cadence™

Conformal® ECO User Guide

Product Version 23.2
October 2023

Cadence Design Systems, Inc., 2655 Seely Avenue, San Jose, CA 95134, USA

Contents

Preface ECO

About This Manual

The Cadence® Encounter® Conformal® ECO Designer enables designers to implement RTL engineering change orders (ECOs) for pre- and post-mask layout. It features automatic ECO analysis, logic optimization, and design netlist modification, with equivalence checking.

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.

How This Manual Is Organized

The chapters in this manual are organized to follow the flow of tasks through the design process. Because of variations in design implementations and methodologies, the order of the chapters will not correspond to any specific design flow.

Each chapter focuses on the concepts and tasks related to the particular design phase or topic being discussed.

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.

- *Conformal Equivalence Checking Reference*

  Describes the commands and modeling messages for the Encounter® Conformal® Equivalence Checking solutions, including Conformal ECO Designer.

- *Conformal Equivalence Checking User Guide*

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

Additional Learning Resources

Cadence offers the following training course on Conformal ECO:

- *Conformal ECO*

Conventions

Syntax Structure

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 shows that the command you are typing continues on the next line.
...	Indicates multiple entries of an argument.
*	Indicates that Conformal lets the wildcard (*) represent any zero or more characters.

GUI Convention

Convention	Definition
Menu – Command	Indicates command sequences under a menu. For example: Choose *File – Read Design.*
Left-click	Click the left mouse button on the specified item.
Right-click	Click the right mouse button on the specified item.
Click	Click the left mouse button unless otherwise specified.
Double-Click	Click twice on the left mouse button.
Drag	Press and hold the left mouse button, and then move the pointer to the destination and release the button.

1

Introduction to the Conformal ECO Solution

This manual documents the commands, features, and flows that are specific to the Encounter® Conformal® ECO solution:

Overview

Note: This feature requires a Conformal ECO XL or GXL license.

Conformal ECO Designer offers functional ECO analysis, optimization, and generation capability. It combines proven equivalence checking and functional checks, and uses formal techniques to analyze and implement the functional ECO.

This chapter describes the Conformal ECO methodology and how to use the ECO Designer software to automatically implement functional ECOs. You can provide the resulting netlist to SoC Encounter for incremental optimization and place-and-route.

ECOs can also be classified according to optimization stage:

- Pre-mask ECOs are done before the chip-mask is manufactured. For a pre-mask ECO, the software assumes any cell in the library is available for implementing the ECO.
- Post-mask ECOs are done after the chip-mask is manufactured. For a post-mask ECO, the software restricts the implementation to only use spare cells based on their physical location in the design and their timing requirements.

Note: Conformal ECO does not support retimed designs and/or blocks.

Features

Conformal ECO Designer combines logic equivalence checking (for the most complex SoC and datapath-intensive designs) with functional ECO analysis, design netlist modification, and logic optimization.

- Equivalence checking for ECOs

  During development, a design undergoes numerous iterations prior to final layout, and each step in this process has the potential to introduce logical bugs. Conformal ECO Designer checks the functional equivalence of different versions of a design at these various stages and enables designers to identify and correct errors as soon as they are introduced. Equivalence checking also plays an important role in the ECO implementation process. It helps the ECO analysis tool identify which modules and logic cones in the design require change to implement the ECO. Equivalence checking is also used at the end of the process to ensure the ECO implementation was successful both for front-end and back-end signoff.

- Functional ECO analysis

  Conformal ECO Designer has a built-in ECO analysis engine that can identify the differences between the original design netlist and the new design netlist. Users can perform ECO analysis on the entire design or on specific modules within the design hierarchy, which is typically more efficient. Once the ECO analysis step is completed and the logic change optimized, Conformal ECO Designer performs the necessary netlist modifications to achieve the new function in the original design netlist. The output is the ECO netlist. Alternately, Conformal ECO Designer can write out an ECO script that can be used to make direct changes to the place-and-route database.

- Physically-aware premask ECOs

  Conformal ECO Designer (GXL) has the ability to read the DEF layout database corresponding to the original place-and-route design netlist, capTbl, LEF, Liberty synthesis libraries, and SDC timing constraints to optimize, estimate routing, and legally place the generated ECO logic into the design floorplan. The output of Conformal ECO Designer is the ECO netlist and the corresponding placement DEF file. This flow can reduce timing closure iterations during place and route especially in late state pre-mask ECO situations.

- Spare gate mapping for post-mask ECOs

  Conformal ECO Designer (GXL) has the ability to read the DEF layout database corresponding to the original design netlist, capTbl, LEF, Liberty synthesis libraries, and

SDC to optimally map ECO logic to standard cell and gate array spare gates. The mapping engine is timing- and spare-cell-location aware. This capability enables the designer to get an early estimate of the ECO feasibility and effectively drive the backend implementation flow. Conformal ECO Designer (GXL) can also recycle freed-up cells in the ECO mapping process. The output is the ECO netlist and a spare gate mapping file. This mapping file instructs the place-and-route tool how to map the newly added ECO logic to specific spare logic resources in the layout.

- Integrated environment

  An intuitive graphical user interface (GUI) is provided for setup and debugging, allowing the user to work more productively and quickly pinpoint the cause of equivalence mismatches.

  Included are:

  - Graphical debugging through an integrated schematic viewer that shows logic values for each error vector
  - Full cross-highlighting between RTL model and circuit
  - Automatic error candidate identification with assigned and weighted percentages
  - Logic-cone pruning to focus debugging on relevant information

- Smart setup and diagnosis

  Conformal ECO Designer includes a set of intelligent `ANALYZE` commands to ease setup and diagnosis. For example, `ANALYZE SETUP` investigates the current environment and automatically remedies common setup issues sometimes experienced by new users. In tandem, `ANALYZE NONEQUIVALENT` can be invoked if non-equivalences are encountered. The command then presents a one-line answer as to what is wrong.

Conformal ECO Flows

The following lists the flows described in this document:

- Conformal Flattened ECO Flow on page 23

  This is the recommended Conformal ECO flow. In this flow, you run a flat compare on a hierarchical design. This flow can help when your hierarchical dofile is causing false non-equivalent points (due to things like clock gate cloning/decloning, inverter pushes, or boundary optimization).

- Conformal ECO for Hierarchical Files on page 27

With this flow, Conformal ECO Designer will write out as many modules as possible for hierarchical compare. Conformal ECO Designer will use the module boundaries to create as small a patch as possible.

- Pre-Mask ECO Flows on page 29

  Pre-mask ECOs are performed during place and route and before the design is taped out.

  - Pre-Mask ECO on page 30
  - Pre-Mask Physical ECO on page 33

- Post-Mask ECO Flows on page 37

  Post-mask ECOs are performed after the design has been sent to manufacturing. Once fabrication has begun, the number of gates on the die is fixed and any changes will need to be accomplished with these resources.

  - Post-Mask ECO with Standard Cells on page 39
  - Post-Mask ECO with Gate Array Cells on page 42

- Conformal ECO Power-Aware Flow on page 45

  In the Conformal ECO power-aware flow, functional ECO is performed on a design within the same domain. Conformal ECO passes power intent files to RTL compiler for optimization.

Supported Platforms

- Linux (32-bit, 64-bit)
- Sun Solaris (64-bit)
- IBM AIX (64-bit)

Language Support

- Verilog® (1995, 2001, 2005)
- SystemVerilog
- VHDL (87, 93)
- SPICE (traditional, LVS)

- EDIF
- Liberty
- Mixed languages

Starting the ECO Designer Software

You can start the Conformal ECO Designer software with an XL license or GXL license. Conformal ECO XL is primarily targeted for pre-mask ECOs and does not require physical design awareness. Conformal ECO GXL is targeted for pre and post-mask ECOs. For post-mask ECOs, spare gate physical location, quantity, type, and timing requirements must be considered as input for the tool.

- To start Conformal ECO XL, run the following command:

  ```
  > lec -eco
  ```

- To start Conformal ECO GXL, run the following command:

  ```
  > lec -ecogxl
  ```

The `lec` command has the following additional options. This list is also available using the `lec -help` command *before* you start your session.

`-32`	Runs the Encounter® Conformal® software in 32-bit mode. *This is the default.*
`-64`	Runs the Encounter® Conformal® software in 64-bit mode.
`-Dofile <filename>`	Runs the script *<filename>* after starting LEC.
`-LOGfile <filename>`	Sets up a log filed called *<filename>*.
`-Gui | -NOGui`	Starts the session in GUI or non-GUI mode.
`-TclMode`	After the session starts, the tool enters Tcl mode.
`-NOColor | -Color`	In the non-GUI mode, you can start the Conformal software with the `-color` option to specify that you want all of the messages to be color-coded. (For example, error messages appear in red text.) By default, color-coding is off in non-GUI mode.
`-DARK_Color | -LIGHT_Color`	If you are using the `-color` mode, adjusts the brightness of the colors used.

`-DEFault [init_filename] | -NODEFault]`

Specifies whether to process the initial command file (`init_filename` or `.conformal_lec`) by default during startup.

`-Banner | -NOBanner` — Specifies whether to display the LEC banner during startup.

`-RESETrc` — Reset GUI default settings.

`-NOLIcwait` — If all licenses are checked out, exit immediately.

`-Info` — Display the product information and exit.

`-Version` — Displays the product version. Once you have started your session, you can also use the `VERSION` command to display the Conformal software version number. This is useful when starting a transcript log file to ensure that the file contains a reference to the Conformal version that created the results.

`-restore_session <session_file_name>`

Restores a session that was saved using the checkpoint/restart facility (`SAVE SESSION -checkpoint` command).

1

Data Requirements

This chapter describes everything that you should prepare before starting the flows described in this book (if an item is specific to a particular flow, it will be noted as such).

Input Requirements

The following tables summarize the various input data mentioned in the Pre-Mask ECO Flows and the Post-Mask ECO Flows.

Required Data	Description	Requirement Level	Applicable Flow
G1	Original netlist (after layout)	Mandatory	Pre-Mask Post-Mask
G2	Netlist after synthesis	Mandatory	Pre-Mask Post-Mask
Liberty library	Library for RC	Mandatory	Pre-Mask Post-Mask
Verilog library	Verilog simulation library	Optional	Pre-Mask Post-Mask
RTL1	Original RTL	Recommended	Pre-Mask Post-Mask
RTL2	RTL with ECO	Recommended	Pre-Mask Post-Mask
Design constraints	DFT constraints and/or renaming rules	Recommended	Pre-Mask Post-Mask

Required Data	Description	Requirement Level	Applicable Flow
ECO Information ■ Name of modules that require an ECO (Recommended) ■ Names of ports that were added or deleted due to an ECO (Recommended) ■ Names of ports that are renamed by an ECO (Optional) ■ Names of registers that were added or renamed by an ECO (Optional)			Pre-Mask Post-Mask
Spare cell name	Names of the spare cells that you can use for ECO	Mandatory	Pre-Mask Physical Post-Mask
LEF	Contains library information for a class of designs	Mandatory	Pre-Mask Physical Post-Mask
DEF	Describes the logical and physical layout of design	Mandatory	Pre-Mask Physical Post-Mask
CapTbl	Contains parasitic resistance and capacitance information	Recommended	Pre-Mask Physical Post-Mask

OPTIMIZE PATCH File Requirements

In several ECO flows, you will read in several types of files: Synopsys Design Constraints (SDC), Library Exchange Format files (LEF), Design Exchange Format files (DEF), capacitance tables (capTbl), and timing libraries (lib) for both gate array filler cells and gate array cells. These files are read in using the `OPTIMIZE PATCH` command. For example:

```
optimize patch -workdir WORK_POST -usespare \
  -library ../../lib/liberty/FreePDK45_lib_v1.0_typical.lib \
  -sdc ../../pr/g1.pr.sdc \
  -lef ../../lib/lef/FreePDK45_lib_v1.0.lef \
  -def ../../pr/g1.pr.def \
  -captable ../../pr/g1.cap.gz \
...
```

This section provides a brief overview of those files.

■ Timing Libraries on page 17

■ SDC on page 17

- LEF on page 17
- DEF on page 19
- capTbl on page 20

Conformal ECO sends these input files to RTL Compiler

Timing Libraries

Using the `OPTIMIZE PATCH` command, you will read in the timing libraries. For the post-mask flow with gate-array cells, the libraries must contain:

- The gate array filler cells; these cells typically do not contain timing arcs or functionality.
- The function for the gate array. Conformal ECO use this to determine how to map the filler cells to the cells for the ECO changes.

Note: The tool will run a consistency check between the LEF files and the timing library files to ensure there is a one-to-one mapping between them. The technology functional libcells must be present in both the LEF and the library files. Otherwise, the tool will error out.

SDC

Synopsys Design Constraint (SDC) is a format that allows designers to utilize the same sets of constraints to drive synthesis, timing analysis, and place-and-route.

LEF

A Library Exchange Format (LEF) file contains library information for a class of designs. Library data includes layer, via, placement site type, and macro cell definitions.

You can define all of your library information in a single LEF file; however this creates a large file that can be complex and hard to manage. Instead, you should divide the information into two files:

- Technology LEF Files on page 18
- Cell Library LEF Files on page 19

General Rules

Note the following information when creating LEF files:

- Lines in the LEF file are limited to 2,048 characters (extra characters are truncated). Distance is specified in microns.
- Distance precision is controlled by the `UNITS` statement.
- LEF statements end with a semicolon (`;`). You *must* leave a space between the last character in the statement and the semicolon.
- You can specify statements in any order; however, data must be defined before it is used.

Requirements for the Post-Mask Flow

- You must read in all LEF files (technology LEF, cell library LEF, std cells, and memories)
- The LEF files must contain the defintions for gate array and gate array filler cells
- Gate array cells should not be modeled as `CLASS CORE SPACER` in the LEF file. Use `CLASS CORE` instead.
- Gate array filler cells can contain either `CLASS CORE` or `CLASS CORE SPACER`. "`SPACER`" specifies that this cell is used to fill in space between regular core cells.
- The tool will run a consistency check between the LEF files and the timing library files to ensure there is a one-to-one mapping between them. The technology functional libcells must be present in both the LEF and the library files. Otherwise, the tool will error out.

Technology LEF Files

A technology LEF file contains all of the LEF technology information for a design, such as placement and routing design rules, and process information for layers. A technology LEF file can include any of the following LEF statements:

```
[VERSION statement]
[BUSBITCHARS statement]
[DIVIDERCHAR statement]
[UNITS statement]
[MANUFACTURINGGRID statement]
[USEMINSPACING statement]
[CLEARANCEMEASURE statement ;]
[PROPERTYDEFINITIONS statement]
[LAYER (Nonrouting) statement
| LAYER (Routing) statement] ...
[SPACING statement ]
[MAXVIASTACK statement]
[VIA statement] ...
[VIARULE statement] ...
[VIARULE GENERATE statement] ...
[NONDEFAULTRULE statement] ...
[SITE statement] ...
[BEGINEXT statement] ...
[END LIBRARY]
```

Cell Library LEF Files

A cell library LEF file contains the macro and standard cell information for a design. A library LEF file can include any of the following statements:

```
[VERSION statement]
[BUSBITCHARS statement]
[DIVIDERCHAR statement]
[VIA statement] ...
[SITE statement]
[MACRO statement
[PIN statement] ...
[OBS statement ...] ] ...
[BEGINEXT statement] ...
[END LIBRARY]
```

When reading in LEF files, always read in the technology LEF file first.

DEF

The Design Exchange Format (DEF) is file format that can describe the logical and physical layout of design. It represents the netlist and circuit layout. DEF is used in conjunction with the Library Exchange Format (LEF) to represent complete physical layout of an integrated circuit while it is being designed.

General Rules

Note the following when creating DEF files.

- Lines in the DEF file are limited to 2,048 characters (extra characters are truncated on input).
- Net names and cell names also are limited to 2,048 characters.
- DEF statements end with a semicolon (`;`). You *must* leave a space before the semicolon.
- Each section can be specified only once. Sections end with `END` *`SECTION`*.
- You must define all objects before you reference them except for the `+ ORIGINAL` argument in the `NETS` section.
- No regular expressions or wildcard characters are recognized except for `(*` *`pinName`* in the `SPECIALNETS` section.

Requirements for the Post-Mask Flow

- The information about your design must be described in a single DEF file. Multiple DEF files are not supported in this flow.
- The gate array filler cells must already have been placed in the DEF file.
- The placement attribute for these cells should be `FIXED` attribute in the DEF file.

Order of Statements

Standard DEF files can contain the following statements and sections. You can define the statements and sections in any order; however, data must be defined before it is used.

```
[ VERSION statement ]
[ DIVIDERCHAR statement ]
[ BUSBITCHARS statement ]
DESIGN statement
[ TECHNOLOGY statement ]
[ UNITS statement ]
[ HISTORY statement ] ...
[ PROPERTYDEFINITIONS section ]
[ DIEAREA statement ]
[ ROWS statement ] ...
[ TRACKS statement ] ...
[ GCELLGRID statement ] ...
[ VIAS statement ]
[ STYLES statement ]
[ NONDEFAULTRULES statement ]
[ REGIONS statement ]
[ COMPONENTS section ]
[ PINS section ]
[ PINPROPERTIES section ]
[ BLOCKAGES section ]
[ SLOTS section ]
[ FILLS section ]
[ SPECIALNETS section ]
[ NETS section ]
[ SCANCHAINS section ]
[ GROUPS section ]
[ BEGINEXT section ] ...
END DESIGN statement
```

capTbl

Although the DEF file can provide data for parasitic extraction, the parasitic resistance and capacitance information defined in the capacitance table is very detailed and provides a much better approximation of the actual parasitics.

The sections of the capacitance table are used by Encounter RTL Compiler:

- `ShrinkFactor`

- PROCESS_VARIATION
- BASIC_CAP_TABLE

Note: You should use the same process corner capTbl file as used for the timing libraries that are read in with the OPTIMIZE PATCH command.

2

Conformal Flattened ECO Flow

This section describes the Flattened ECO Flow (FEF). This flow is the recommended Conformal ECO flow.

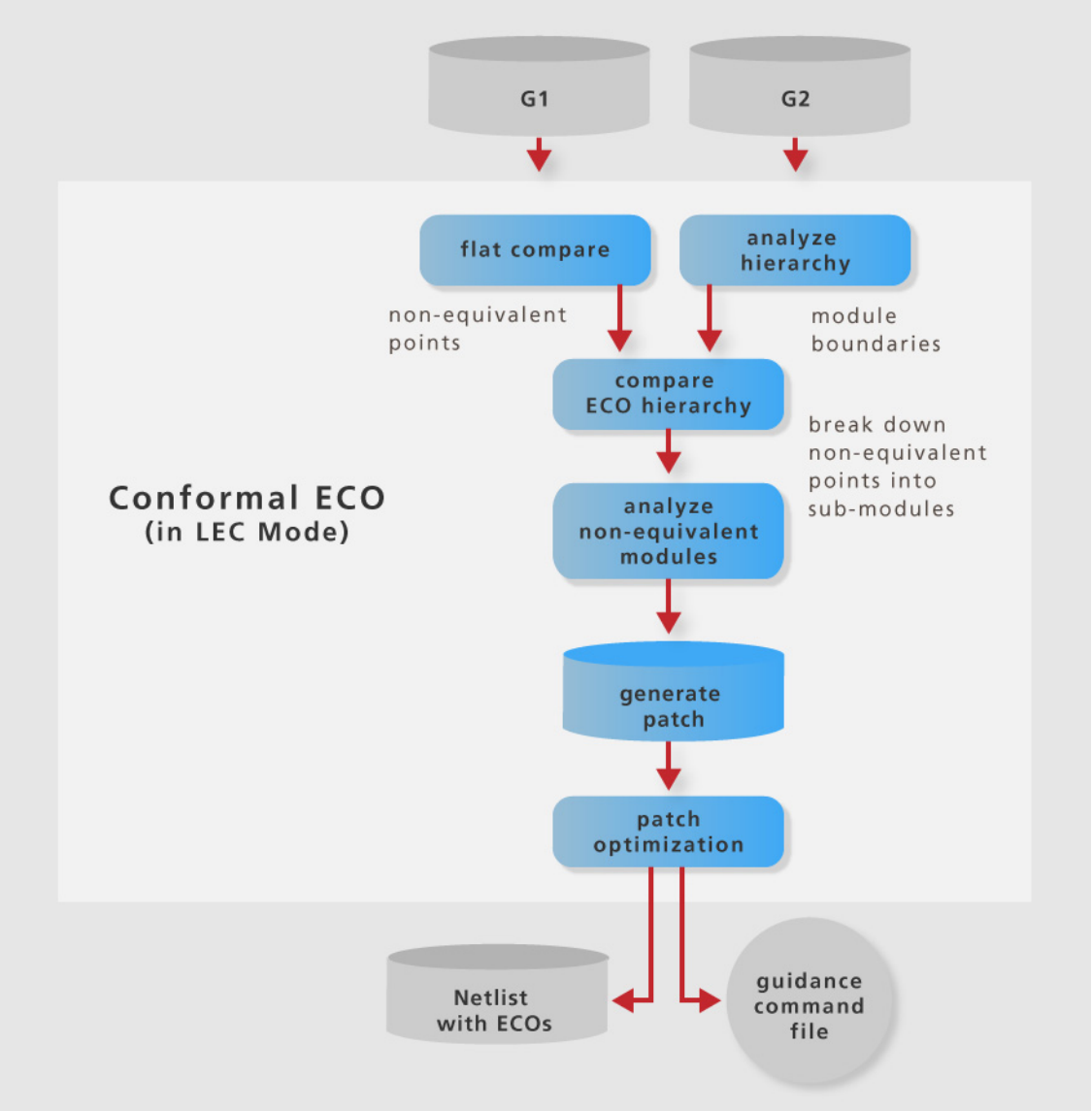
In this flow, you run a flat compare on a hierarchical design. This flow can help when your hierarchical dofile is causing false non-equivalent points (due to things like clock gate cloning/decloning, inverter pushes, or boundary optimization).

This flow is not ideal for flat designs or when you have clean hierarchical dofiles (for example, two dofiles with ECO modules are compared and their NEQs match those reported by `ANALYZE HIER_COMPARE`).

This flow has the following benefits:

- Focuses ECO analysis on the non-equivalent key points identified from a flat compare
- Supports hierarchical clock gating that is not a part of an ECO. For example, if you have two hierarchical gates that do not match, you can perform a flat compare, but create a patch to the hierarchical netlist.
- Works with the current hierarchical flow. You can also run a combination of the hierarchical ECO flow and the FEF flow (running FEF on select modules).
- On-demand insertion of ECO pin/ports through an ECO pin dofile.
- Easier set up. All ECO patches can be created with one command. Running a hierarchical dofile is not required.

Flow Diagram



LEGEND:
G1 Original netlist
G2 Synthesis netlist from modified RTL

Flow Steps

Once you enter LEC mode, a typical scenario for a flattened ECO flow involves the following steps:

1. Use the `SET ECO OPTION -flat` command (in Setup mode) to specify that your intention is to run the FEF flow. Specifying this command 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
   ```

   Note: When running a flattened ECO flow, set the `-threshold` of `ANALYZE HIER_COMPARE` to 0, so that the tool can determine all of the module boundaries.

2. Add any additional options to the `ANALYZE HIER_COMPARE` command,

3. Use the `ADD COMPARE POINTS -all` and `COMPARE` commands to start the comparison between the flattened designs. This will determine the non-equivalent points.

4. Use the `COMPARE ECO HIERARCHY` command to break down the non-equivalent points (determined in step 2) into their sub-modules, based on the module boundaries (determined by step 1).

5. Use the `ANALYZE ECO -hierarchical` command to create all the necessary patches.

   During this step, you can also create a dofile that creates only ECO pins that are part of the non-equivalent cones (through the `-ecopin_dofile` option).

6. Apply and optimize the patch using the `APPLY PATCH` and `OPTIMIZE PATCH` commands, respectively.

Sample Dofiles

For a sample dofile that illustrates this flow, refer to the “Sample Dofiles” section on the web interface.

To launch the web interface, use the `set web on` command. This command outputs a URL, which you can paste into any of the following web browsers: Internet Explorer 9, Firefox 4, and Chrome 10. The web interface contains the documentation and FAQs for 12.1 features.

3

Conformal ECO for Hierarchical Files

With this flow, Conformal ECO Designer writes out as many modules as possible for hierarchical compare. With this approach, Conformal ECO Designer will use the module boundaries to create as small a patch as possible.

The LEC "`write hier_compare dofile -balanced_extractions`" command 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). This command is recommended when one of the netlists has gone through back-end synthesis, such as clock-tree-synthesis (CTS), resulting in an inversion push across submodule boundaries, extra ports, or ports with mismatched names.

For a full sample dofile that illustrates this flow, refer to the “Sample Dofiles” section on the web interface.

To launch the web interface, use the `set web on` command. This command outputs a URL, which you can paste into any of the following web browsers: Internet Explorer 9, Firefox 4, and Chrome 10. The web interface contains the documentation and FAQs for 12.1 features.

4

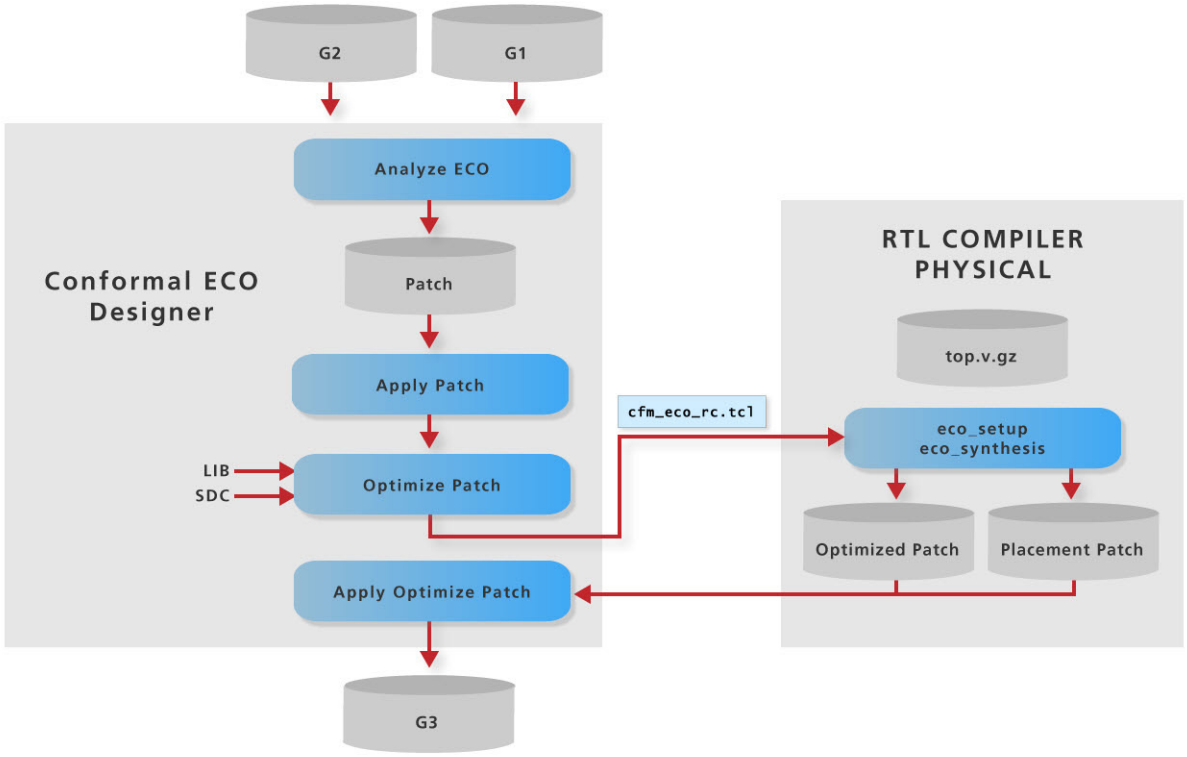
Pre-Mask ECO Flows

This section describes how to perform pre-mask ECO optimization using Conformal ECO.

In the flows described in this chapter, you will read in several types of data and files. These requirements are described in Chapter 1, “Data Requirements”.

Pre-Mask ECO

Pre-mask ECOs are performed during place and route and before the design is taped out.



LEGEND:

G1	Original netlist
G2	Synthesis netlist from modified RTL
G3	Optimized Verilog netlist after ECO
LIB	Timing library
SDC	SDC file
cfm_eco_rc.tcl	Tcl script to map and

Steps for Implementing a Pre-Mask ECO

A typical scenario for a pre-mask ECO flow involves the following steps:

The following steps outline a typical situation:

1. Compare the old RTL to the old netlist.

   This comparison should be equivalent. You might need to disable the scan by adding scan constraints.

2. Synthesize the new RTL.

   If possible, use the same synthesis tool, version, and script that was used to create the old netlist. Minimize any changes. Synthesizing the new RTL is required to increase the probability of the ECO meeting timing.

3. Compare the new RTL to the new netlist.

   This comparison should be equivalent.

4. Compare the old RTL to the new RTL.

   This comparison should be nonequivalent.

   a. Ensure to set the X conversion to ‘Don’t Care’ for the new RTL and ‘E’ for the old RTL. This is the default setting if you read the new RTL as the Golden design and the old RTL as the Revised design.

   b. Run both flat and hierarchical comparison; do not run a dynamic, hierarchical comparison. Note the nonequivalent modules. You will specify these as ECO modules in step 5 using the `ADD MODULE ATTRIBUTE -eco_module` command.

   c. Run a UNIX `diff` on the RTL files to see exactly what was changed. The comparison results and the `diff` results should be consistent.

5. Compare the old netlist to the new netlist.

   a. Run hierarchical compare on the old netlist and the new netlist; compare as many modules as possible. Again, do not use dynamic hierarchical compare. You will most likely need to use the same scan constraints used in step 1.

   b. Compare the results from 5.a. to the results from step 4.b. The nonequivalent modules should be consistent with the comparison results and `diff` results should be consistent with that of step 4.

6. Create ECO patch files using the old netlist and the new netlist.

After verifying that all the nonequivalent points in step 5 are caused by the change in functionality, you can use the Conformal software to create a patch file for each nonequivalent module. The patch contains the changes that will transform the functionality of the design.

Important

Any `ADD ECO PIN` commands used during this step must also be used for steps 7 and 8.If there is any flattening done during the creation of the ECO patch file, you must perform the same flattening for steps 7 and 8.

See "Creating ECO Patch Files" on page 49 for more information.

7. Apply patch and write out ECO netlist.

You can use a patch file to create the ECO netlist after analyzing all of the ECO modules, then write out a netlist that has not been mapped or optimized. See "Applying the ECO Patch and Writing Out the ECO Netlist" on page 52 for more information.

8. Use the `OPTIMIZE PATCH` command to:

- Read in SDC, QRC technology file and LEF. For more information on these formats, see "OPTIMIZE PATCH File Requirements" on page 16.
- Map and optimize ECO patch file.

  Use the Conformal software to map and optimize the patch file using Genus technology. The software then writes out an Genus script in the working directory that will optimize the patches. See "Mapping and Optimizing the ECO Patch File" on page 53 for more information.

9. Check the ECO netlist.

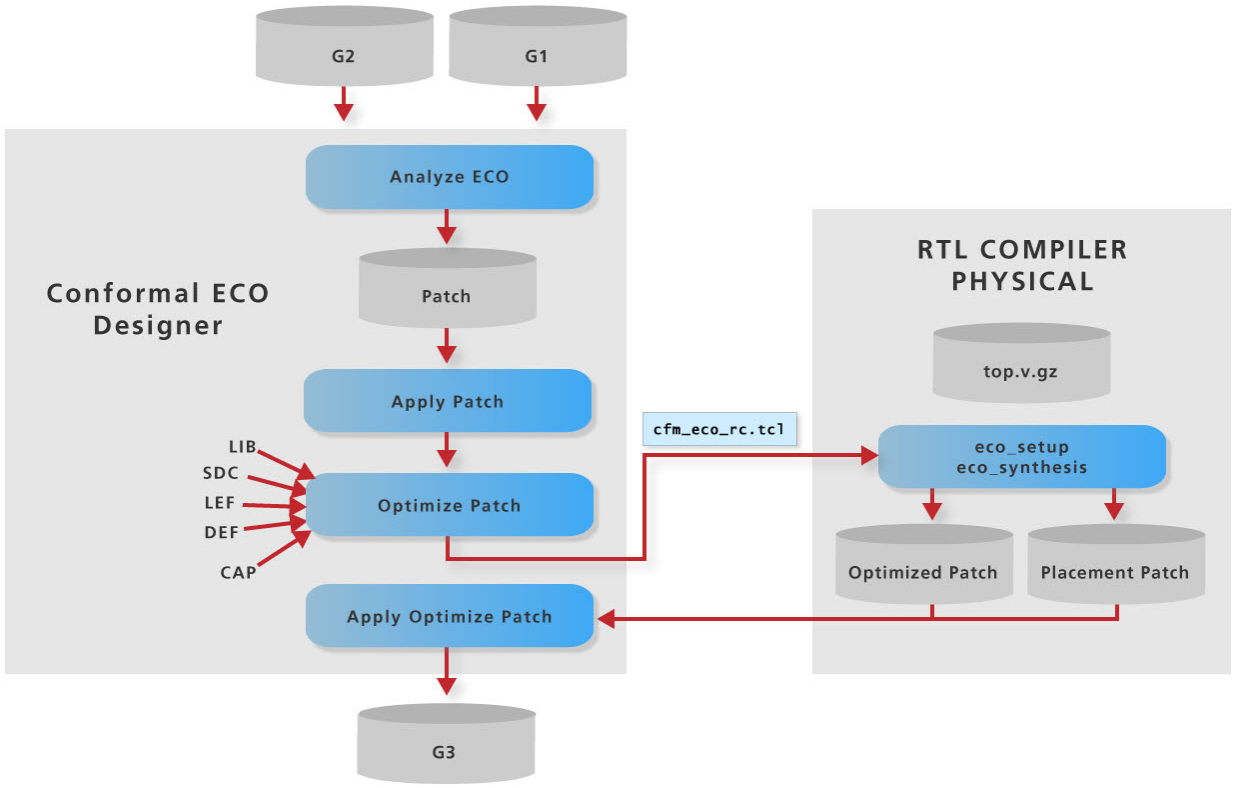
You should run equivalency checking on the ECO netlist versus the G2 netlist. After that, the netlist can be read into the Innovus tool for incremental optimization and place and route. See "Checking the ECO Netlist" on page 55 for more information.

Pre-Mask Physical ECO

This feature requires a Conformal ECO GXL license.

In the following diagram, Conformal ECO generates an optimized Verilog G3 netlist and an instance placement file (in Tcl/text format).

Figure 4-1 Generating an Instance Placement File (in Tcl/Text Format)

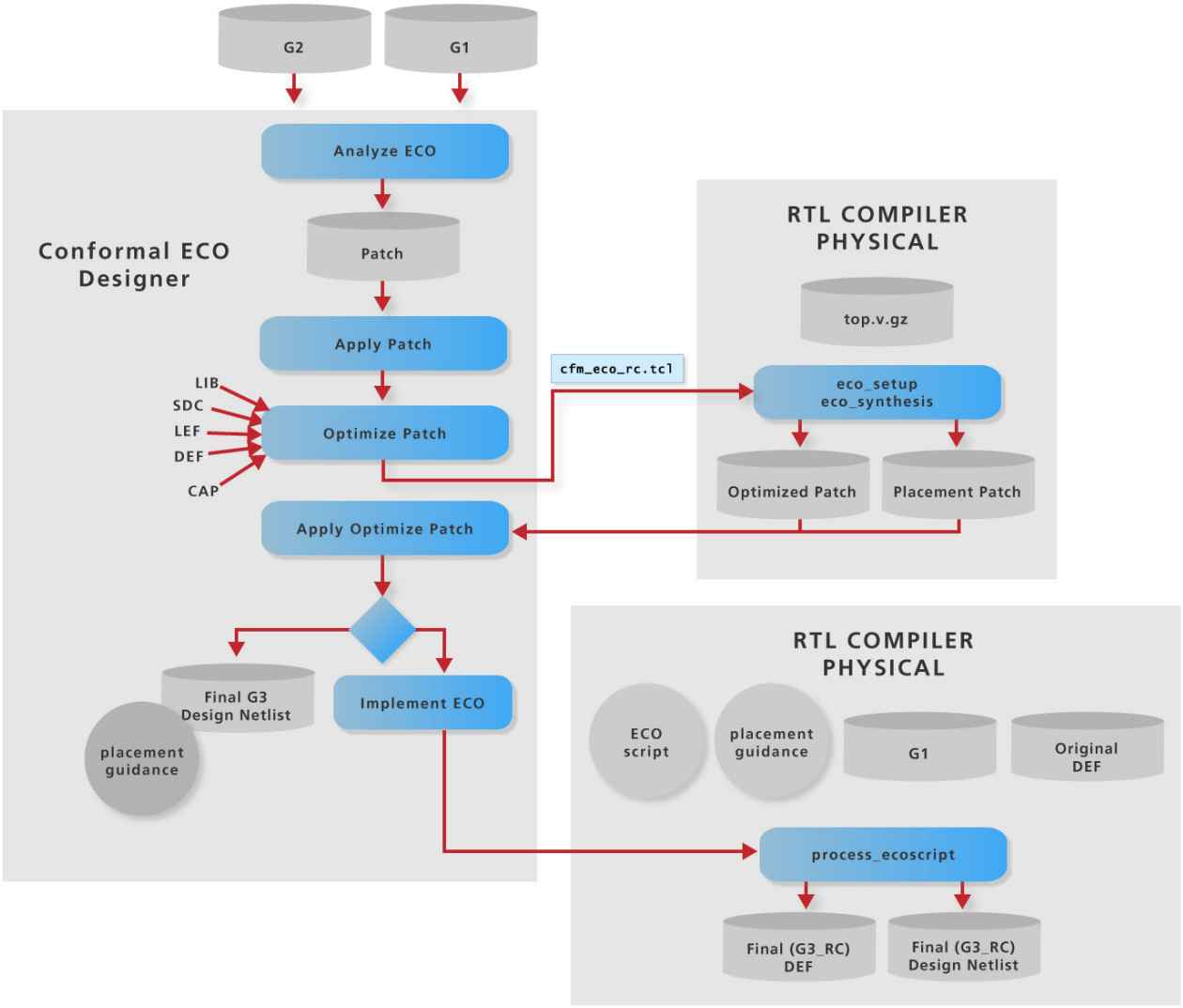


LEGEND:

G1	Original Netlist
G2	Netlist after ECO
G3	Optimized Verilog netlist
LIB	Timing library
SDC	SDC file
LEF	Library exchange format file
DEF	Design exchange format file
QRC	QRC technology file
cfm_eco_rc.tcl	Instance placement file

In the following diagram, Conformal ECO outputs either a DEF or placement guidance file.

Figure 4-2 Generating a DEF or Placement Guidance File



LEGEND:

G1	Original netlist
G2	Synthesis netlist from modified RTL
G3	Optimized Verilog netlist after ECO
LIB	Timing library
SDC	SDC file
LEF	Library exchange format file
DEF	Design exchange format file
QRC	QRC technology file
cfm_eco_rc.tcl	Tcl script to map and optimize the patch

Steps for Implementing a Physically-Aware, Pre-Mask ECO

The following outlines the typical scenario for implementing physically-aware, pre-mask ECOs:

1. Hierarchically compare the original netlist with the modified netlist. This step involves using the following commands:

   - `WRITE HIER_COMPARE DOFILE -balanced_extraction` (suggested)—Extracts a balanced set of hierarchical constraints by simultaneously using flattened golden and flattened revised designs.
   - `RUN HIER_COMPARE -nodynamic`—Runs static hierarchical comparison without auto-flattening the submodules.

2. Create ECO patch files using the original netlist (G1) and the netlist after ECO (G2).

   After verifying that all the nonequivalent points in the previous step are caused by the change in functionality, you can use the Conformal software to create a patch file for each nonequivalent module. The patch contains the changes that will transform the functionality of the design.

   See Creating ECO Patch Files on page 49 for more information.

3. Apply the patch and write out the ECO netlist (G3).

   You can use a patch file to create the ECO netlist after analyzing all of the ECO modules, then write out a netlist that has not been mapped or optimized. See Applying the ECO Patch and Writing Out the ECO Netlist on page 52 for more information.

4. Use the `OPTIMIZE PATCH` command to:

   - Read in SDC, timing libraries, QRC technology file, LEF, and DEF files. For more information on these formats, see "OPTIMIZE PATCH File Requirements" on page 16.
   - Map and optimize ECO patch file.

     Use the Conformal software to map and optimize the patch file using Genus technology. The software then writes out an Genus script in the working directory that will optimize the patches. See Mapping and Optimizing the ECO Patch File on page 53 for more information.
   - Specify that this optimization is for physically-aware, premask ECOs (`-placescript` option). This option also writes out the placement script (in Innovus Tcl script format) in the working directory that will optimize the patches and execute the script.

5. For each ECO module in the list, Genus Physical will write out the placement coordinates of each ECO cell instance in the patch. Conformal ECO will read back the optimized patches and placement files and write out the Verilog G3 netlist and the consolidated placement guidance file (in Tcl/text format).

   If you want to output a DEF placement guidance file (instead of Tcl/text format), use the `WRITE ECO DEF` command to generate the DEF (after the `OPTIMIZE PATCH` command).

Sample Dofiles

For a sample dofile that illustrates this flow, refer to the “Sample Dofiles” section on the web interface.

To launch the web interface, use the `“set web on”` command. This command outputs a URL, which you can paste into any of the following web browsers: Internet Explorer 9, Firefox 4, and Chrome 10. The web interface contains the documentation and FAQs for 12.1 features.

5

Post-Mask ECO Flows

Overview

Post-mask ECO uses two types of spare logic resources:

- Post-Mask ECO with Standard Cells—Compact and run faster than gate-array cells. However, their logic type and location is fixed.
- Post-Mask ECO with Gate Array Cells—Offers more flexibility through different configurations. However, they are larger and slower than standard cells.

You can also have a combination of standard and gate array cells.

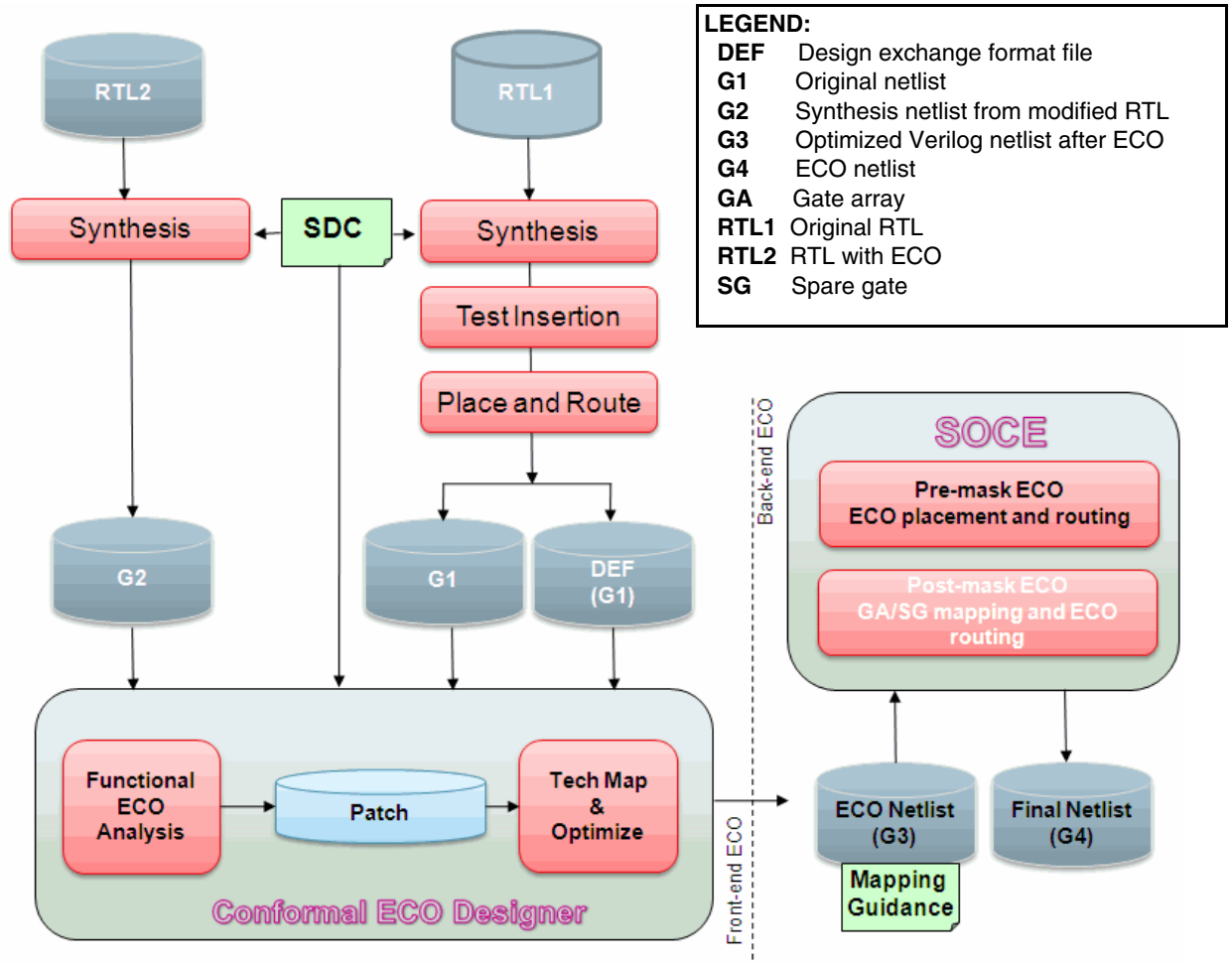
Input Requirements

In the flows described in this chapter, you will read in several types of input data and files. These requirements are described in Chapter 1, “Data Requirements”.

Post-Mask ECO with Standard Cells

This section describes the Encounter Conformal ECO Designer flow for a post-mask ECO with standard cells.

In the following diagram, the design has completed place-and-route. If the functionality needs to be changed (either for a bug or additional functionality), new RTL is written and the new functionality needs to be added to the placed and routed design. The new RTL is synthesized and this new netlist and the old netlist are processed by the Conformal software to create the ECO netlist. The ECO netlist is based on the old netlist, yet has the functionality of the new netlist.



For a post-mask ECO, you must read the original physical database and parasitic files into the software, which will optimize the spare cell mapping based on the available spare cells. If there are very few spare resources, or the ECO change is very large, there might not be

enough resources to implement the ECO. Conformal ECO helps to enable early predictability into the post-mask ECO process.

Steps for Implementing a Post-Mask ECO with Standard Cells

The following steps outline a typical situation:

1. Compare the old RTL to the old netlist.

   This comparison should be equivalent. You might need to disable the scan by adding scan constraints.

2. Synthesize the new RTL.

   If possible, use the same synthesis tool, version, and script that was used to create the old netlist. Minimize any changes. Synthesizing the new RTL is required to increase the probability of the ECO meeting timing.

3. Compare the new RTL to the new netlist.

   This comparison should be equivalent.

4. Compare the old RTL to the new RTL.

   This comparison should be non equivalent. Make sure to set the X conversion to ‘Don’t Care’ for the new RTL and ‘E’ for the old RTL. This is the default setting if you read the new RTL as the Golden design and the old RTL as the Revised design. Run both flat and hierarchical comparison. Do not use dynamic hierarchical compare. Note exactly which modules and registers are affected.

   Run a UNIX `diff` on the RTL files to see exactly what was changed. The comparison results and the `diff` results should be consistent.

5. Compare the old netlist to the new netlist.

   Run hierarchical compare on the old netlist and the new netlist. Try to get hierarchical comparison to compare as many modules as possible. Again, do not use dynamic hierarchical compare. You will probably have to use the same scan constraints used in step 1. Compare these results to the information from step 4. The nonequivalent modules should be consistent with the comparison results and `diff` results in step 4.

6. Create ECO patch files using the old netlist and the new netlist.

   After verifying that all the non equivalent points in step 5 are caused by the change in functionality, you can use the Conformal software to create a patch file for each nonequivalent module. The patch contains the changes that will transform the functionality of the design.

Important

Any `ADD ECO PIN` commands used during this step must also be used for steps 7 and 8.If there is any flattening done during the creation of the ECO patch file, you must perform the same flattening for steps 7 and 8.

See "Creating ECO Patch Files" on page 49 for more information.

7. Add spare cells.

   Use the `ADD SPARE CELL` command to add spare or freed cells as the available cells for the `OPTIMIZE PATCH` command.

   Note: For a post-mask ECO, each ECO patch should have at least one spare tie cell for each phase. For example, a hierarchical design with three ECO modules (resulting in three patches) would need three tie-hi and three tie-low spare cells.

8. Apply patch and write out ECO netlist.

   You can use a patch file to create the ECO netlist after analyzing all of the ECO modules, then write out a netlist that has not been mapped or optimized. See "Applying the ECO Patch and Writing Out the ECO Netlist" on page 52 for more information.

9. Use the `OPTIMIZE PATCH` command to:

   - Read in SDC, capacitance tables and LEF. For more information on these formats, see "OPTIMIZE PATCH File Requirements" on page 16.
   - Map and optimize ECO patch file.

     Use the Conformal software to map and optimize the patch file using Encounter RTL Compiler technology. The software then writes out an RTL Compiler script in the working directory that will optimize the patches. See "Mapping and Optimizing the ECO Patch File" on page 53 for more information.

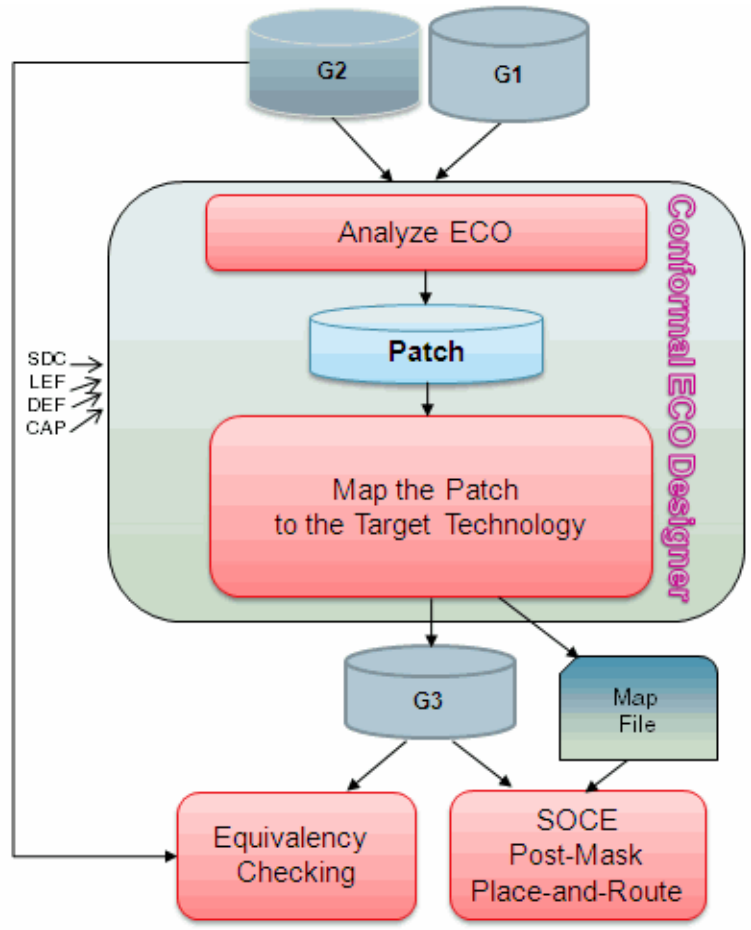
10. Check the ECO netlist.

    You should run equivalency checking on the ECO netlist versus the G2 netlist. After that, the netlist can be read into the SoC Encounter tool for incremental optimization and place and route. See "Checking the ECO Netlist" on page 55 for more information.

Post-Mask ECO with Gate Array Cells

Gate array cells are special cells reserved for ECO changes only. They are added and scattered throughout the design as gate array filler cells, so when there is a need for ECO, these cells can be configured into gate array standard cells (combinational and/or sequential depending on the technology) using top metal changes.

The following diagram illustrates the Encounter Conformal ECO Designer flow for a post-mask ECO with gate array cells.



LEGEND:

G1	Original netlist
G2	Synthesis netlist from modified RTL
G3	Optimized Verilog netlist after ECO
SDC	SDC file
LEF	Library exchange format file
DEF	Design exchange format file
CAP	Capacitance table

In the diagram, the design has completed steps 1-5 of the flow described in "Post-Mask ECO with Standard Cells" on page 39.

Steps for Implementing a Post-Mask ECO with Gate Array Cells

After verifying that all the non equivalent points are caused by the change in functionality, you can use the Conformal software to create a patch file for each nonequivalent module. The patch contains the changes that will transform the functionality of the design.

1. Create ECO patch files using the old netlist and the new netlist.

   After verifying that all the non equivalent points in step 5 are caused by the change in functionality, you can use the Conformal software to create a patch file for each nonequivalent module. The patch contains the changes that will transform the functionality of the design.

2. Add spare cells.

   Use the `ADD SPARE CELL` command to add spare, freed, or gate array filler cells as the available cells for the `OPTIMIZE PATCH` command.

   Use the `-GAlibcell` option of the `OPTIMIZE PATCH` command to specify the gate array library cell instance names.

3. Use the `OPTIMIZE PATCH` command to:

   - Read in SDC, capacitance tables and LEF. For more information on these formats, see "OPTIMIZE PATCH File Requirements" on page 16.
   - Map and optimize ECO patch file.

     Use the Conformal software to map the patch file to the target technology cells using the restricted set specified in step 2. The software then writes out an RTL Compiler script in the working directory that will optimize the patches.

     The tool then applies the mapped patch file to the old gate netlist (G1) and writes out an ECO netlist (G3) and map file.

     See Mapping and Optimizing the ECO Patch File on page 53 for more information.

4. Check the ECO netlist (G3).

   You should run equivalency checking on the ECO netlist versus the G2 netlist. After that, the netlist can be read into the SoC Encounter tool for incremental optimization and place and route.

Sample Dofiles

For a sample dofile that illustrates this flow, refer to the “Sample Dofiles” section on the web interface.

To launch the web interface, use the `“set web on”` command. This command outputs a URL, which you can paste into any of the following web browsers: Internet Explorer 9, Firefox 4, and Chrome 10. The web interface contains the documentation and FAQs for 12.1 features.

6

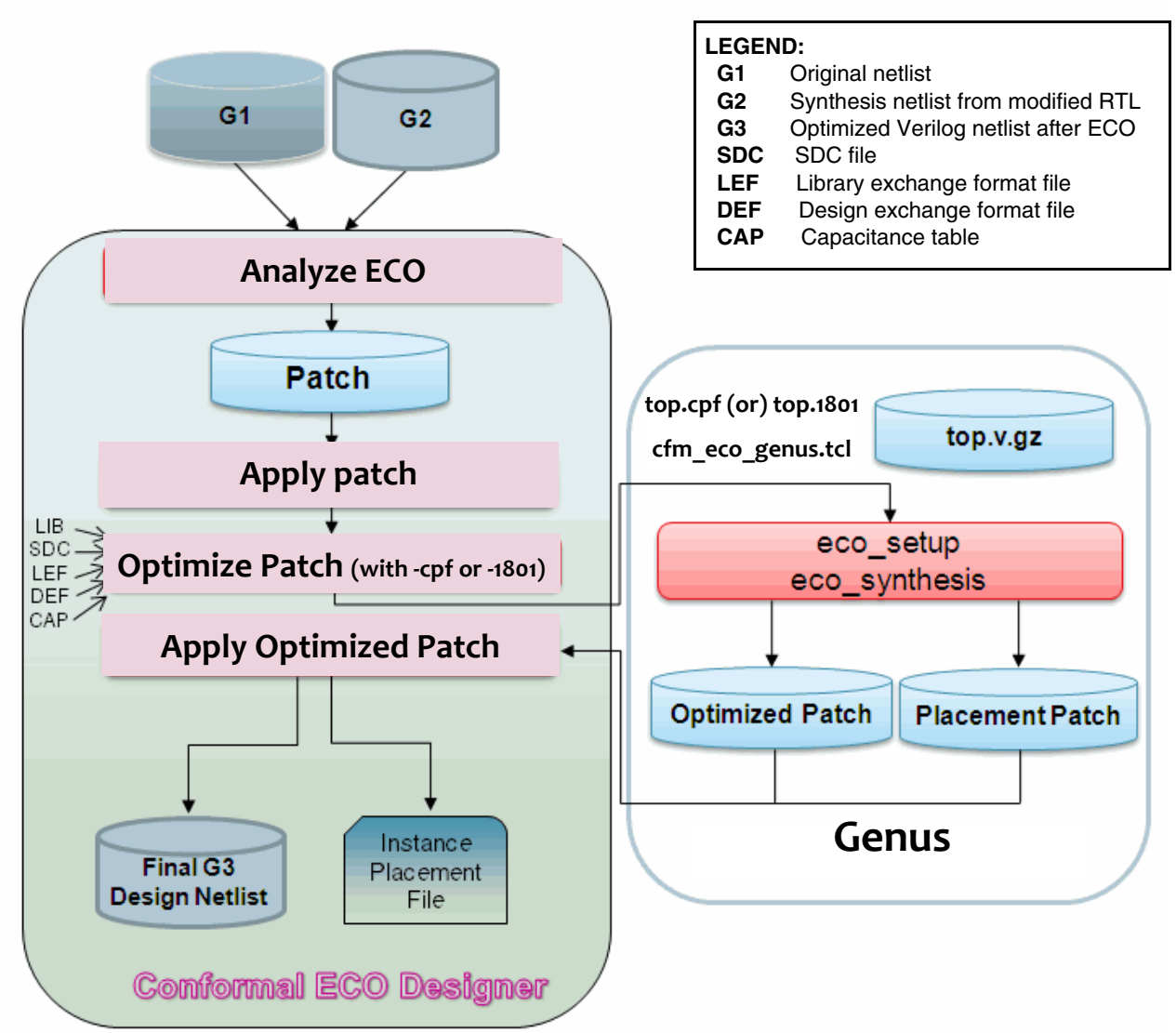
Conformal ECO Power-Aware Flow

This feature requires a Conformal ECO GXL license.

In the Conformal ECO power-aware flow, functional ECO is performed on a design within the same domain. Conformal ECO passes power intent files to Genus for optimization (using the `-cpf` or `-1801` option of `OPTIMIZE PATCH`). ECOs can be analyzed in flattened or hierarchical comparisons, as long as the domain boundaries are not modified. You can perform pre- or post-mask mapping of library cells based on the power domains specified in the power intent file. Note the following limitations:

- Combinational ECO must be fully contained within a power domain
- Sequential ECO must be fully contained within a power domain that cannot remap to a detention DFF
- The CPF or 1801 specification must remain unchanged

Figure 6-1 Power Aware Flow



The pre-mask and post-mask physical flows require input files (LIB, SDC, LEF, DEF, and CAP) described in “OPTIMIZE PATCH File Requirements” on page 16. The pre-mask, non-physical flow does not require these files.

The following illustrates the command sequence for the power aware pre-mask flow:

Conformal ECO

```
# Import the design
read lib  lib.v
read design -gol  g1.v
read design -rev  g2.v

## Create the patch
analyze eco patch.v
apply patch -auto

### Optimize the patch in Genus
with CPF or 1801
optimize patch [-cpf <cpf_file> |
-1801 <1801_file>]

report eco changes
write eco design
```

Genus (cfm_eco_genus.tcl)

```
set_attribute use_tiehilo_for_const unique /

eco_setup non_physical \
  -degenerate_combo_elts_to_basic \
  -read_nl_elab \
  -library { } \
  -sdc ../ivs_pr_power_cts_wc.sdc \
  -cpf ../ivs_pr_power_L.cpf \
  or
  -1801 ../ivs_pr_power_L.upf \
  -top_mod ivs_pr_power \
  -netlist top.v.gz
  eco_synthesis pre_mask \
  -top_mod ivs_pr_power \
  -effort high \
  -eco_module_names { …….}
quit
```

The following illustrates the command sequence for the power aware post-mask flow:

Conformal ECO

```
# Import the design
read lib  lib.v
read design -gol  g1.v
read design -rev  g2.v

## Create the patch
analyze eco patch.v
apply patch -auto
add spare cell …

### Optimize the patch in Genus
with CPF or 1801
optimize patch [-cpf <cpf_file> |
-1801 <1801_file>]

Report eco changes

Write eco design
```

Genus Physical (cfm_eco_genus.tcl)

```
set_attribute use_tiehilo_for_const unique /

eco_setup physical \
  -degenerate_combo_elts_to_basic \
  -cpf ../../cpf/top.cpf \
  or
  -1801 ../../upf/top.upf \
  -def ../../pr/g1.pr.def \
  -lef_library {../../lib/lef \
     FreePDK45_lib_v1.0.lef} \
  -top_mod vga_enh_top \
  -netlist top.v.gz
  eco_synthesis post_mask_physical \
  -top_mod vga_enh_top \
  -effort high\
  -eco_module_names \
    {{/vga_enh_top/vga_vtim_eco}}\
  -sp_inst_list { ……..}
quit
```

7

ECO Patch Files

Creating ECO Patch Files

Use the ANALYZE ECO command to create a patch file for each nonequivalent module. The patch file defines a patch module and contains the changes that will transform the functionality of the design. The patch module name is the original nonequivalent module name with an _eco appended.

The following is an example script which first compares the old RTL to the new RTL, and then compares the old netlist to the new netlist prior to running ANALYZE ECO:

```
//Script 1: Compares old RTL to new RTL
set log file r1r2.log -replace
usage -auto
read library typical.lib -liberty
read design rtl1.v  -golden //  RTL1
read design rtl2.v -revised // RTL2
report design data
write hier dofile hier_R1R2.do -constraint -noexact -usage -replace \
-balanced_extractions -function_pin_mapping -input_output_pin_equivalence \
-threshold 0 \
-prepend_string "analyze datapath  -verbose"
run hier hier_R1R2.do -nodynamic

// Script 2: Compares old netlist to new netlist
set log file g1g2_eco.log  -rep
usage -auto
read library typical.lib -liberty
```

```
read design top.pr.gv -golden // Old netlist
read design top.1.gv -revised // New netlist
set flatten model -eco
set flatten model -gated_clock
add pin constraint 0 scan_se -golden
add ignored output scan_out_0 scan_out_1 -golden
uniquify -all -nolibrary -revised
add module attribute mod* -eco_module -noflatten -both
write hier dofile  analyze.do -constraints -noexact_pin_match \
-balanced_extractions -function_pin_mapping -input_output_pin_equivalence \
-eco_aware \
-threshold 0 \
-ecopin_dofile ecopins.do -replace
dofile ecopins.do
run hier_compare analyze.do -nodynamic
// modules mod1 and mod2 are affected by the ECO
set root module mod1 -both
set system mode lec
add compare point -all
compare
analyze eco patch1.v -preserve_clock -replace
set system mode setup
set root module mod2 -both
set system mode lec
add compare point -all
compare
analyze eco patch2.v -preserve_clock -replace
```

Adding Input and Output Ports

If the ECO changes require new module ports, use the ADD ECO PIN command. The following example adds several input and output ports to a module called `colproc`:

```
add eco pin colproc clut_req addr_1 -golden -input
add eco pin colproc xcnt ctrl_sync -golden -output
```

Important

When using `ADD ECO PIN`, make sure you run it before hierarchical compare.

Using `ADD ECO PIN` is not necessary for submodules if you generate the patch with the flat methodology.

There are situations where you might not want to perform the ECO with a hierarchical methodology; for instance, where the old netlist is flat. An example of the flat methodology is as follows:

```
set log file g1g2.eco.log -replace
usage -auto
read library typical.lib -liberty
read design top.pr.gv -golden // Old netlist
read design top.1.gv -revised // New netlist
report design data
report black box
set flatten model -gated_clock
set flatten model -eco
add pin constraint 0 scan_se -golden
add ignored output scan_out_0 scan_out_1 -golden
set system mode lec
add compare point -all
compare
analyze eco patch.v -preserve_clock -replace
```

Automatically Adding and Deleting ECO Pins

To automatically add and/or delete ECO pins by executing either of the following commands. They both write out a dofile for adding/deleting ECO pins to the golden design as compared to the revised design.

- `write hier_compare dofile -ecopin_dofile <`*`filename`*`>`

  For example:

  ```
  write hier_compare dofile hier.do -replace \
  -constraint -noexact_pin_match \
  -balanced_extractions \
  -function_pin_mapping  \
  -input_output_pin_equivalence \
  -eco_aware \
  -threshold 0 \
  -ecopin_dofile ecopins.do  \
  -verbose
  ```

  This is the recommended command. The `-constraint` and `-noexact_pin_match` command options are required, because you cannot delete pins in the middle of a bus and the added cell must be immediately adjacent to the original bus.

- `analyze hier_compare -ecopin_dofile <filename>`

  With this command, you do not need to go back to LEC mode to execute `analyze_hier.do`. For example:

  ```
  analyze hier_compare -dofile hier_analyze.do -replace \
  -constraints \
  -noexact_pin_match \
  -function_pin_mapping  \
  -input_output_pin_equivalence \
  -eco_aware \
  -threshold 0 \
  -ecopin_dofile ecopins.do \
  ```

  The `-constraint` and `-noexact_pin_match` command options are required, because you cannot delete pins in the middle of a bus and the added cell must be immediately adjacent to the original bus.

Applying the ECO Patch and Writing Out the ECO Netlist

You can use a patch file to create the ECO netlist after analyzing all of the ECO modules (in the following example, `mod1` and `mod2`). Go back into Setup mode and set the root module back to the top-level module, delete all of the blackboxes from the hierarchical compare, then apply the patches. Running the `APPLY PATCH` command changes the design based on the patch module.

```
set system mode setup
set root module top -both
delete black box -all -hier -both
apply patch -keephierarchy -golden -auto
write design top.eco.gv -replace -auto
```

At this point you can also write out a netlist that has not been mapped or optimized.

Notes:

- The changes made by the `APPLY PATCH` command are not reflected in the schematic of the hierarchical design.
- The `APPLY PATCH` command's `-auto` option automatically reads in and applies all patches that were created with the `ANALYZE ECO` command in the current session.

Mapping and Optimizing the ECO Patch File

The patch files created by the ANALYZE ECO command contains logic that is unoptimized and might contain unmapped logic cells.

The Conformal software can map and optimize the patch file using Encounter RTL Compiler technology. Make sure that the proper RTL Compiler version (10.1 or later) is installed and that it is the first rc command in the search path. If you want to override the search path, you can specify the exact rc command.

Pre-mask Flow Script

The OPTIMIZE PATCH command writes out an RTL Compiler script (cfm_eco_rc.tcl) in the working directory that will optimize the patches and automatically executes the RC script. You must run the APPLY PATCH -keephierarchy command before running OPTIMIZE PATCH. 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. In most pre-mask cases, you can check the ECO netlist at this point (see Checking the ECO Netlist on page 55 for more information).

```
apply patch -keephierarchy -auto
optimize patch -workdir WORK_PRE \
-library ../../lib/liberty/FreePDK45_lib_v1.0_typical.lib \
-sdc ../../syn1/g1.rc.sdc \
-INStancenaming "ECO2inst_%d" -NETnaming "ECO2net_%d" -SEQuentialnaming \
"ECO2reg_%s" \
-rcexec "rc -nogui" \
-verbose
// Write out the G3 ECO Netlist
write eco design -newfile %s.pre.G3 -replace -report ECOprelogics.rpt
```

You can retain all freed instances in the original netlist and apply a value of X (unconnected), 0 (tie low), or 1 (tie high) to input pins of any freed instance with the APPLY PATCH command's -keepfreed, -tiefreed0, and -tiefreed1 options, respectively.

Note: The WRITE ECO DESIGN command attempts to minimize the text differences in the netlist that is written out. This command will not work if the FLATTEN command was previously run against G1; in this case, use WRITE DESIGN to write out the G3 netlist.

To customize the RC script or map the patch with some other process, you must apply the patch and write out the design (see Applying the ECO Patch and Writing Out the ECO Netlist on page 52).

Post-mask Flow Script

The following shows an example of a post-mask flow script where Conformal ECO does physically-aware spare-gate mappin

g. This feature requires an ECO-GXL license. A DEF file corresponding to the old netlist is necessary for the post-mask flow to derive spare gate physical information. You can read in other physical information (LEF and CAP tables) to help the physically aware mapping. You can also read in the SDC (Design Constraints) to derive timing constraints for each module.

```
apply patch -keephierarchy -auto
```

Use the `ADD SPARE CELL` command to specify the DEF file and spare cells.

```
add spare cell -def ../top.pr.def -spare spare_*/spr_gate*
add spare cell -freedcell
report spare cell
```

Use the `OPTIMIZE PATCH` command's `-mapscript` option to write out the location-aware spare-gate mapping result in the form of a TCL script that you can use with your back-end tool.

```
optimize patch -workdir WORK_POST -usespare \
  -library ../lib/typical.lib \
  -sdc ../sdc/top.sdc \
  -lef ../lib/lef/top.lef \
  -def ../pr/top.pr.def \
  -mapscript swapcells.tcl \
  -INStancenaming "ECO2inst_%d" -NETnaming "ECO2net_%d" -SEQuentialnaming
  "ECO2reg_%s" \
  -rcexec "rc -nogui" \
  -verbose
write eco design -newfile %s.post.G3 -replace -report ECOpostlogics.rpt
```

Notes:

- There might not be enough spare and free resources available to implement the ECO. Make sure that all spare cells are specified with the `ADD SPARE CELL` command.
- The mapping process will not map registers and latches to spare or freed cells. You can manually edit the ECO netlist to change the register and latch cell type.

8

Checking the ECO Netlist

Run equivalency checking on the new netlist to the ECO to make sure the ECO netlist has the correct functionality. If test needs to be disabled, also perform a comparison with test enabled to the old netlist. This comparison should be equivalent.

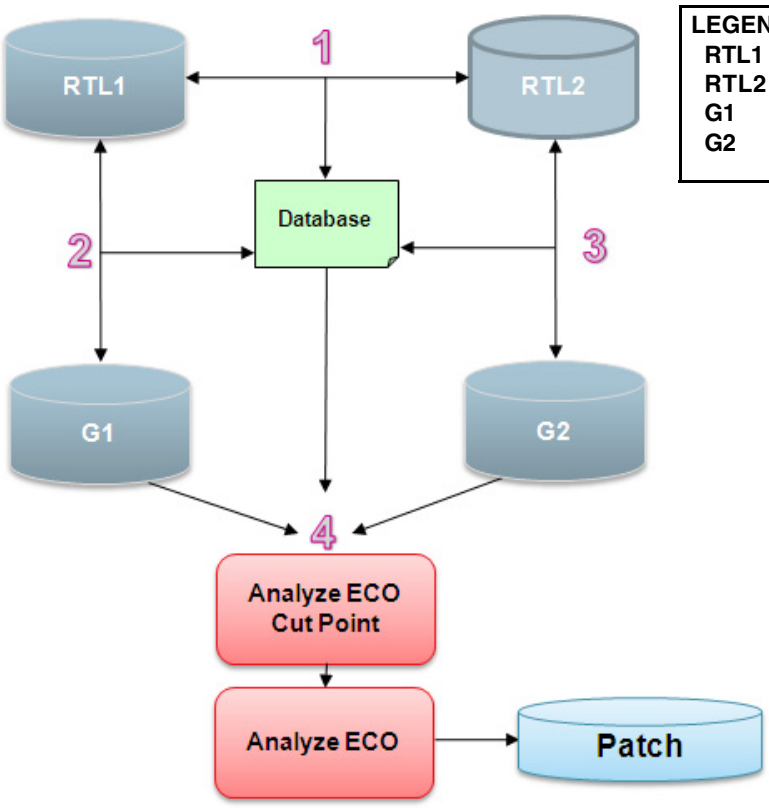
The ECO netlist can now be read into the SoC Encounter tool for incremental optimization and place and route.

9

ECO Cut Point Flow

Use the ECO cut point flow when you want an automated way to refine and possibly reduce the size of an ECO patch. This flow leverages ECO input data and existing RTL- and gate-level netlist information to create a database of reference points; this database will drive the reduction of the ECO patch size.

ECO Cut Point Flow (Diagram)



LEGEND:
RTL1 Original RTL
RTL2 RTL with ECO
G1 Original netlist
G2 Synthesis netlist from modified RTL

ECO Cut Point Flow (Steps)

A typical scenario for the ECO cut point flow involves the following steps:

1. Compare RTL1 and RTL2.
   a. Use the `READ DESIGN` command to read in the original RTL (RTL1) and the RTL (RTL2) that has the ECO.
   b. In LEC mode, use the `ADD COMPARED POINTS` command to add mapped points to the compare list.
   c. Use the `COMPARE` command to start the comparison.

d. Use the ANALYZE ECO command with the -create_db option to analyze the comparison, find potential RTL net cutpoint candidates, and to save the candidates into the cut point database.

For example:

```
set x conversion e -both
read design r1.v -golden -replace
read design r2.v -revised -replace
set system mode lec
add compare point -all
compare
analyze eco -create_db r1r2 r1r2.db
```

2. Compare RTL1 and G1.

a. Use the READ DESIGN command to read in the original RTL (RTL1) and the original netlist (G1).

b. In LEC mode, use the ADD COMPARED POINTS command to add mapped points to the compare list.

c. Use the COMPARE command to start the comparison.

d. Use the ANALYZE ECO command with the -cutpoint option to analyze the comparison, find equivalent “RTL1 net- G1 gate” pairs into the cut point database.

For example:

```
read design r1.v -golden -replace
read design g1.v -revised -replace
set system mode lec
add compare point -all
compare
analyze eco -create_db r1g1 r1g1.db -import_db r1r2.db
```

3. Compare RTL2 and G2.

a. Use the READ DESIGN command to read in the RTL2 and G2.

b. In LEC mode, use the ADD COMPARED POINTS command to add mapped points to the compare list.

c. Use the COMPARE command to start the comparison.

d. Use the ANALYZE ECO command with the -cutpoint option to analyze the comparison, find equivalent “RTL2 net- G1 gate” pairs in the cut point database.

For example:

```
read design r2.v -golden -replace
read design g2.v -revised -replace
set system mode lec
add compare point -all
compare
analyze eco -create_db r2g2 r2g2.db -import_db r1r2.db
```

4. Combine the database entries from steps 2 and 3 to form triples (RTL nets, G1 gates, and G2 gates) and insert into G1 and G2. For example:

```
read library ../LIB/VERILOG/*.v -both -replace

read design g1.v -golden -replace
read design g2.v -revised -replace

set eco option -flat
flatten -nolib -matchhier -revised
uniquify -all -nolib -revised

add pin constraint 0 scan_se -golden

set system mode lec
analyze hier -eco_aware
add compare points -all
compare

compare eco hier
report eco cut  -import_db ./DB/r1g1.db ./DB/r2g2.db -auto -file
addcut.do
dofile addcut.do
analyze eco cutpoint
analyze eco patch.v -replace
set system mode setup
apply patch -auto
```

Sample output for `ANALYZE ECO CUTPOINT`:

```
// Command: analyze eco cutpoint
// Compare result for original non-equivalent compare points
=================================================================
PO        Total
-----------------------------------------------------------------
Equivalent        24        24
-----------------------------------------------------------------
Non-Equivalent    0         0
=================================================================
```

Sample contents for cut.do file:

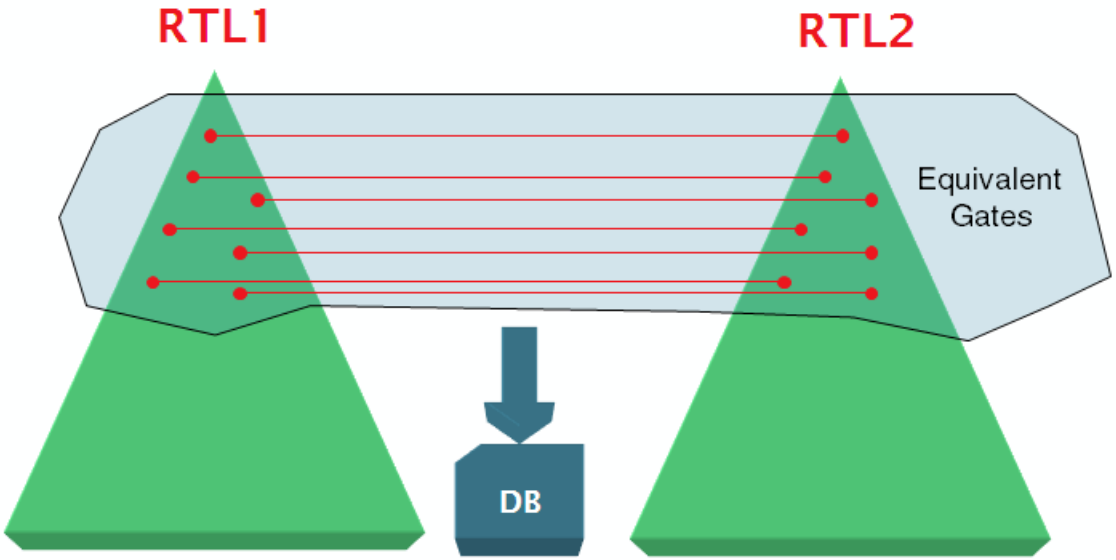
```
// 1: Cutpoint data_in[16]
add eco cutpoint data_in[16] \
-GOLDEN  + U45/g4 \
-REVISED  + U179/g1/__UU$3 \

// 2: Cutpoint data_in[19]
add eco cutpoint data_in[19] \
-GOLDEN  + U35/g4 \
-REVISED  + U173/g1/__UU$3 \

// 3: Cutpoint data_in[23]
add eco cutpoint data_in[23] \
-GOLDEN  + U97/g3 \
-REVISED  + U164/g1/__UU$3 \
...
```

Finding Cut Point Candidates

Comparing RTL1 and RTL2

The following illustrates how the tool finds cut point candidates by comparing RTL1 and RTL2 (step 1 of "ECO Cut Point Flow (Steps)" on page 58).

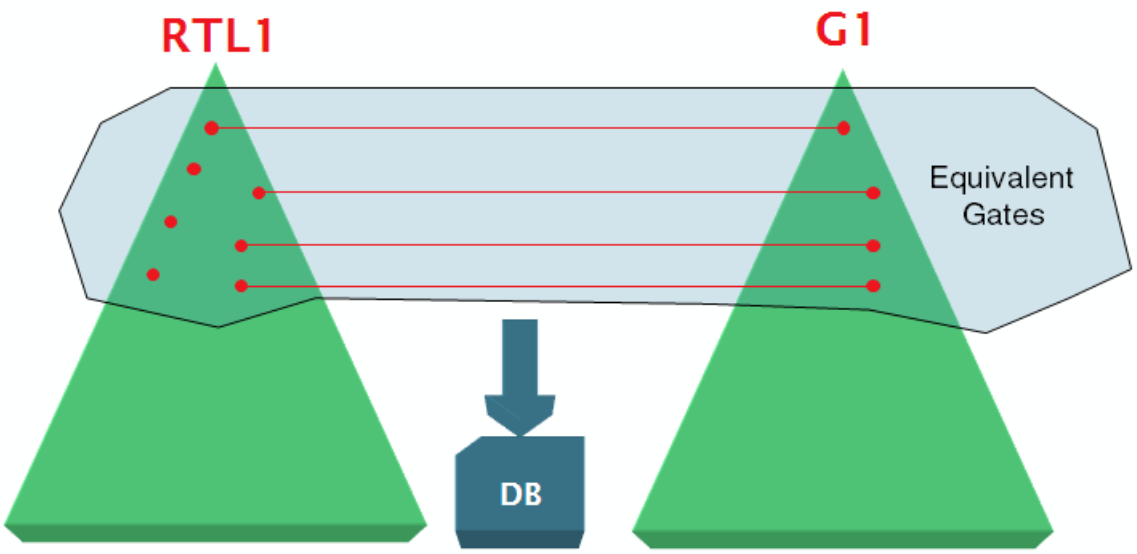
To find cut point candidates, the tool searches through RTL1 and RTL2 in a top-down fashion to find matching gates. The matching gates are then considered cut point candidates and are stored in the cut point database.



Database entries (RTL1, Net Name, and G1 Gate)

Comparing RTL 1 and G1

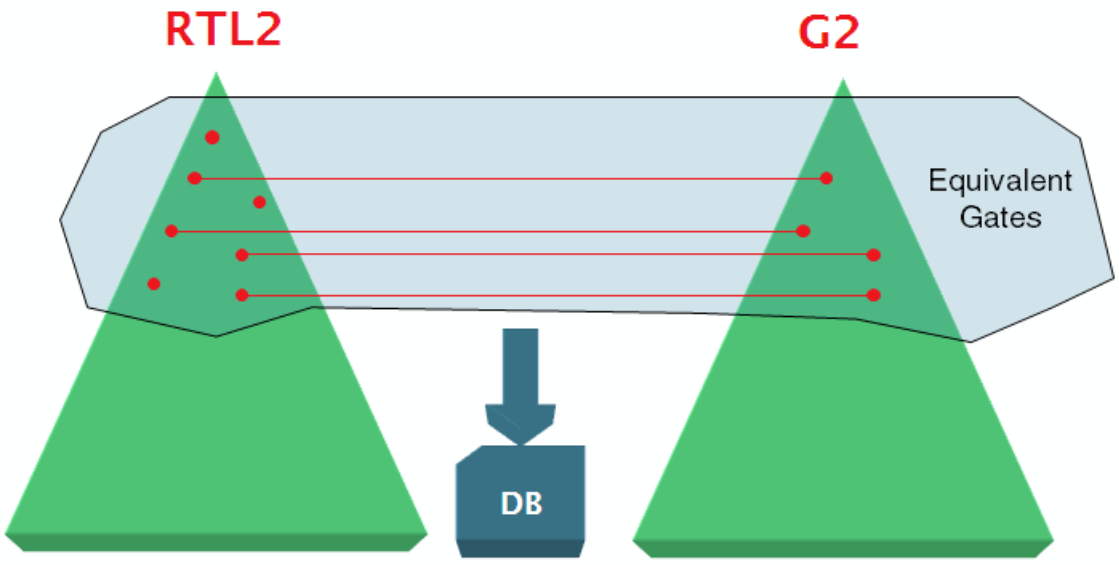
The following illustrates how the tool finds cut point candidates by comparing RTL1 and G1 (step 2 of “ECO Cut Point Flow (Steps)” on page 58).



Database entries (RTL1 Net Name, G1 Gate)

Comparing RTL2 and G2

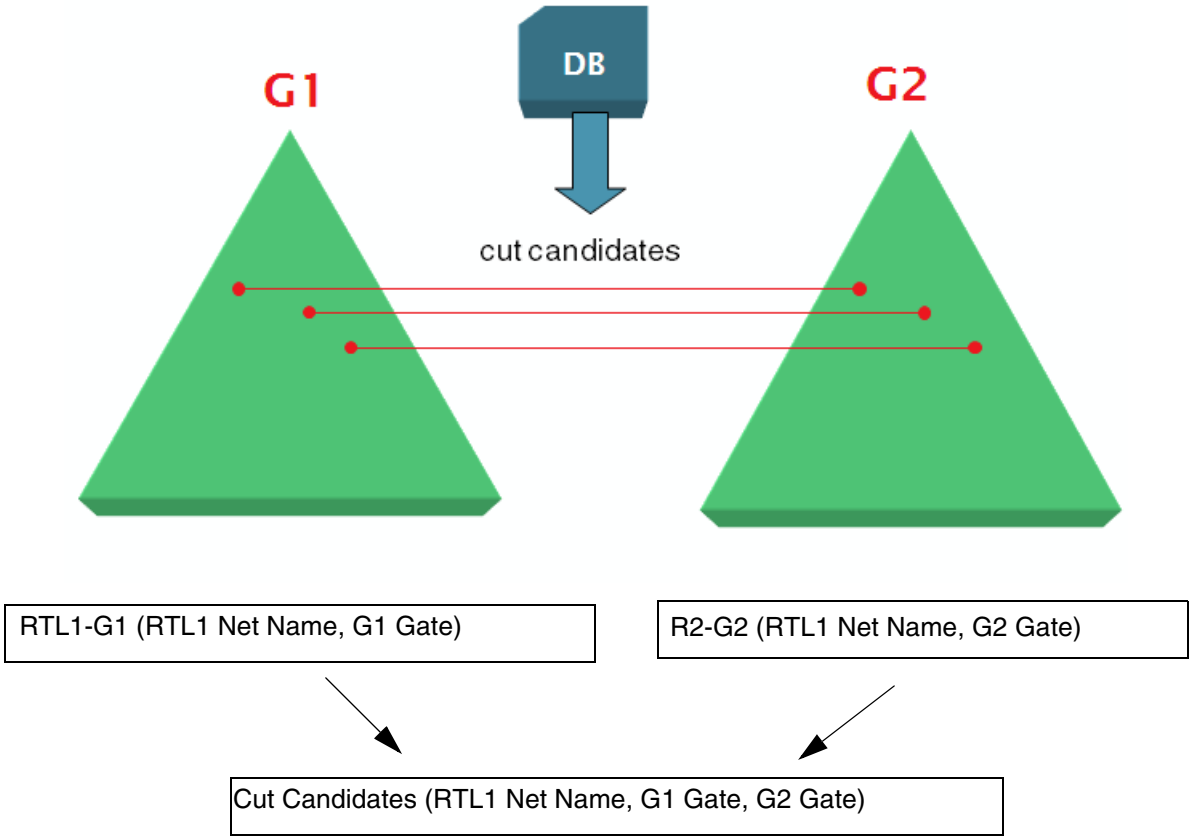
The following illustrates how the tool finds cut point candidates by comparing RTL2 and G2 (step 3 of “ECO Cut Point Flow (Steps)” on page 58).



Database entries (RTL1 Net Name, G2 Gate)

Inserting Cut Points

The following illustrates how the tool finds cut point candidates by comparing RTL2 and G2 (step 4 of “ECO Cut Point Flow (Steps)” on page 58).



10

Generic Script Format

The `REPORT ECO CHANGES -script` command reports ECO changes and then generates a generic script that can be used with other tools. This section describes the generic script format.

1. `set_root_module <module_name>`

   Sets the current root module to `<module_name>`. The editing script that follows this command will be applied to the objects in specified module.

2. Add ports:

   a. `add_port <port_name>`

      Adds the scalar port with `<port_name>`.

   b. `add_port <port_name>[msb:lsb]`

      Add or augments the bus port with `<port_name>`. The bus is specified by `[msb:lsb]`. Refer to the `ADD ECO PIN` command for more information.

3. `delete_port <port_name>`

   Deletes the port specified by `<port_name>`.

4. `add_instance <instance_name> <module_name>`

   Adds an instantiation of module `<module_name>` with name `<instance_name>`.

5. `add_primitive <instance_name> <primitive_type> <input_count>`

   Adds an instantiation of the Verilog primitive of `<primitive_type>` with the name `<instance_name>` and `<input_count>` inputs. If the patch is mapped with command `OPTIMIZE PATCH`, all the primitives should already be mapped to the library cells.

6. `delete_instance <instance_name>`

   Deletes the instance specified by the `<instance_name>`.

7. `add_assign <lhs_net_name> <rhs_net_name>`

Adds an assign statement that assigns the net <rhs_net_name> to net <lhs_net_name>.

8. delete_assign <lhs_net_name> <rhs_net_name>

   Deletes the assign statement that assigns the net <rhs_net_name> to net <lhs_net_name>.

9. connect_pin <instance_name> <pin_name> <net_name>

   Connects the instance pin to the specified net.

10. disconnect_pin <instance_name> <pin_name>

    Disconnects the instance pin from the net to which it connects.

11. Add nets:

    a. add_net <net_name>

       Add a net with name <net_name>.

    b. add_net <net_name>[msb:lsb]

       Add the bus net with the name specified by <net_name>. The bit range of that bus net is specified by [msb:lsb]. Adds a net with name <net_name>.

12. delete_net <net_name>

    Deletes the net with name <net_name>.

13. rename_net <orig_net_name> <new_net_name>

    Renames the net from <orig_net_name> to <new_net_name>.

14. tie_net <net_name> <0|1>

    Ties the net <net_name> with constant 0 or constant 1.

15. untie_net <net_name>

    Unties the net <net_name> if it is tied to constant 0 or constant 1.

16. tie_pin <instance_name> <pin_name> <0|1>

    Ties the specified instance pin to constant 0 or constant 1.

11

Debugging and Troubleshooting in Conformal ECO

List of Debugging Topics

Debugging Nonequivalent Points

Failures or Unexpected Results

Invalid, Missing, or Redundant Data

Spare Cells

ANALYZE ECO Fails

If the `ANALYZE ECO` command fails, try the following:

- Enable `SET FLATTEN MODEL -eco`
- Run the `ANALYZE ECO` command again with `-effort medium`
- Check the dofile for errors, such as:
  - `ECO` modules that are not written out for hierarchical comparison
  - The `READ LIBRARY` and `OPTIMIZE PATCH` commands do not use both liberty libraries

[Back to List of Debugging Topics]

APPLY PATCH or OPTIMIZE PATCH Fails Due to Referenced Cells/Modules

Issue	The tool reports that you referenced a cell/module that is not defined.

Message	Message similar to the following is displayed.
	
	Scenario 1) `APPLY PATCH` errors out:
	
	`// Command: apply patch -auto`
	`...`
	`// Warning: results/dpatch.v:15 Module 'NR2' is referenced but not defined.`
	`// Note: Read VERILOG design successfully`
	`// Error: Cannot find module mod_vol_eco in Golden design`
	
	Scenario 2) `APPLY PATCH` completes, but `OPTIMIZE PATCH` errors out:
	
	`...`
	`// Note: read design patch.v -append -lastmod -norulesummary`
	`// Parsing file patch.v ...`
	`// Error: patch.v:9 Module 'OR3X1' is referenced but not defined.`

Explanation	In Scenario 1, a Verilog command file is used to read in the design. In this case, the tool reads in only the modules/cells that it needs into the library space. This can cause this error if you are trying to access a cell/module that exists in G2 only, because the tool will not read that cell/module into the G1 library space. In Scenario 2, `OPTIMIZE PATCH` tries to use a library cell/module that has not been read in. This can happen if you have not read in all of the library cells/modules by the time you execute `OPTIMIZE PATCH`.

Workaround	Try not to use a Verilog command file to read in the design. Or, explicitly read in the library cells/modules in the command file. For example, instead of:
	
	`-v slow_cg_mb.v`
	`R1.v`
	
	Try:
	
	`read library slow_cg_mb.v -both`
	`read design R1.v -golden`
	`read design R2.v -revised`
	
	Also, use only Liberty library files, since RTL Compiler uses Liberty library files.

[Back to List of Debugging Topics]

Cannot Find Corresponding Nets

Issue	The tool cannot find a gate's corresponding net.

Message	A message similar to the following is displayed:
	`// Command: analyze eco ...`
	`// Error: Failed to find corresponding net(s) in the root module for the following gate(s) (G) 471 OR / rd_col_ptr_cell_reg_regx0x/U$5`
	`// Error: Failed to find a valid patch`

Workaround	Run the following command: `set flatten model -eco` In addition, ensure that: ■ The technology LEF libraries are read in as part of the library, and not as part of the design. ■ The `READ LIBRARY` and `OPTIMIZE PATCH` commands refer to the same library.

[Back to List of Debugging Topics]

Duplicate Fanout Branches

Issue	The tool reports a duplicate fanout branch.

Message	A message similar to the following is displayed:
	`// Command: apply patch ...`
	`// Error: Duplicate fanout branch 1 for net 'cculsdc_lsdc_ls_lsmoq_blm_ls_moq_com1__RCLK_Int2Moq6_AR_com_clk_gate_root_CLOCK_ROOT_864_1104_S2_out' (pin:btcore_ls_moq_1_eco.cculsdc_lsdc_ls_lsmoq_blm_ls_moq_com1__RCLK_Int2Moq6_AR_com_clk_gate_root_CLOCK_ROOT_864_1104_S2_out_1o1)`

Workaround	Run the following command: `ANALYZE ECO -effort medium`

[Back to List of Debugging Topics]

ECO_SYNTHESIS Needs Information

Issue	`eco_synthesis` command needs particular information to proceed.

Message	A message similar to the following is displayed:
	`// ECO-ERROR: Either spare gate or freed gate or filler box list needs to be passed to the eco_synthesis command`
	`Encountered problems processing file: cfm_eco_rc.tcl`
	`Abnormal exit.`
	`//Warning: RC returns error code 244`

Explanation	RTL Compiler cannot find available spare cells, freed cells, or gate array filler cells for mapping.

Workaround	Gate array filler cells are passed to RTL Compiler using the `ADD SPARE CELL -gafiller` command. If this command did not add any filler cells, the cell name might be incorrect.
	For example:
	`// Command: add spare cell -def ../tapeout.def -gafiller GA_FILL*`
	`// Note: 0 cell added`
	`// Command: add spare cell -def ../tapeout.def -gafiller GAFILL*`
	`// Note: 12932 cells added`

[Back to List of Debugging Topics]

Extra or Invalid Ports are Reported

Issue	The tool reports extra or invalid ports.

Message	A message similar to the following is displayed:
	`// Warning: Primary input 'port123' in Golden has no`
	`// correspondence in Revised`
	`// Warning: Primary input 'port234' in Revised has no`
	`// correspondence in Golden`

Explanation/ Workaround	Extra or invalid ports can lead to nonequivalent points, invalid patch generation, and unexpected errors.
	
	Ensure renaming rules are in place to handle any port naming between G1 (original netlist) and G2 (netlist after ECO). For example:
	`add renaming rule rule1 net123 net234 -pin -bbox <moduleA> \`
	`  -golden`
	
	Ensure the `ADD ECO PIN` and `DELETE ECO PIN` commands are used when a pin is added or deleted due to an ECO. For example:
	`add eco pin <moduleA> <port> –input -golden`

Applicable Step	Mapping of G1and G2.
See Also	ANALYZE ECO Fails

[Back to List of Debugging Topics]

Invalid Data Value

Issue	The tool reports invalid attributes.

Message	A message similar to the following is displayed:
	`Error : The data value for this attribute is invalid. [TUI-24] [set_attribute]`
	`: The value '/projects/…/hif.lef cannot be set for attribute 'lef_library'.`
	`: To see the usage/description for this attribute, type 'set_attribute -h <attr_name> *'.`
	`// Warning: RC returns error code 244`

Explanation	The specified LEF file does not contain information about metal layers and vias. This information is in a separate file that must be read in *before* the LEF file.

[Back to List of Debugging Topics]

Fanout Branch Missing

Issue	The tool reports that it cannot find a fanout branch for a particular net.

Message	A message similar to the following is displayed:
	`// Error: Cannot find fanout branch 8 for net`
	`// 'hdlc_ser_rstn_L2_N14'`
	`// (pin:hdlc_ser_rx_eco.hdlc_ser_rstn_L2_N14_8o)`

Explanation	The patch cannot modify the logic of the <id>th fanout, because the net specified by <net_name> does not have enough fanouts.

Workaround	Ensure that the hierarchy is the same when you run `ANALYZE ECO` and `APPLY PATCH`. If the hierarchy is not the same, blackboxes might have been deleted prior to executing `APPLY PATCH`.
	
	You can also try flattening the netlists by using the following commands:
	`flatten -nolib -golden`
	`flatten -nolib -revised`

[Back to List of Debugging Topics]

Tool Reports No NEQs to ECO

Issue	The tool reports that there are no nonequivalent keypoints to ECO.

Message	A message similar to the following is displayed:
	`// Error: Cannot find any diff key point`
	`Message:`
	`========================================`
	`Compared points PO BBOX Total`
	`----------------------------------------`
	`Equivalent 103 0 103`
	`----------------------------------------`
	`Non-equivalent 0 6 6`
	`========================================`
	`// Command: analyze eco ...`
	`// Error: Cannot find any diff key point`

Explanation/ Workaround	You might not have any correspondence points, as indicated by the following message: `// Warning: Primary input 'port123' in Golden has no correspondence in Revised` If it is an ECO module, you may need to insert an ECO pin by using the `ADD ECO PIN` command. There could also be pin differences in your blackboxes. For example, only G1, and not G2, has scan pins. In this case, this is not a ECO module and you do not need to run `ANALYZE ECO` on it. You can also try running the flattened ECO flow.

[Back to List of Debugging Topics]

Lines are Commented Out After -eco_aware

Issue	When writing out the hierarchical dofile using the `-eco_aware` option, the tool comments out lines that include the `ADD IGNORED INPUT` command.

Example	For example, the dofile contains commented-out lines similar to the following:
	`// Commented out for -ECO_aware option`
	`// add ignored inputs PRESETDBGn -Golden`
	…
	`// Commented out for -ECO_aware option`
	`// add ignored inputs ATRESETn_hfs_netlink_232 -Golden`

Explanation	These commands are commented out because the pin in the Revised design is reachable, and its corresponding pin in the Golden design will be reachable once the ECO is complete.

[Back to List of Debugging Topics]

Missing Modules

Issue	The tool reports that it cannot find a module.

Message	A message similar to the following is displayed:
	`// Warning: Module 'AAA_1' exists in Golden only.`
	`// Skip it for hierarchical comparison`
	`// Warning: Module 'AAA_N_3' exists in Revised only. Skip it for`
	`// hierarchical comparison`

Workaround	Run the `UNIQUIFY` command to make the specified module, which has multiple instances, unique. If the tool cannot make the modules unique, they are not included in the hierarchical dofile.
	This command lets you remedy the "incompatible" instantiations warnings during hierarchical script generation.
	For example:
	`uniquify AAA_N_3 -nolib -revised`
	`uniquify -all -nolib -revised`

Applicable Step	Mapping of G1and G2.
See Also	ANALYZE ECO Fails

[Back to List of Debugging Topics]

Missing Tie Cells

Issue	The tool cannot proceed with tiecell insertion.
Message	A message similar to the following is displayed: `// Warning : No tie hi/lo cell found for tiecell insertion. [UTUI-204]: Could not find a tie hi/lo cell to insert tiecells in /designs/my_eco.: Possible reason is that the tiecells in library are avoided, if present. Unavoid them to use for tiecell insertion. // Error: Cannot proceed with tiecell insertion. [UTUI-216]`
Explanation	The tool cannot find tie high or tie low cells for tiecell insertion. This is most likely due to the use of `-nousetiecell` option of `OPTIMIZE PATCH`. This option specifies that the RTL Compiler can leave constants as assign statements instead of using tie high or tie low cells (thus avoiding them).
Workaround	Do not use the `-nousetiecell` option of `OPTIMIZE PATCH`.

[Back to List of Debugging Topics]

Modules in the Patch File

Issue	The tool reports that modules are showing up in the patch file.
Message	A message similar to the following is displayed: `// Warning: 1 module instance is in the patch`

Explanation/ Workaround	This usually occurs when there are nonequivalent points in the module. Nonequivalent points can appear in the module when the hierarchical comparison does not write out the module.
	Do the following:
	■ Set the threshold to zero using `WRITE HIER_COMPARE DOFILE -threshold 0` or `ANALYZE HIER_COMPARE -threshold 0`
	■ Map renaming rules
	You can map renaming rules to match the instance pathnames between golden and revised. For example:
	`// ha_rx_ard_ipc_add_13_6 (Matched in Golden) (Instance: u110_u1) (R)`
	You can also use the following renaming rule to ensure that the instance name matches the golden instance name:
	`add renaming rule r1 u110_u1 I648_u1 -map -rev`
	■ Check the logfile. Sometimes, the `WRITE HIER_COMPARE DOFILE` command writes out messages similar to the following to the logfile:
	`"7 module pairs are not written because of non-matching instance pathnames, although the module names match"`
	■ Use `UNIQUIFY -all -revised -nolib` to make all of the modules in the revised design that have multiple instances unique.

[Back to List of Debugging Topics]

NEQ point(s) Found in a Submodule

Issue	The tool found nonequivalent points in a submodule.

Message	A message similar to the following is displayed:
	`// Command: analyze eco ...`
	`    // Warning: NEQ point(s) in the submodules found:`
	`    + 5205 DFF /mod1_u/regA + 5208 DFF /mod1_u/regA`
	`    // Note: Please use hierarchical method to generate the patch file for each`
	`    // sub-module`
	
	Or:
	`// Note: 1746611 library cell(s) are in the patch`
	`// Warning: 16 module instance(s) are in the patch`
	`// Note: Please use hierarchical method to generate patch file for each hierarchy`
	`// for better quality`

Explanation	The tool encountered nonequivalent points in a subhierarchy. Either hierarchical comparison was not performed, or the hierarchical comparison failed.

Workaround	Run a hierarchical comparison if you have not already done so.
	
	During the hierarchical comparison, write out the affected submodule by using the following command:
	`add module attribute <sub_module_name> -eco_module \`
	`  -noflatten -both`

[Back to List of Debugging Topics]

No Mapping for Gate Array

Issue	The tool specifies that none of the gate array library cells can be slotted into the filler cells.

Message	A message similar to the following is displayed:
	`// Command: add spare cell -def ../pnr1/OV2710_ECOAC_0605.def -gafiller FILLER_GA*`
	`// Note: 87136 cells added`
	`// Command: optimize patch ...`
	`ECO-INFO: Final summary of the number of GA libcells that be can be slotted into the filler cells`
	`============================================================`
	`Generated by: Encounter(R) RTL Compiler v09.10-s203_1`
	`...`
	`/libraries/LIB_Gatearray_SS_v9/libcells/ga_nor2c 0`
	`/libraries/LIB_Gatearray_SS_v9/libcells/ga_nor3b 0`
	`/libraries/LIB_Gatearray_SS_v9/libcells/ga_nor4b 0`
	`ECO-INFO: Total ECO Violations -15`
	`Normal exit`
	`Problem: No ERROR message & No mapping to gate array`

Explanation	An LEF component for the gate array filler cell is missing.
Workaround	Do the following: **1.** Confirm that you have all the required DEF/LEF/LIB components: ❑ DEF file: GA filler cells placement ❑ LEF: GA cells and GA filler cells ❑ LIB: GA cells and GA filler cells **2.** Check the availability of spare gates by using the `REPORT SPARE CELL` command.

[Back to List of Debugging Topics]

Spare Cells Used That are Not Part of the Spare Cell List

Issue	Conformal ECO uses cells that are not part of the spare cell list.

Message	For example, the Tcl script that Conformal ECO writes for RC—such as `cfm_rc_eco.tcl`—contains the following:
	`addInst -moduleBased afblock -cell NAND2BX1 -inst {g16917}`
	`addInst -moduleBased afblock -cell AOI22X1 -inst {g16919}`
	`addInst -moduleBased afblock -cell OAI2BB2XL -inst {g15}`
	None of these cells are part of the spare cell list.

Explanation/ Workaround	There might not be enough spare cells to map the ECO. To determine whether this is the issue, check the RC run. If you see a cell that is mapped to `NO_MAP`, it means there are no spare cells available to map this cell: `g16917 NO_MAP` By default, RC continues with synthesis even if there no spare cells are available. To disable this feature, set the RC `eco_synthesize_without_spare_cells` attribute to `FALSE`. This causes RC to exit with error if no spare cells are available at the beginning of patch synthesis. If some spare cells available, RC proceeds with synthesis even with ECO violations.

[Back to List of Debugging Topics]

Spare Cell DFF is Not Available

Issue	You have a post-mask ECO that uses spare cells. Part of the ECO adds a `PRESET DFF`; however, you have only a `RESET DFF` spare cell available.
Workaround	Add the following attribute to the pre-library script (this script is read in before the libraries, and is specified using the `-PRELIBscript` option of the `OPTIMIZE PATCH` command): `set_attr lbr_seq_in_out_phase_opto true /` This allows Conformal ECO and RTL Compiler Physical to map a spare DFF of an inverter type.

[Back to List of Debugging Topics]

Super Threading is Disabled

Issue	The tool specifies that super threading is disabled in ECO mode.
Message	A message similar to the following is displayed in the logfile even though you specified super threading when you executed the RTL Compiler: `Info: Super-threading will be disabled in ECO mode.`
Explanation	Super threading is disabled in Conformal ECO because global resource constraints cannot be processed in parallel.

[Back to List of Debugging Topics]

NEQs Due to Unexpected Blackboxes

Issue	The `REPORT BLACK BOX` command reports unexpected blackoxes.

Message	A message similar to the following is displayed:
	`// Command: report black box -details`
	`SYSTEM (notranslate): (G R) RAM1`
	`SYSTEM (undefined): (G) AAA //Unexpected`
	`SYSTEM (undefined): (R) BBB //Unexpected`

Explanation/ Workaround	Unexpected blackboxes (except RAMs or hard macros) can lead to nonequivalent points, abort patches, and unnecessary logic in the design. Confirm that the Liberty and Verilog files were read correctly or if any files are missing.
Applicable Step	While reading in the design
See Also	ANALYZE ECO Fails

[Back to List of Debugging Topics]

Unexpected NEQ Between G2 and G3

Issue	After applying a patch and confirming its validity, the tool reports that G2 and G3 are not equivalent.

Explanation/ Workaround	There might be a mapping problem or an issue with the comparison setup.
	
	A. When `OPTIMIZE PATCH -sequentialnaming` is used, newly-added DFFs are renamed in G3. Therefore, the flop names in G2 (the ECOed netlist) will not match those in G3.
	
	This can cause false NEQs when name-only mapping is enabled (`SET MAPPING METHOD -name only`).
	
	B. You should use the same setup to compare G1 and G2, and to compare G2 and G3. For example, the following can cause an unexpected NEQ because `set analyze opt -auto` was used to compare G1 and G2, but was not used to compare G2 and G3
	
	`...`
	`set analyze opt -auto`
	`compare g1g2`
	`analyze eco`
	`apply patch`
	`optimize patch`
	
	`reset`
	`compare g2g3`

Workaround	A. Avoid using both options in the same flow. Or, use `SET MAPPING METHOD -name first` (instead of "`only`"). B. Set up your comparisons consistently.
Applicable Step	Comparing G2 and G3.

[Back to List of Debugging Topics]

Unexpected Patch Size

Issue	The tool reports an unexpected patch size.

Message	A message similar to the following is displayed:
	`// Command: analyze eco ...`
	`// Note: 27 group(s) added`
	`// Note: 86 library cell(s) are in the patch`
	`/// Warning: 1 DFF clock(s) is changed in the patch`
	`// Note: 276 primitive(s) are in the patch`

Workaround	Set the threshold to zero by using either of the following commands:
	`WRITE HIER_COMPARE DOFILE -threshold 0`
	`ANALYZE HIER_COMPARE -threshold 0`

[Back to List of Debugging Topics]

NEQs Due to Unmapped Points

Issue	The `REPORT UNMAPPED POINT -notmapped` command reports unmapped points.

Message	A message similar to the following is displayed:
	`==========`
	`Revised:`
	`----------`
	`Unmapped points DFF DLAT BBOX Total`
	`----------`
	`Not-mapped 100 0 0 100`
	`==========`
	`// Warning: Key point mapping is incomplete`

Explanation/ Workaround	Unmapped points should be from either newly-added or deleted DFFs or ports. Other types of unmapped points (such as unmapped points from renamed DFFs) can lead to nonequivalent points, abort patches, and unnecessary logic in the paths.
Applicable Step	Mapping of G1 (original netlist) and G2 (netlist after ECO)
See Also	ANALYZE ECO Fails

[Back to List of Debugging Topics]