORT TCL ATTRIBUTES`.

`-COLlection` Collects the results of the find command as a Conformal collection object type. This option is useful when querying for a large amount of data. The `-collection` option creates a handle to the C language that is not restricted to the Tcl variable size. See examples section for more information.

`-HIERarchical` This option can be used only with the pattern option. This option specifies that the pattern should be matched hierarchically. For example, if the pattern matches the hierarchical name of an instance, it is included in the result.

`-HSC <hierarchical_separator>`

Specifies a user-defined hierarchical separator. By default, Conformal uses "/".

`-LIMit <int>` Specifies a limit on the number of returned objects.

`-REGEXP`	Specifies a regular expression for pattern matching. Curly brackets are required.
`-SCOPE <pathname>`	Specifies the root of the searching tree. By default, Conformal uses the root of the design tree.
`-SENsitive`	Specifies that the search is case sensitive. This is the default.
`-NOSENsitive`	Specifies that the search is case insensitive.
`-Golden`	Searches the Golden design. This is the default.
`-Revised`	Searches the Revised design.
`-Both`	Searches both the Golden and Revised design.

Examples

For example, the following code uses a regular expression to find all the instances whose name starts with RAM under the instance of DTMF_INST.

```
TCL_SETUP> find_cfm -instance -regexp {/DTMF_INST/RAM\w+} /DTMF_INST/
RAM_128x16_TEST_INST /DTMF_INST/RAM_256x16_TEST_INST
```

For example, to find all blackbox instances:

```
find_cfm -instance -hierarchical  -filter "is_bbox==1"
```

The following generates a collection using the results of a find command:

```
TCL_SETUP> find -instance xL* -collection
collection_0
```

Where collection_0 is the name of the collection, which is auto-generated by the the tool. Conformal auto increments the name for generated collections.

To view the contents of a collection, first assign the collection to a variable and use the `query_collection` Tcl command to view the contents:

```
TCL_SETUP> set z [find -instance xL* -collection]
collection_1
TCL_SETUP> query_collection $z
xLog
```

Wildcards in find_cfm -pin -hierarchical do not match across hierarchical separators. In this case, IC/I1 are two design hierarchies before the flatten command, and therefore IC*I1 will not match it.

```
TCL_SETUP> find_cfm -pin -hierarchical *
/IA/A /IA/Y /IA/IB/A /IA/IB/Y /IA/IB/IC/A /IA/IB/IC/Y /IA/IB/IC/I1/A /IA/IB/IC/I1/
Y
```

The following example returns empty.

```
TCL_SETUP> find_cfm -pin -hierarchical *IC*I1/A
```

foreach_in_collection

```
foreach_in_collection <var>
     <collection>
     <body>
```
(TCL_SETUP/TCL_LEC mode)

Iterate through all objects in the collection and apply the commands in the script provided as <body>.

Parameters

`<var>`	the variable used to iterate through the collection.
`<collection>`	Specifies the collection name.
`<body>`	Specifies the subsequent commands that are going to be applied to each item in the collection.

Example

```
TCL_SETUP> foreach_in_collection y $z {puts $y}
u_gsw
u1
```

get_attribute

```
get_attribute
     <object | object_list> [<-all | attribute_name>] [<var_name>]
```
(TCL_SETUP/TCL_LEC mode)

Retrieves the value of a specific attribute.

Parameters

`<object_type>`	Specifies the object type. For a list of all the available object types, use the `report_tcl_attribute` command.
`-all`	Returns all attribute names with their values in a list format, as `{<attribute_name> <value> <attribute_name> <value> ... }`
`<attribute_name>`	Specifies the name of the attribute. The default is * (wildcard).
`<var_name>`	Specifies the name of the Tcl variable that will contain the value of the requested attribute. If the variable does not exist in the current scope, it is created automatically. When this option is used, the command itself has no return value. When a list of objects is specified, the option `var_name` is not supported. For example, the following is not supported: `get_attr [find_cfm -pin {i1/din[*]}] bus_idx var1`

Examples

- In the following example, `OBJ1` is a port object. The following command returns an attribute name of `name` with a value of `clk1[7]`, an attribute name of `location` with a value of `test.v 41 {}`, and so on:

```
TCL_VERIFY> get_attribute OBJ1 -all
name {clk1[7]} location {test.v 41 {}} ...
```

- In the following example, the `get_attribute` command retrieves the value of two attributes for object `OBJ1`:

```
TCL_VERIFY> get_attribute OBJ1 bus_width var1 bus_idx var2
```

- In the following example, the `get_attribute` command retrieves the value for the `bus_idx` variable for objects `OBJ1` and `OBJ2`:

```
TCL_VERIFY> get_attribute OBJ1 OBJ2 bus_idx var1
```

- The following example illustrates that this command can accept a list of objects of the same type. For example:

  ```
  get_attribute [find_cfm -instance] library
  ```

  to return all the library names of the instances in the design.

- The following two commands set the same value in variable `varB`:

  ```
  set varB [get_attribute $mycell ref_name]
  get_attribute $mycell ref_name varB
  ```

get_compare_points

```
get_compare_points
        [-PO | -DFf| -DLat |-Bbox| -Cut] [-SIze | -SUPport]
        [-EQuivalent |-INvequivalent| -NONequivalent| -ABort| -NOtcompared]
        [-COunt]
```
(TCL_LEC mode)

Queries for multiple objects and returns a list of compare point object handles or a count of the selected compare points. A compare point handle is a `MAP_POINT` object handle.

Parameters

`-PO | -DFf| -DLat |-Bbox| -Cut`	
	Specifies the key point type(s), which can be one or more of
`-SIze | -SUPport`	Specifies the sort type.
`-EQuivalent |-INvequivalent| -NONequivalent| -ABort| -NOtcompared`	
	Specifies the result type(s).
`-COunt`	Returns a count of the selected compare points rather than a list of compare point handles.

Example

```
TCL_LEC> get_compare_points
(G)6(R)6 (G)7(R)7 (G)8(R)8 (G)9(R)9
```

get_compare_result

```
get_compare_result <obj_handle>
```
(TCL_LEC mode)

Returns compare results for the specified compare point. The key point is an object handle of the `COMPARE_POINT` type. The returned result is `POS_EQ`, `NEG_EQ`, `DIFF`, `NOT_COMPARED`, or `ABORT`.

Parameters

None.

Example

`get_compare_result` takes only arguments of the object handle type generated from the command `get_compare_point` The following example illustrates how to use this command:

```
TCL_LEC> set a [get_compare_points -dff]
(G)9(R)9
TCL_LEC> get_compare_result [lindex $a 0]
Equivalent
```

get_exit_code

```
get_exit_code
```
(TCL_SETUP/TCL_LEC mode)

Returns the run status without exiting the Conformal software. The status codes are:

Bit	Condition
0	Internal Error
1	No equivalent points during comparison
2	Command error
3	Unmapped points or extra POs
4	Non-equivalent points during comparison
5	Abort or uncompared points exist during any comparisons.
6	Abort or uncompared points exist during the last comparision or hierarchical comparison.

Note: For bits 0, and 2 through 5, once they are set to 1, they will remain at 1. For bit 1, once it is set to 0, it will remain at 0.

Examples

- Example Case 1:

  Start Conformal and then exit immediately
  Status = 2 (00010 in binary). There are no equivalent points since there was no comparison. Thus, bit 1 is set.

- Example Case 2:

  Comparison produced a non-equivalent point, an abort point, and an equivalent point.
  Status = 48 (110000 in binary). Bits 4 and 5 are set to flag the abort and non-equivalent points.

- Example Case 3:

  Comparison produced all non-equivalent points.
  Status = 18 (010010 in binary). Bits 1 and 4 are set to show two conditions: During this

session, Conformal found no equivalent points *and* the comparison produced non-equivalent points.

get_current_module

```
get_current_module
```
    (TCL_SETUP/TCL_LEC mode)

Returns the `MODULE` handle for the current module.

Parameters

None.

Example

```
TCL_LEC> get_current_module
err_detect
```

You can also use the related command `report_modules -all` to show all the available module names.

get_fanins

```
get_fanins <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Queries for multiple objects and returns a list of fan-in gate handles for the specified gate object handle. The type of object handle is a `FLAT_GATE`.

Example

The argument of the command `get_fanins` is not a gate name, but the type of an object handle, which is the 0th (first) element from a Tcl list generated from the Tcl built-in command `lindex`.

For example, instance `/syndrm/gen_synvalid1/del_regout` is an instantiation of the module `reg_rst (.d, .clk, .rst, .q [7:0] ),`

1. Use `report_gate` command to get the gate name that corresponds to the instance pin `q[0]`:

```
TCL_LEC> report_gate /syndrm/gen_synvalid1/del_regout/q\[0\]
=================================================================================
Pin-name  ID (Golden)  Type   Tie   Gate-name
=================================================================================
171          DFF           /syndrm/gen_synvalid1/del_regout/regout_reg[0]
------ Fanins --------------------------------------------------------------------
1: 'S'   533          ZERO         /syndrm/gen_synvalid1/del_regout/N$1
2: 'R'   533          ZERO         /syndrm/gen_synvalid1/del_regout/N$1
3: 'CK'  14           PI           /clk
4: 'D'   2109         MUX          /syndrm/gen_synvalid1/del_regout/U$1/U$1
------ Fanouts -------------------------------------------------------------------
1:       524          AND    (L0)  /syndrm/gen_synvalid1/dec/U$23
2:       2004         AND          /syndrm/gen_synvalid1/dec/U$22
3:       2067         XOR          /syndrm/gen_synvalid1/dec/U$2
=================================================================================
```

2. Use `get_fanins` to get list of fanin gates of the corresponding pins

```
TCL_LEC> get_fanins [lindex [find -gate \
{/syndrm/gen_synvalid1/del_regout/regout_reg[0]}] 0]
{/syndrm/gen_synvalid1/del_regout/N$1} {/syndrm/gen_synvalid1/del_regout/N$1} \
/clk {/syndrm/gen_synvalid1/del_regout/U$1/U$1}
```

This is consistent with the previously shown result of `report_gate`.

get_fanouts

```
get_fanouts <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Queries for multiple objects and returns a list of fan-out gate handles for the specified gate object handle. The type of object handle is a `FLAT_GATE`.

Parameters

None.

Example

The argument of the command `get_fanins` is not a gate name, but the type of an object handle, which is the 0th (first) element from a Tcl list generated from the Tcl built-in command `lindex`.

For example, in the design, instance /`syndrm/gen_synvalid1/del_regout` is an instantiation of the module `reg_rst (.d, .clk, .rst, .q [7:0] ),`

1. Use `report_gate` command to get the gate name corresponding to the instance pin q[0]:

```
TCL_LEC> report_gate /syndrm/gen_synvalid1/del_regout/q\[0\]
============================================================================
Pin-name  ID (Golden)  Type   Tie   Gate-name
============================================================================
171          DFF           /syndrm/gen_synvalid1/del_regout/regout_reg[0]
------ Fanins -------------------------------------------------------------
1: 'S'   533         ZERO         /syndrm/gen_synvalid1/del_regout/N$1
2: 'R'   533         ZERO         /syndrm/gen_synvalid1/del_regout/N$1
3: 'CK'  14          PI           /clk
4: 'D'   2109        MUX          /syndrm/gen_synvalid1/del_regout/U$1/U$1
------ Fanouts ------------------------------------------------------------
1:       524         AND    (L0)  /syndrm/gen_synvalid1/dec/U$23
2:       2004        AND          /syndrm/gen_synvalid1/dec/U$22
3:       2067        XOR          /syndrm/gen_synvalid1/dec/U$2
============================================================================
```

2. Use `get_fanouts` to get list of fanin gates of the corresponding pins

```
TCL_LEC> get_fanouts [lindex [find -gate \
{/syndrm/gen_synvalid1/del_regout/regout_reg[0]}] 0]
{/syndrm/gen_synvalid1/dec/U$23} {/syndrm/gen_synvalid1/dec/U$22} \
{/syndrm/gen_synvalid1/dec/U$2}
```

This is consistent with the previously shown result of report_gate.

get_gate_count

```
get_gate_count [-golden |-revised]
```
(TCL_SETUP/TCL_LEC mode)

Returns the gate count for the specified design. The default design is Golden.

Parameters

None.

Example

The following example illustrates how to use the command to get a gate count of the design:

```
TCL_LEC> get_gate_count
2116
```

get_gate_id

```
get_gate_id <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Returns the gate id for the specified flattened gate object handle.

Note: The Conformal software automatically assigns ID numbers. They can differ from one version to another. Always use ID numbers assigned by the software version you are running.

Parameters

None.

Example

The following example shows how to get the id number of the gate t[0]:

```
TCL_LEC> get_gate_id [lindex [find -gate {t[0]}] 0]
13
```

get_gate_type

```
get_gate_type <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Returns the gate type of the specified `obj_handle`. The type of object handle is `FLAT_GATE`.

Parameters

None.

Example

The following example shows how to get a gate's type. For hierarchical instances, use the command `report_gate` first to retrieve the gate name.

```
TCL_LEC> report_gate /syndrm/gen_synvalid1/del_regout/q\[0\]
================================================================================
Pin-name  ID (Golden)  Type   Tie   Gate-name
================================================================================
171          DFF           /syndrm/gen_synvalid1/del_regout/regout_reg[0]
------ Fanins ------------------------------------------------------------------
1: 'S'   533          ZERO         /syndrm/gen_synvalid1/del_regout/N$1
2: 'R'   533          ZERO         /syndrm/gen_synvalid1/del_regout/N$1
3: 'CK'  14           PI           /clk
4: 'D'   2109         MUX          /syndrm/gen_synvalid1/del_regout/U$1/U$1
------ Fanouts -----------------------------------------------------------------
1:       524          AND    (L0)  /syndrm/gen_synvalid1/dec/U$23
2:       2004         AND          /syndrm/gen_synvalid1/dec/U$22
3:       2067         XOR          /syndrm/gen_synvalid1/dec/U$2
================================================================================

TCL_LEC> get_gate_type [lindex [find -gate \
{/syndrm/gen_synvalid1/del_regout/regout_reg[0]}] 0]

DFF
```

get_handle_type

```
get_handle_type <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Returns the object type of the specified object handle. The returned result is one of the following strings:

- `MODULE`
- `MODULE_INSTANCE`
- `MODULE_PORT`
- `MODULE_INSTANCE_PIN`
- `MODULE_NET`
- `HIERARCHY_INSTANCE`
- `HIERARCHY_PORT`
- `HIERARCHY_INSTANCE_PIN`
- `HIERARCHY_NET`
- `FLAT_GATE`
- `MAP_POINT`

Parameters

None.

Example

The following examples illustrates how to use this command to retrieve various handle types:

```
TCL_LEC> get_handle_type [lindex [find -module {syndrome_erdet}] 0]
MODULE
TCL_LEC> get_handle_type [lindex [find -gate \
         {/syndrm/gen_synvalid1/del_regout/regout_reg[0]}] 0]
FLAT_GATE
TCL_LEC> get_handle_type [lindex [find -instance \
         {/syndrm/gen_synvalid1/del_regout}] 0]
HIERARCHY_INSTANCE
TCL_LEC> get_handle_type [lindex [find -net /syndrm/gen_synvalid1/rst] 0]
HIERARCHY_NET
```

get_instances

```
get_instances [-all_hierarchy] <obj_handle>
```
    (TCL_SETUP/TCL_LEC mode)

Queries for multiple objects and returns a list of instances associated with the specified object handle.

Parameters

`-all_hierarchy`	Use this argument in combination with the `HIERARCHY_NET` object handle type. It returns a list of hierarchical instance object handles that are associated with the given hierarchical net object.
`<obj_handle>`	The specified `obj_handle` as one of the following types: `MODULE`: lists the instances in the module `MODULE_INSTANCE`: lists the instance that identifies itself `MODULE_PORT`: not applicable `MODULE_INSTANCE_PIN`: lists the instances whose pins are listed in the specified object handle `MODULE_NET`: lists the instances that connect with the net `HIERARCHY_INSTANCE`: lists the hierarchical instance that identifies itself `HIERARCHY_PORT`: lists the hierarchical instance of which pins include this pin `HIERARCHY_INSTANCE_PIN`: lists the hierarchical instance of which pins include this pin `HIERARCHY_NET`: lists the hierarchical instances that connect with the net in a hierarchical context `FLAT_GATE`: lists the hierarchical instance that represents the flattened gate `MAP POINT`: not applicable

Example

This command takes the argument of the type of an object handle, which should be generated by both Tcl command `lindex` and Conformal Tcl command `find`.

The following example illustrates how to list the instances in a module

```
TCL_LEC> get_instances [lindex [find -module reg_rst] 0] \
  {regout_reg[7]} {regout_reg[6]} {regout_reg[5]} {regout_reg[4]} \
  {regout_reg[3]} {regout_reg[2]} {regout_reg[1]} {regout_reg[0]} {U$1} {U$2} \
  {U$3} {U$4} {U$5} {U$6} {U$7} {U$8} {U$9} {U$10} {U$11} {U$12} {U$13} {U$14} \
  {U$15} {U$16} {U$17} {U$18}
```

The following example illustrates how to list the instance that identifies itself

```
TCL_LEC> get_instances [lindex [find -gate \
  {/syndrm/gen_synvalid1/del_regout/regout_reg[0]}] 0] \
  {/syndrm/gen_synvalid1/del_regout/regout_reg[0]}
```

get_keypoint

```
get_keypoint [-golden | -revised] <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Returns the key point (`FLAT_GATE` type) of a specified compare point. The object handle will be a `MAP_POINT` type.

Parameters

`-golden`	Specifies that the compare point is in the Golden design.
`-revised`	Specifies that the compare point is in the Revised design.

Example

The following example illustrates how to get the key point from a specified pair of mapped points

```
TCL_LEC> set a [get_map_points -dff]
(G)26(R)171 (G)27(R)74 (G)28(R)75 (G)29(R)76 (G)30(R)77

TCL_LEC> get_keypoint [lindex $a 0]
/uncorrected_error_reg

TCL_LEC> get_keypoint [lindex $a 1]
/syndrm/macreg/q_reg[7]
```

get_license_mode

```
get_license_mode
```
    (TCL_SETUP/TCL_LEC mode)

Returns the current license mode.

License	License Mode
`-L`	`LEC-L`
`-XL`	`LEC-XL`
`-GXL`	`LEC-GXL`
`-LP`	`Low_Power_EC`
`-LPGXL`	`Low_Power_EC_GXL`
`-ECO`	`ECO`
`-ECOGX`	`ECO_GXL`
`-SL4`	`Smart_LEC_4CPU`
`-SL4 -LP`	`Low_Power_EC Smart_LEC_4CPU`
`-SL4 -LPGXL`	`Low_Power_EC_GXL Smart_LEC_4CPU`
`-CCD`	`ccd_l`
`-CCD XL`	`ccd_xl`

Parameters

None.

Example

The following example illustrates how to get the license being used in the current session:

```
TCL_LEC> get_license_mode
Low_Power_EC

TCL_SETUP> get_license_mode
Low_Power_GXL
```

get_map_points

```
get_map_points
         [-PO | -DFf| -DLat |-Bbox| -Cut] [-SIze | -SUPport]
         [-EQuivalent |-INvequivalent| -NONequivalent| -ABort| -NOtcompared]
         [-COunt]
```
(TCL_LEC mode)

Queries for multiple objects and returns a list of map point object handles or a count of the selected map points. A map point handle is a `MAP_POINT` object handle.

Parameters

`-PO	-DFf| -DLat |-Bbox| -Cut`	
	Specifies the key point type(s) .
`-SIze	-SUPport`	Specifies the sort type.
`-EQuivalent |-INvequivalent| -NONequivalent| -ABort| -NOtcompared`	
	Specifies the result type(s).
`-COunt`	Returns a count of the selected map points rather than a list of map point handles.

Example

The following example shows how to get the map points of the selected category:

```
TCL_LEC> get_map_point -PO
(G)24(R)24 (G)25(R)25

TCL_LEC> get_map_point -DFF
(G)26(R)171 (G)27(R)74 (G)28(R)75 (G)29(R)76 (G)30(R)77

TCL_LEC> get_map_point -nonequivalent
(G)24(R)24 (G)155(R)154
```

get_module_definition

```
get_module_definition <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Returns a module definition for the specified object handle, where `<obj_handle>` is one of the following types:

- `MODULE`: define itself
- `MODULE_INSTANCE`: define the module of the specified instance
- `MODULE_PORT`: not applicable
- `MODULE_INSTANCE_PIN`: not applicable
- `MODULE_NET`: not applicable
- `HIERARCHY_INSTANCE`: define the module of the specified instance in a hierarchical context.
- `HIERARCHY_PORT`: not applicable
- `HIERARCHY_INSTANCE_PIN`: not applicable
- `HIERARCHY_NET`: not applicable
- `FLAT_GATE`: define the module of the specified flattened gate
- `MAP POINT`: not applicable

Parameters

None.

Example

For modules, this command returns the definition:

```
TCL_LEC> get_module_definition [lindex [find -module reg_3] 0]
reg_3
```

For instances, this command returns the module name from which this instance is instantiated:

```
TCL_LEC> get_module_definition [lindex [find -instance {/syndrm}] 0]
syndrome_erdet
```

get_module_instances

```
get_module_instances <modTclObj>
```
(TCL_SETUP/TCL_LEC mode)

Returns a list of instances for the specified module.

Parameters

None.

Example

```
TCL_SETUP> set my_mod [find -single -module my_dlat]
my_dlat

TCL_SETUP> set j [get_module_instances $my_mod]
u1 u2 u3
```

get_names

```
get_names <obj_handles>
```
(TCL_SETUP/TCL_LEC mode)

Queries for multiple objects and returns a name or list of names for the specified object handles.

Parameters

None.

Example

The following example illustrates how to use the commands to retrieve the names from a list of objects

```
TCL_LEC> find -instance {/syndrm/gfcmult1*}
/syndrm/gfcmult1 /syndrm/gfcmult1_1 /syndrm/gfcmult1_2 /syndrm/gfcmult1_3 /syndrm/
gfcmult1_4 /syndrm/gfcmult1_5 /syndrm/gfcmult1_6 /syndrm/gfcmult1_7 /syndrm/
gfcmult1_8 /syndrm/gfcmult1_9 /syndrm/gfcmult1_10 /syndrm/gfcmult1_11 /syndrm/
gfcmult1_12 /syndrm/gfcmult1_13 /syndrm/gfcmult1_14 /syndrm/gfcmult1_15

TCL_LEC> get_names [lindex [find -instance {/syndrm/gfcmult1*}] 1]
/syndrm/gfcmult1_1

TCL_LEC> get_names [lindex [find -instance {/syndrm/gfcmult1*}] 14]
/syndrm/gfcmult1_14
```

get_nets

```
get_nets [-all_hierarchy] <obj_handle>
```
    (TCL_SETUP/TCL_LEC mode)

Queries for multiple objects and returns a list of nets for the specified object handle.

Parameters

`-all_hierarchy`	Use this argument in combination with the `HIERARCHY` object handle type. It returns a list of hierarchical net object handles that are associated with the given hierarchical design object (for example, `HIERARCHY_INSTANCE`).
`<obj_handle>`	The specified `obj_handle` as one of the following types: `MODULE`: lists the nets of the specified module `MODULE_INSTANCE`: lists the nets that connect with the specified instance `MODULE_PORT`: lists the nets that connect with the specified port `MODULE_INSTANCE_PIN`: lists the nets that connect with the specified pin `MODULE_NET`: lists the net that identifies itself `HIERARCHY_INSTANCE`: lists the nets that connect with the specified instance in a hierarchical context `HIERARCHY_PORT`: lists the nets that connect with the specified port in a hierarchical context `HIERARCHY_INSTANCE_PIN`: lists the nets that connect with the specified pin in a hierarchical context `HIERARCHY_NET`: lists the net that identifies itself `FLAT_GATE`: lists the nets that connect with the specified flattened gate in a hierarchical context `MAP POINT`: not applicable

Example

User can use `get_nets *` to report all the nets used in the design.

The following code illustrates using the command to get a list of the nets that connects to instance /syndrm/gfcmult1_1

```
TCL_LEC> get_nets [lindex [find -instance {/syndrm/gfcmult1_1}] 0]

{/syndrm/synout1_14[7]} {/syndrm/synout1_14[6]} {/syndrm/synout1_14[5]} {/syndrm/
synout1_14[4]} {/syndrm/synout1_14[3]} {/syndrm/synout1_14[2]} {/syndrm/
synout1_14[1]} {/syndrm/synout1_14[0]} /syndrm/clk {/syndrm/mout_14[7]} {/syndrm/
mout_14[6]} {/syndrm/mout_14[5]} {/syndrm/mout_14[4]} {/syndrm/mout_14[3]} {/
syndrm/mout_14[2]} {/syndrm/mout_14[1]} {/syndrm/mout_14[0]}
```

get_parent

```
get_parent <obj_handle>
```
    (TCL_SETUP/TCL_LEC mode)

Returns a handle to the parent of the specified object handle, where `<obj_handle>` is one of the following types:

- `MODULE`: not applicable
- `MODULE_INSTANCE`: return the module that contains the specified instance
- `MODULE_PORT`: return the module that contains the specified port
- `MODULE_INSTANCE_PIN`: return the module that contains the specified pin
- `MODULE_NET`: return the module that contains the specified net
- `HIERARCHY_INSTANCE`: return the module definition of the instance
- `HIERARCHY_PORT`: return the hierarchical instance that contains the specified port in a hierarchical context
- `HIERARCHY_INSTANCE_PIN`: return the hierarchical instance that contains the specified pin in a hierarchical context
- `HIERARCHY_NET`: return the hierarchical instance that contains the specified net in a hierarchical context
- `FLAT_GATE`: return the hierarchical instance that contains the specified flattened gate in a hierarchical context
- `MAP POINT`: not applicable

Parameters

None.

Example

Refer to the following example on how to get a handle to the parent of the specified object handle

```
TCL_LEC> get_parent [lindex [find -instance {/syndrm/gfcmult1_1}] 0]
/syndrm
```

```
TCL_LEC> get_parent [lindex [find -net {/syndrm/gfcmult1_1/clk}] 0]
/syndrm/gfcmult1_1
```

get_pins

```
get_pins [-all_hierarchy] <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Queries for multiple objects and returns a list of pins associated with the specified object handle.

Parameters

`-all_hierarchy`	Use this argument in combination with the `HIERARCHY_NET` object handle type. It returns a list of hierarchical instance pin object handles that are associated with the given hierarchical net object.
`<obj_handle>`	The specified `obj_handle` as one of the following types: `MODULE`: lists the pins of the specified module `MODULE_INSTANCE`: lists the pins of the specified instance `MODULE_PORT`: lists the pins that connect with the net of the specified port `MODULE_INSTANCE_PIN`: lists the pin that identifies itself `MODULE_NET`: lists the pins that connect with the specified net `HIERARCHY_INSTANCE`: lists the pins of the specified instance in a hierarchical context `HIERARCHY_PORT`: lists the pins that connect with the net of the specified port in a hierarchical context `HIERARCHY_INSTANCE_PIN`: lists the nets that connect with the specified pin in a hierarchical context `HIERARCHY_NET`: lists the pin that identifies itself `FLAT_GATE`: lists the pins of the specified flattened gate in a hierarchical context `MAP POINT`: not applicable

Example

```
TCL_LEC> get_pins [lindex [find -instance {/syndrm/macreg}] 0]
// Command: find -instance /syndrm/macreg
```

```
{/syndrm/macreg/d[7]} {/syndrm/macreg/d[6]} {/syndrm/macreg/d[5]} {/syndrm/
macreg/d[4]} {/syndrm/macreg/d[3]} {/syndrm/macreg/d[2]} {/syndrm/macreg/d[1]} {/
syndrm/macreg/d[0]} /syndrm/macreg/clk {/syndrm/macreg/q[7]} {/syndrm/macreg/
q[6]} {/syndrm/macreg/q[5]} {/syndrm/macreg/q[4]} {/syndrm/macreg/q[3]} {/syndrm/
macreg/q[2]} {/syndrm/macreg/q[1]} {/syndrm/macreg/q[0]}

TCL_LEC> get_pins [lindex [find -module {reg_3}] 0]

// Command: find -module reg_3

{U$1/I0} {U$1/O} {U$2/I0} {U$2/O} {U$3/I0} {U$3/O} {U$4/I0} {U$4/O} {U$5/I0} {U$5/
O} {U$6/I0} {U$6/O} {U$7/I0} {U$7/O} {U$8/I0} {U$8/O} {q_reg[7]/S} {q_reg[7]/R}
{q_reg[7]/CK} {q_reg[7]/D} {q_reg[7]/Q} {q_reg[7]/QN} {q_reg[6]/S} {q_reg[6]/R}
{q_reg[6]/CK} {q_reg[6]/D} {q_reg[6]/Q} {q_reg[6]/QN} {q_reg[5]/S} {q_reg[5]/R}
{q_reg[5]/CK} {q_reg[5]/D} {q_reg[5]/Q} {q_reg[5]/QN} {q_reg[4]/S} {q_reg[4]/R}
{q_reg[4]/CK} {q_reg[4]/D} {q_reg[4]/Q} {q_reg[4]/QN} {q_reg[3]/S} {q_reg[3]/R}
{q_reg[3]/CK} {q_reg[3]/D} {q_reg[3]/Q} {q_reg[3]/QN} {q_reg[2]/S} {q_reg[2]/R}
{q_reg[2]/CK} {q_reg[2]/D} {q_reg[2]/Q} {q_reg[2]/QN} {q_reg[1]/S} {q_reg[1]/R}
{q_reg[1]/CK} {q_reg[1]/D} {q_reg[1]/Q} {q_reg[1]/QN} {q_reg[0]/S} {q_reg[0]/R}
{q_reg[0]/CK} {q_reg[0]/D} {q_reg[0]/Q} {q_reg[0]/QN}
```

get_ports

```
get_ports [-all_hierarchy] <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Queries for multiple objects and returns a list of ports associated with the specified object handle.

Parameters

`-all_hierarchy`	Use this argument in combination with the `HIERARCHY_NET` object handle type. It returns a list of hierarchical port object handles that are associated with the given hierarchical net object.
`<obj_handle>`	The specified `obj_handle` as one of the following types: `MODULE`: lists the ports of the specified module `MODULE_INSTANCE`: lists the ports of the specified instance `MODULE_PORT`: lists the port that identifies itself `MODULE_INSTANCE_PIN`: returns the relative port of the specified pin `MODULE_NET`: lists the ports that connect with the specified net `HIERARCHY_INSTANCE`: lists the ports of the specified instance in a hierarchical context `HIERARCHY_PORT`: lists the port that identifies itself `HIERARCHY_INSTANCE_PIN`: returns the relative port of the specified pin in a hierarchical context `HIERARCHY_NET`: lists the ports that connect with the specified net in a hierarchical context `FLAT_GATE`: lists the ports of the specified flattened gate `MAP POINT`: not applicable

Example

```
TCL_LEC> get_ports [lindex [find -module {reg_3}] 0]
// Command: find -module reg_3
{d[7]} {d[6]} {d[5]} {d[4]} {d[3]} {d[2]} {d[1]} {d[0]} clk {q[7]} {q[6]} {q[5]}
{q[4]} {q[3]} {q[2]} {q[1]} {q[0]}
```

```
TCL_LEC> get_ports [lindex [find -instance {/syndrm/macreg}] 0]
// Command: find -instance /syndrm/macreg

{syndrm/macreg/d[7]} {syndrm/macreg/d[6]} {syndrm/macreg/d[5]} {syndrm/macreg/
d[4]} {syndrm/macreg/d[3]} {syndrm/macreg/d[2]} {syndrm/macreg/d[1]} {syndrm/
macreg/d[0]} syndrm/macreg/clk {syndrm/macreg/q[7]} {syndrm/macreg/q[6]} {syndrm/
macreg/q[5]} {syndrm/macreg/q[4]} {syndrm/macreg/q[3]} {syndrm/macreg/q[2]}
{syndrm/macreg/q[1]} {syndrm/macreg/q[0]}
```

get_primitive_type

```
get_primitive_type <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Returns the primitive type for the specified instance object handle, where `<obj_handle>` is one of the following:

- `MODULE_INSTANCE`
- `HIERARCHY_INSTANCE`
- `FLAT_GATE`

Parameters

None.

Example

The following example illustrates how to use the command to get the gate type of the specified gate.

```
TCL_LEC> report_gate /u1/x0/Q
================================================================================
Pin-name  ID (Golden)  Type   Tie   Gate-name (Library: LP_DFF)
================================================================================
9            DFF          /u1/x0/U$1
------ Fanins ------------------------------------------------------------------
1: 'S'   12           ZERO         /ZERO
2: 'R'   13           ZERO         /ZERO
3: 'CK'  14           AND    (L0)  /u1/x0/U$3
4: 'D'   18           INV          /u1/u0/U$1
------ Fanouts -----------------------------------------------------------------
1:       19           BUF          /u1/x0/U$1/Q
================================================================================
0
TCL_LEC> get_primitive_type [lindex [find -gate /u1/x0/U\$1/Q] 0]
dff
```

get_project_name

```
get_project_name
```
(TCL_SETUP/TCL_LEC mode)

Returns the current project name.

Parameters

None.

Example

```
TCL_SETUP> get_project_name
// Error: Project is not set

TCL_SETUP> set_project_name project1
TCL_SETUP> get_project_name
project1
```

get_property

```
get_property <obj_handle> <property_type>
```
(TCL_SETUP/TCL_LEC mode)

Returns the property of the specified object handle. The following lists the object handle one of the following types, with their property types and return values.

- `MODULE`

  Property Type: `BlackBox`; Return Value: `YES`, `NO`

  Property Type: `InLib`; Return Value: `YES`, `NO`

  Include the `InLib` property to test whether the module definition is defined in the library space (`YES`) or design space (`NO`): that is, it tests whether the module definition is imported by the `READ LIBRARY` or `READ DESIGN` command.

- `MODULE_INSTANCE`

  Property Type: `Primitive`; Return Value: `YES`, `NO`

  Property Type: `CutPoint`; Return Value: `YES`, `NO`

  Include the CutPoint property to test whether the module definition is a cut point: that is, it tests whether you have run the `ADD CUT POINT` command on this module definition.

- `MODULE_PORT`

  Property Type: `Direction`; Return Value: `INPUT`, `OUTPUT`, `BIDIR`

- `MODULE_INSTANCE_PIN`

  Property Type: `Direction`; Return Value: `INPUT`, `OUTPUT`, `BIDIR`

- `MODULE_NET`

  Property Type: `NetType`; Return Value: `TIE0`, `TIE1`, `WAND`, `WOR`, `TRI`, `TRI0`, `TRI1`, `TRIAND`, `TRIOR`, `TRIREG`, `REG`, `GLOBAL`, `TIEX`, `TIEZ`, `WIRE`

  Note: `GLOBAL` applies to those signals that cross multiple modules (for example, assertions).

- `HIERARCHY_INSTANCE`

  Property Type: `BlackBox`; Return Value: `YES`, `NO`

- `HIERARCHY_PORT`

  Property Type: `Direction`; Return Value: `INPUT`, `OUTPUT`, `BIDIR`

- HIERARCHY_INSTANCE_PIN

  Property Type: Direction; Return Value: INPUT, OUTPUT, BIDIR

- HIERARCHY_NET

  Property Type: NetType; Return Value: TIE0, TIE1, WAND, WOR, TRI, TRI0, TRI1, TRIAND, TRIOR, TRIREG, REG, GLOBAL, TIEX, TIEZ, WIRE

  Note: GLOBAL applies to those signals that cross multiple modules (for example, assertions).

- FLAT_GATE

  Property Type: BlackBox; Return Value: YES, NO

  Property Type: Golden; Return Value: YES, NO

  Property Type: Revised; Return Value: YES, NO

- MAP POINT: not applicable

  Property Type: Result; Return Value: POS_EQ, NEG_EQ, DIFF, ABORT, UNKNOWN

Parameters

None.

Example

```
TCL_LEC> get_property [lindex [find -gate /u1/x0/U\$1/Q] 0] golden
YES

TCL_SETUP> get_property [lindex [find -module RAM] 0] Blackbox
YES
```

get_relative_path

```
get_relative_path  <full_file_path>
```
(TCL_SETUP/TCL_LEC mode)

Returns an equivalent file path that is relative to the current working directory.

Parameters

`full_file_path`	Specifies the path for which to return a relative path.

Example

For example, assume the current working directory is /home/conformal:

```
TCL_LEC> pwd
/home/conformal
TCL_LEC> get_relative_path /home/conformal/foo/bar
foo/bar
TCL_LEC> get_relative_path /home/library/etc/mylib.lib
../library/etc/mylib.lib
```

get_root_module

```
get_root_module [-golden | -revised]
```
(TCL_SETUP/TCL_LEC mode)

Returns the name of the root module for the specified design. The default design is Golden.

Parameters

None.

Example

The following example illustrates how to look up the root module name.

```
TCL_LEC> get_root_module
err_detect
```

Furthermore, users can also use the command `report_modules -all` to list all the module names on both Golden and Revised side.

get_top_module

```
get_top_module [-golden | -revised]
```
(TCL_SETUP/TCL_LEC mode)

Returns the name of the top module for the specified design. The default design is Golden.

Parameters

None.

get_unmap_points

```
get_unmap_points
    [-PI|-PO|-DFF|-Dlat|-Bbox|-Cut|-Z]
    [-INPUT][-OUTPUT]
    [-Size|-SUpport]
    [-Extra|-UNReachable|-NOTmapped]
    [-GOLden|-REvised|-BOth]
    [-COunt]
```
(TCL_LEC mode)

Queries for multiple objects and returns a list of unmap point object handles or a count of the selected unmap points. An unmap point handle is a `MAP_POINT` object handle.

Parameters

`-PI	-PO	-DFf| -DLat |-Bbox| -Cut	-Z`	
	Specifies the key point type(s).
`-INPUT`	Lists unmapped inputs of blackbox or sequential elements.
`-OUTPUT`	Lists unmapped outputs of blackbox or sequential elements.
`-Size	-SUpport`	Specifies the sort type.
`-Extra	-UNReachable	-NOTmapped`	
	Specifies the result type(s).
`-GOlden`	For the Golden design.
`-REvised`	For the Revised design.
`-BOth`	For both the Golden and Revised design. *This is the default*.
`-COunt`	Returns a count of the selected unmap points rather than a list of unmap point handles.

get_version_info

```
get_version_info
```
(TCL_SETUP/TCL_LEC mode)

Returns a TCL list of the version related info, including `version_num`, `build_date`, 32 or 64 bit, host name, and platform.

Parameters

None.

help

```
help [command_name]
```
(TCL_SETUP/TCL_LEC mode)

Returns the command usage of the specified command or a list of all Conformal Tcl commands if you do not specify a command.

Parameters

None.

index_collection

```
index_collection  <collection> <index>
```
(TCL_SETUP/TCL_LEC mode)

Returns a object that exists at the specified index of the specified object collection.

Parameters

`<collection>`	Specifies the collection name.
`<index>`	Specifies the index number. The index starts from 0.

Example

The following command set returns the object that exists at the index number 2 in the object collection referenced by $insts:

```
TCL_LEC> set insts [find_cfm -instance * -collection]
collection_0
TCL_LEC> set obj [index_collection $insts 2]
/ur1
```

ncdecrypt

```
ncdecrypt <input_file | -version>
     (TCL_SETUP/TCL_LEC mode)
```

Decrypts the specified file, which was encrypted using the `ncencrypt` command. Writes the contents into the buffer for the Tcl interpreter to evaluate. This command does not write out to a file.

If the encrypted file uses a user-defined key, you must set the environment variable `NCPROTECT_KEYDB` to the directory that contains the public key needed to decrypt the file before using the `ncdecrypt` command.

```
setenv NCPROTECT_KEYDB <path>
```

Parameters

input_file	File to decrypt.
-version	Report the NC library version.

objtype

```
objtype <obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Returns the type of an native TCL object, for example string and list, or Conformal design object handle type (i.e. `vpxhandle`).

Parameters

None.

Example

```
TCL_LEC> objtype [lindex [find -instance {/syndrm/gfcmult1_1}] 0]
vpxhandle

TCL_LEC> objtype [lindex [find -net {/syndrm/gfcmult1_1/clk}] 0]
vpxhandle

TCL_LEC>  set a [get_map_points -po]
(G)24(R)24 (G)25(R)25

TCL_LEC> objtype a
parsedVarName
```

query_collection

```
query_collection <collection>
     [-limit <integer>]
```
    (TCL_SETUP/TCL_LEC mode)

Returns as a TCL list all or specified number of objects from a collection of objects.

Parameters

`<collection>`	Specifies the collection that will be sorted.

Example

Please refer to the example of sort_collection.

redirect

```
redirect
     [-append] [-variable] <target> <command>
```
(TCL_SETUP/TCL_LEC mode)

Redirects output from a specified command to a file or Tcl variable.

Parameters

`-append`	Appends the generated output to the specified Tcl variable or file.
`-variable`	Redirects output to a Tcl variable. If `-variable` is not specified, a file is created with the specified target name.
`<target>`	Redirects output from the specified command to this file/ variable. If the target file or variable already exists and `-append` is not specified, the command overwrites the existing variable or file.
`<command>`	Tcl command to redirect. Commands must be quoted. For example: `redirect -variable gates "report_gate"`

Example

Tcl command to redirect. Commands must be quoted. For example:

```
TCL_LEC> redirect -variable mod "report_modules -golden"
0
TCL_LEC> echo $mod
Golden:
err_detect
  decode
  syndrome_erdet
     gfcmult_12
     reg_3
     gfcmult_14
     gfcmult_16
     gfcmult_18
     gfcmult_20
     gfcmult_22
     gfcmult_24
     gfcmult_26
     gfcmult_28
     gfcmult_30
```

```
    gfcmult_32
    gfcmult_34
    gfcmult_36
    gfcmult_38
    gfcmult_40
    gfcmult_42
    strobex_rst_44
         add_45
         mux_46 reg_rst
```

remove_from_collection

```
remove_from_collection <collection1>
     <collection2 | object_list>
     [-intersect]
```
(TCL_SETUP/TCL_LEC mode)

Creates a new collection by removing the objects specified as a collection <collection2> or as a TCL list from collection <collection1>.

Parameters

`<collection1>`	Specifies the first collection.
`<collection2 | object_list>`	Specifies the second collection.
`-intersect`	Specifies reverse matching, the tool removes the items that are not collection2/object_list from the base collection.

Example

```
TCL_LEC> set b [find_cfm -instance ur1]
/ur1
TCL_LEC> set c [remove_from_collection [find_cfm -instance * -collection] $b]
collection_2
TCL_LEC> query_collection $c
/ud1 /u_iso1 /ur2 /ur3
```

set_current_module

```
set_current_module <[which_design] module_name | obj_handle>
```
(TCL_SETUP/TCL_LEC mode)

Changes the current module. By default, the current module is the root module.

Parameters

`which_design`	Includes one or more of the following options. **Note:** `which_design` is ignored if you use `<obj_handle>`. `-golden` sets the current module for the Golden design. *This option is the default value.* `-revised` sets the current module for the Revised design.
`<module_name>`	The specified name is a module name.
`<obj_handle>`	This argument is a `MODULE` handle.

Example

The following example illustrates how to use the command to reset the current module.

```
TCL_LEC> get_current_module
err_detect
TCL_LEC> report_modules -all
Golden design: err_detect(T)  decode  syndrome_erdet  gfcmult_22  gfcmult_12
  gfcmult_34  gfcmult_16  gfcmult_18  gfcmult_20  gfcmult_32  gfcmult_24
  gfcmult_30  gfcmult_38  gfcmult_40  gfcmult_42  strobex_rst_44  gfcmult_14
  gfcmult_26  gfcmult_28  gfcmult_36  add_45  mux_46  reg_3  reg_rst
Revised design: err_detect(T)  add_45  reg_rst  mux_46  gfcmult_12  reg_3
  gfcmult_14  gfcmult_16  gfcmult_18  gfcmult_20  gfcmult_22  gfcmult_24
  gfcmult_26  gfcmult_28  gfcmult_30  gfcmult_32  gfcmult_34  gfcmult_36
  gfcmult_38  gfcmult_40  gfcmult_42  strobex_rst_44  syndrome_erdet  decode
  0
TCL_LEC> set_current_module decode
TCL_LEC> get_current_module
decode
```

sizeof_collection

```
sizeof_collection <collection>
```
(TCL_SETUP/TCL_LEC mode)

Returns size of the specified object collection.

Parameters

`<collection>`	Specifies the collection.

Example

The following example shows how to query the size of a collection:

```
TCL_LEC> set insts [find_cfm -instance * -collection]
collection_0
TCL_LEC> set obj_cnt [sizeof_collection $insts]
5
```

sort_collection

```
sort_collection <collection> [-descending]
```
    (TCL_SETUP/TCL_LEC mode)

Returns a sorted collection of objects by ascending (default) or descending order.

Parameters

`<collection>`	Specifies the collection that will be sorted.
`<-descending>`	Specifies that the output will be descending order.

Example

```
TCL_LEC> set a [find_cfm -instance u* -collection]
collection_0
TCL_LEC> sort_collection $a
collection_1
TCL_LEC> query_collection $a
/ud1 /u_iso1 /ur1 /ur2 /ur3
```

tcl_set_command_name_echo

```
tcl_set_command_name_echo <OFF | ON>
```
(TCL_SETUP/TCL_LEC mode)

Turns Tcl command set (including nested Tcl commands and Tcl commands in a sourced Tcl script/file) name echoing on or off. Default is OFF.

Parameters

None.