Formality ECO User Guide

Version S-2021.06, June 2021

SYNOPSYS®

Copyright and Proprietary Information Notice

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Feedback

Contents

Contents

About This User Guide

The Formality ECO User Guide describes how to use Formality ECO tool to automatically generate an ECO patch. The Formality ECO tool produces a Tcl script that can be applied to the original netlist using IC Compiler, IC Compiler II, Design Compiler, or Fusion Compiler to make it functionally equivalent to ECO changes in the RTL.

This guide requires you to be familiar with using the Formality tool for verification, and capable of creating and modifying verification scripts. You also need to understand how to modify and use synthesis scripts because the Formality ECO flow requires the use of either the Design Compiler or Fusion Compiler tool for ECO synthesis.

This preface includes the following sections:

New in This Release

Information about new features, enhancements, and changes, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the Formality ECO Release Notes on the SolvNetPlus site.

Related Products, Publications, and Trademarks

For additional information about the Formality ECO tool, see the documentation on the Synopsys SolvNetPlus support site at the following address:

https://solvnetplus.synopsys.com

You might also want to see the documentation for the following related Synopsys products:

- Formality®

Conventions

The following conventions are used in Synopsys documentation.

Feedback

Convention	Description
`Courier`	Indicates syntax, such as `write_file`.
`Courier italic`	Indicates a user-defined value in syntax, such as `write_file` *`design_list`*
`Courier bold`	Indicates user input—text you type verbatim—in examples, such as `prompt>` **`write_file top`**
`Purple`	• Within an example, indicates information of special interest. • Within a command-syntax section, indicates a default, such as `include_enclosing = true | false`
[ ]	Denotes optional arguments in syntax, such as `write_file [-format` *`fmt`*`]`
...	Indicates that arguments can be repeated as many times as needed, such as *`pin1 pin2 ... pinN`*.
|	Indicates a choice among alternatives, such as `low | medium | high`
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Bold	Indicates a graphical user interface (GUI) element that has an action associated with it.
Edit > Copy	Indicates a path to a menu command, such as opening the **Edit** menu and choosing **Copy**.
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing C.

Customer Support

Customer support is available through SolvNetPlus.

Accessing SolvNetPlus

The SolvNetPlus site includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The SolvNetPlus site also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access the SolvNetPlus site, go to the following address:

https://solvnetplus.synopsys.com

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to sign up for an account.

If you need help using the SolvNetPlus site, click REGISTRATION HELP in the top-right menu bar.

Contacting Customer Support

To contact Customer Support, go to https://solvnetplus.synopsys.com.

1

Introduction to Formality ECO

The Formality ECO tool uses specialized commands to generate an ECO patch that can be used to modify an original netlist to be functionally equivalent to its ECO RTL. The tool can also generate an ECO patch if you start with a previously patched netlist. This tool significantly reduces the turnaround time for generating functional ECOs in complex designs.

Formality ECO Features and Advantages

The Formality ECO tool supports the following features:

- All varieties of pre mask functional ECOs
  - Combinational logic changes, addition or removal of registers, datapath or control path changes
  - Small rewiring ECOs (directly patched without ECO synthesis)
- Clock trees
- Design Compiler or Fusion Compiler optimizations
- DesignWare including pipelined components
- DFT transformations and scan chains (generated by Synopsys or other third-party tools)
- ECOs on low-power (UPF) designs when the ECO does not cross power domain boundaries
- ECOs on retimed designs (if the ECO is not in the fan-in or fan-out range of the retimed design portion)
- Multibit registers, constant registers, and merging, phase inversion, and replication of registers
- SVF checkpoint flows using the Design Compiler Graphical and Fusion Compiler tools

The Formality ECO tool has the following advantages:

- Generates an ECO patch for any Design Compiler or Fusion Compiler optimization
- Maintains the multibit register mapping of the original design when the ECO is implemented
- Maintains the scan chain order, though you need to connect new scannable flip-flops to the scan chain if required
- Uses the Design Compiler or Fusion Compiler tool so that timing and other important considerations are utilized during ECO RTL synthesis

Formality ECO Input Requirements

The Formality ECO flows require the following input sources:

- original RTL
- original netlist (along with SVF)
- ECO RTL

Formality ECO Licensing

To invoke the Formality ECO tool, use the `fmeco_shell` executable. This executable includes all Formality, Formality Ultra, and Formality ECO functionalities. It requires the use of the `Formality-LogicECO` feature license key.

Formality ECO Classifications

The Formality ECO tool consists of the following flows:

- Formality ECO RTL flow
- Formality ECO netlist flow

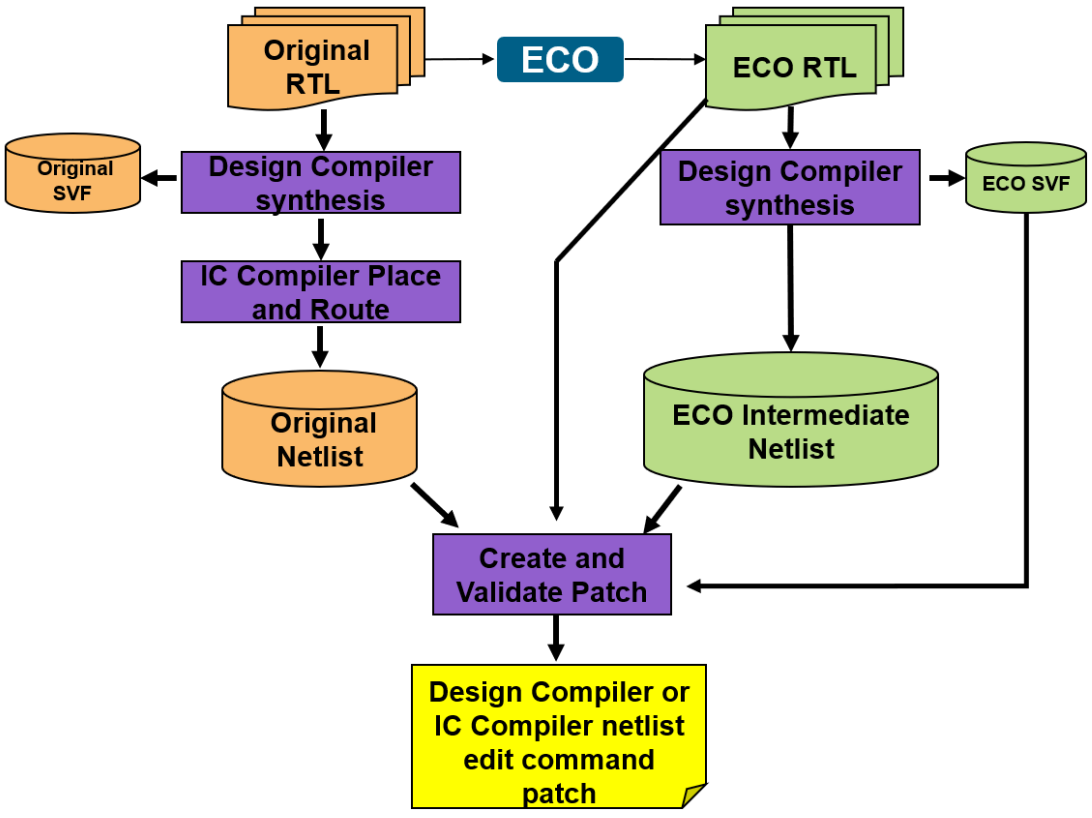
2

Formality ECO RTL Flow

The Formality ECO RTL flow compares an ECO RTL against the original RTL to find ECO changes in the affected regions. It uses synthesis of the ECO RTL to create an intermediate ECO implementation.

This flow produces a Tcl script using downstream tools (Design Compiler, Fusion Compiler, IC Compiler, and IC Compiler II), which modifies the original netlist (O-NET) to be functionally equivalent to the ECO RTL (E-RTL). Figure 1 shows the basic Formality ECO RTL flow:

Figure 1 Formality ECO RTL Flow



Feedback

ECO RTL Flow Prerequisites

The Formality ECO RTL flow assumes that the following steps are already completed:

- The original netlist is synthesized using Design Compiler or Fusion Compiler.
  - Using netlists and SVF files generated from the latest versions of Design Compiler or Fusion Compiler is always recommended. However, older versions of Design Compiler or Fusion Compiler may still work when used for the original synthesis of the netlist and SVF.
  - The synthesis of the ECO RTL using the latest version of Design Compiler or Fusion Compiler is recommended.
- The original RTL and original netlist are successfully verified using Formality.
- The original SVF file is available.
- The ECO RTL is simulated and considered to be the new golden source code.

ECO RTL Flow Summary

The Formality ECO RTL flow requires the original RTL, the original netlist (along with SVF), and the ECO RTL as inputs. The Formality ECO tool compares the ECO RTL against the original RTL to find the affected regions from the ECO.

The Design Compiler or Fusion Compiler tool synthesizes the ECO RTL to create an intermediate ECO implementation and does not require user involvement. The gate-level netlist resulting from this synthesis already meets, or closely meets the timing and area quality of results (QoR).

Although the entire ECO RTL is synthesized, only the affected regions of the ECO are used to create a patch for the original netlist. After the intermediate ECO netlist is synthesized, the Formality ECO tool identifies the ECO regions in the original netlist and automatically generates netlist edit commands (the patch) for the IC Compiler II tool to rectify the function of the original netlist.

This flow produces a Tcl script using downstream tools (Design Compiler, Fusion Compiler, IC Compiler, and IC Compiler II), which modifies the original netlist (O-NET) to be functionally equivalent to the ECO RTL (E-RTL). This final netlist should be verified against the E-RTL using the Formality tool to yield a successful verification result.

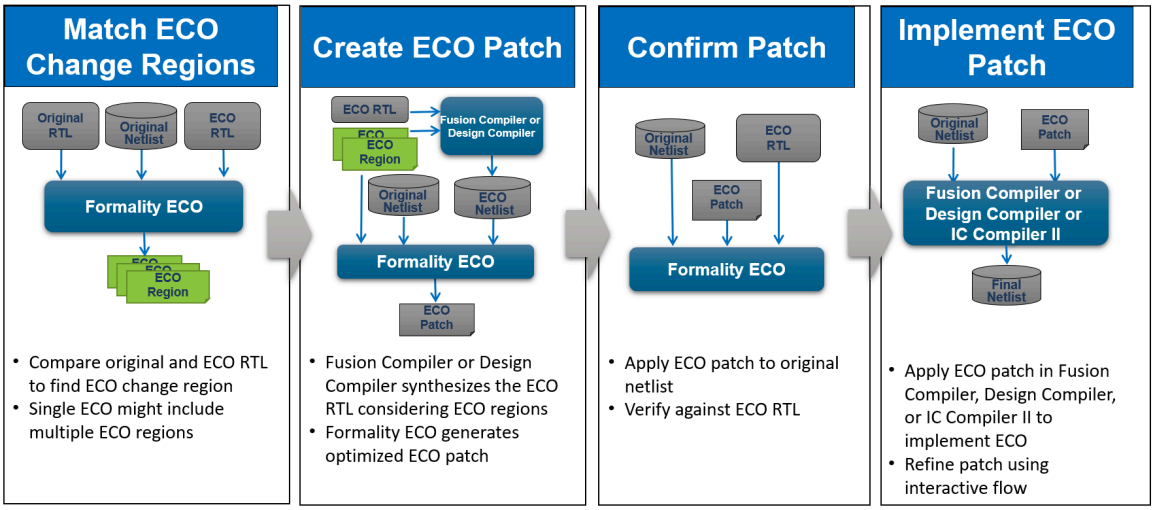
The Formality ECO tool retains existing scan chains. Any registers that are removed by an ECO are still maintained in the scan chain. New registers involved with an ECO are not inserted into scan chains. The tool reports added and restored registers so that the DFT scan chain can be manually edited.

Feedback

ECO RTL Flow Stages and Steps

The Formality ECO flow comprises four stages as shown in Figure 2:

Figure 2 Formality ECO RTL Flow Stages



1. Match ECO regions

   a. Identify the ECO regions: Compare the ECO RTL against the original RTL to identify the ECO regions, that is, the portions of the original RTL design that were modified by the ECO.

   b. Compare the ECO regions: Locate and match together the affected regions in the original RTL, ECO RTL, and original netlist.

   c. Group the ECO regions: Create ECO region groups that encapsulate the ECO regions in both the ECO RTL and original netlist.

2. Create an ECO patch

   a. Synthesize the ECO RTL: Create new ECO region designs for the ECO RTL. Synthesize and disable boundary optimization in the ECO region designs. Disabling boundary optimization preserves the boundaries in the intermediate ECO netlist.

   b. Generate a patch: Automatically generate the netlist edit commands. These netlist edit commands copy the ECO region designs from the intermediate ECO netlist and replace their counterparts in the original netlist.

3. Confirm the patch

   Verify the rectified original netlist against the ECO RTL while applying the ECO SVF to the ECO region designs and the original SVF to the remainder of the ECO RTL.

This step creates an ECO Tcl script for use in the IC Compiler, IC Compiler II, Design Compiler, or Fusion Compiler tool, which modifies the original netlist and makes it functionally equivalent to the ECO RTL.

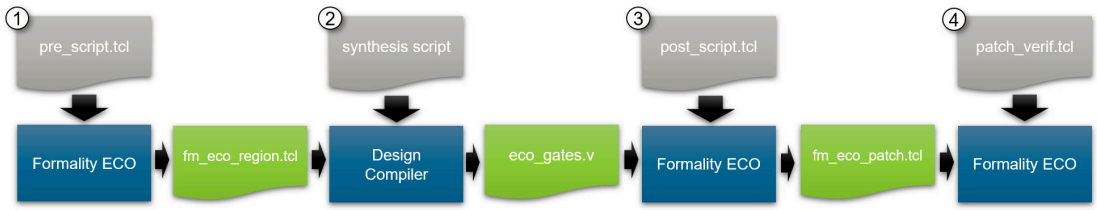
4. Implement the ECO patch

   Take the ECO Tcl script into the IC Compiler, IC Compiler II, Design Compiler, or Fusion Compiler tool to implement the gate changes on the original netlist and write out the patched ECO netlist.

ECO RTL Flow Patch Generation Scripts

Figure 3 shows the four Tcl scripts you have to create and run in the Formality ECO RTL flow to provide the final patch for the IC Compiler, IC Compiler II, Design Compiler, or Fusion Compiler tool:

Figure 3 Formality ECO RTL Flow Scripts



1. Match ECO regions (pre-synthesis)
2. ECO synthesis
3. Create ECO patch (post-synthesis)
4. Confirm ECO patch

Note:

For automatic script generation using the Formality ECO RTL flow, see Appendix A.

Table 1 includes the common terminologies used in all the four Tcl scripts of the Formality ECO RTL flow.

Table 1 RTL Flow Terminologies and Descriptions

Terminology	Description
`ortl`	This container includes the original RTL without any SVF processing.

Feedback

Table 1 RTL Flow Terminologies and Descriptions (Continued)

Terminology	Description
`ertl`	The ECO RTL is read into this container first to utilize `guide_hierarchical_map` (GHM) commands in the SVF file while setting the top design. It is subsequently saved for later use.
`r`	The original RTL is read into this container and SVF is applied to it.
`i`	The original netlist is read into this container.
O-RTL	Original RTL
E-RTL	ECO RTL
O-SVF	Original SVF file
E-SVF	ECO SVF
E-NET	ECO netlist

If the designs are being read into a specific work library or library name (not the default), the E-RTL needs to be loaded into the same library name as the O-RTL. This applies to all Tcl scripts in the Formality ECO RTL flow.

For example, consider that the Tcl script uses the following command to read in the O-RTL:

```
read_verilog -r ./orig_rtl/top.v -work_library FM_REF
```

The script needs to use the same `FM_REF` work library when reading in the E-RTL:

```
read_verilog -r ./eco_rtl/top.v -work_library FM_REF
```

See Also

- Match ECO Regions Script
- ECO Synthesis Script
- Create ECO Patch Script
- Confirm ECO Patch Script

Feedback

Match ECO Regions Script

The Formality ECO match-ECO-regions script (also known as the pre-synthesis script) sets up three-way matching between the original RTL (O-RTL), the ECO RTL (E-RTL), and the original netlist (O-NET). This script creates ECO region information for the Design Compiler or Fusion Compiler tool during ECO RTL synthesis. ECO regions are matched boundaries that the Formality ECO tool finds across the original RTL, ECO RTL, and original netlist designs that isolate the ECO changes. The logic outside these ECO regions is equivalent and does not need to be patched.

The Formality ECO transcript log indicates the number of ECO regions to replace in the original netlist. It can also identify rewiring ECOs where no new logic is required. If there are no matched ECO regions caused by a rewire ECO, then the ECO synthesis and verification steps are optional, and can be skipped. The transcript log then indicates that you can proceed directly to the final confirm script.

Example 1 shows the match ECO regions script. See Table 1 for the terminologies used in the script.

Example 1 Match ECO Regions Script

```
set synopsys_auto_setup true

## Read in O-SVF
set_svf "./DC/orig.svf"

## Read libraries
read_db some_tech_lib.db

## Generate and save the E-RTL container first
## Read in the E-RTL
read_verilog -r ./eco_rtl/top.v
set_top top

## Save the E-RTL container for later use
write_container -replace -r ./ECO_WORK/ertl.fsc

## Remove the E-RTL container to verify O-RTL and O-NET
remove_container r

## Read the O-RTL
read_verilog -r ./orig_rtl/top.v
set_top top

## Save O-RTL container before any SVF modification
write_container -replace -r ./ECO_WORK/ortl.fsc

## Read the O-NET
read_verilog -i ./DC/orig.gates.v
set_top top
```

Feedback

```
## Save the implementation container for later use
write_container -replace -i ./ECO_WORK/onet.fsc

## Uncomment the following line for UPF low-power flow
## This constrains isolation and retention cells
# constrain_low_power_intent  $impl

## Read in the O-RTL that is unaffected by SVF processing
read_container -container ortl ./ECO_WORK/ortl.fsc

## Read in the E-RTL container created previously
read_container -container ertl ./ECO_WORK/ertl.fsc

## Perform any setup needed to verify O-RTL and O-NET
## Apply the necessary setup to all applicable containers
## Include commands such as set_constant, set_user_match, set_dont_verify
# set_constant -type port r:/WORK/top/test_en 0
# set_constant -type port i:/WORK/top/test_en 0
# set_constant -type port ortl:/WORK/top/test_en 0
# set_constant -type port ertl:/WORK/top/test_en 0

# set_dont_verify r:/WORK/test/foo_reg
# set_dont_verify i:/WORK/test/foo_reg
# set_dont_verify ortl:/WORK/test/foo_reg
# set_dont_verify ertl:/WORK/test/foo_reg

## Check whether the setup is applied to all needed containers
report_setup_status -all

## This applies SVF to r
preverify

## For debugging purposes replace “preverify” above with the lines below.

## The resulting verification of O-RTL and O-NET should have no failures.
## This will also perform the same function as “preverify”.
##
# set verification_effort_level super_low
# verify

## Set up containers to find ECO regions
set_orig_reference ortl
set_orig_implementation i
set_eco_reference ertl

## Match the ECO regions boundaries
match_eco_regions

## The command write_eco_regions creates these files:
##    a.) fm_eco_region.group.tcl
##    b.) fm_eco_region.data.tcl
##    c.) fm_eco_patched_orig.svf if O-SVF contains checkpoint guidance
```

```
write_eco_regions -replace

if { ![file exists fm_eco_region.group.tcl ] } {
 echo "No functional difference detected for this ECO."
}

quit
```

For this pre-synthesis script, use the `set_svf` command to read the original SVF files as-is. Do not include any supplemental SVF files associated with the ECO RTL, such as eco_change.svf, which is explained in Generating Supplemental SVF for ECO Changes.

If the original SVF contains one or more accepted `guide_checkpoint` guidance commands, the Formality ECO tool creates a new SVF file named fm_eco_patched_orig.svf along with the data and group region scripts while running the `write_eco_regions` command. Use this patched SVF file instead of the original SVF file in the *Confirm ECO Patch* step of the ECO flow.

If a design uses the UPF low-power flow, do not include any `load_upf` commands in this Formality ECO Tcl script. The RTL does not have any UPF constructs. For the original netlist, the Formality ECO tool constrains isolation cells and retention cells using the `constrain_low_power_intent` command.

Note:

The Formality ECO flow does not support ECO changes that cross power-domain boundaries and low-power designs using power switches.

The following Formality ECO commands use either container names or specified designs within a container:

- `set_orig_reference`
- `set_orig_implementation`
- `set_eco_reference`
- `set_eco_implementation`

For example, to specify the top-level design within container `ortl` as the original reference design, use the `set_orig_reference ortl` command. However, to specify a certain subdesign within the `ortl` container, use the `set_orig_reference ortl:/WORK/subdesign_name` command instead.

If there are problems with generating the ECO region files, replace the `preverify` command with the `verify` command and stop the script to debug the failures. Ensure this verification succeeds before proceeding with the rest of the script. The verification of the original RTL and original netlist should be successful before introducing changes from an ECO.

Feedback

If successive or incremental ECOs are done on the netlist, consider using the following command and option to clearly design this as the first `ECO` or `ECO1`:

```
match_eco_regions -name <name>
```

This defines a naming strategy for the entire flow. If you do not specify the `-name` option, the `fm_eco_region.patch.tcl` patch uses the `ECO_cXXX`, `ECO_nXXX`, `ECO_pXXX` naming convention for cells, nets, and ports. If you specify the `-name ECO1` option, the `fm_eco_region.patch.tcl` patch uses the `ECO1_cXXX`, `ECO1_nXXX`, `ECO1_pXXX` naming convention for cells, nets, and ports.

The `-include_unread_compare_points` option of the `match_eco_regions` command specifies whether to include unread compare points when comparing the ECO reference to the original reference designs. When using this option, choose from the following values:

- `all`: Use this value to include all unread compare points in the ECO reference
- `matched`: Use this value to include matched unread compare points in the ECO reference
- `none`: Use this value to not include any unread compare points in the ECO reference

Note:

To use `all` or `matched`, you must enable one of the following variables in all four stages of the Formality RTL ECO flow:

- `verification_verify_unread_compare_points`
- `verification_verify_matched_unread_compare_points`

Ensure that the following conditions are met after running the pre-synthesis script:

- No errors are reported in the Match ECO Regions script log.
- The Formality ECO tool finds logic differences due to the ECO. Failing compare points occur due to the ECO, and not because of an incorrect setup.
- The fm_eco_region.group.tcl file is generated.
- The fm_eco_region.data.tcl file is generated.

See Also

- ECO RTL Flow Patch Generation Scripts
- ECO Synthesis Script
- Create ECO Patch Script
- Confirm ECO Patch Script

Feedback

ECO Synthesis Script

The ECO synthesis script sets up the Design Compiler or Fusion Compiler Tcl script and synthesizes the E-RTL as indicated using the ECO-region information created.

Both the Design Compiler and Fusion Compiler tools have an ECO mode that automatically sets up these tools for ECO RTL synthesis. This mode is initiated by using the `set_fm_eco_mode -region filePath` command. You must invoke this command before the `analyze`, `elaborate`, or `read_file` command in your script. The `set_fm_eco_mode -region filePath` command performs the following tasks:

- Applies the ECO region group file at the appropriate time during synthesis
- Disables presto optimizations
- Disables constant propagation across hierarchical boundaries when boundary optimization is disabled
- Disables register replication
- Disables checkpoints in the Fusion Compiler tool
- Prints a summary of the acceptance status of ECO regions

Note:

When using ECO mode in the Design Compiler tool, you must use the `set_verification_top` command to link the design.

Example 2 shows the ECO synthesis script. See Table 1 for the terminologies used in the script.

Example 2 ECO Synthesis Script

```
set search_path ". ./eco_rtl ./lib"

set link_library "some_tech_lib.db"
set target_library "$link_library"

## This is the E-SVF
set_svf "./eco.svf"

sh mkdir work
define_design_lib WORK -path ./work

## Use the ECO Mode in Design Compiler
## Perform these steps before reading in the ECO RTL:
##
## Ensure that Design Compiler creates guide_hier_map SVF guidance
set_app_var hdlin_enable_hier_map true
##
## Source the ECO region file
```

Feedback

```
set_fm_eco_mode -region ./fm_eco_region.group.tcl


analyze -format verilog ./eco_rtl/top.v
elaborate top

## Ensure that you use the set_verification_top command
set_verification_top


## The remainder of the Design Compiler script remains unchanged
## except the "write …" at the end.
## Formality ECO uses parts of the SVF file and resulting netlist

check_design

create_clock clk -period 5
  set_input_delay 0.1  [all_inputs]
  set_output_delay 10  [all_outputs]

compile_ultra

change_names -rules verilog -hier

## This is E-NET
write -hier -format verilog -out ./eco_gates.v

set_svf -off

quit
```

Confirm the following conditions after running the ECO synthesis script:

- The Design Compiler tool does not report any errors after using the fm_eco_region.group.tcl script.
- E-NET and E-SVF are generated.

See Also

- ECO RTL Flow Patch Generation Scripts
- Match ECO Regions Script
- Create ECO Patch Script
- Confirm ECO Patch Script

Feedback

Create ECO Patch Script

The create ECO patch script sets up the Formality ECO tool to match the E-RTL and E-NET and generate the Formality ECO patch (internal patch file) for the ECO regions. It also creates the ECO region SVF. This script is known as the post-synthesis script because it is run just after ECO synthesis.

If you want to make successive ECOs on a netlist, use the `-prefix` option to clearly designate the first ECO:

```
create_eco_patch -prefix <name>
```

The `-prefix` option only controls the naming of new objects in the patch script. If no `-prefix` is specified, the tool follows the `match_eco_regions` command naming strategy.

Example 3 shows the create ECO patch script. See Table 1 for the terminologies used in the script.

Example 3 Create ECO Patch Script

```
set synopsys_auto_setup true

## Read E-SVF
set_svf "./eco.svf"

read_db "some_tech_lib.db"

## Read E-RTL
## Do not use the previously generated ERTL container
read_verilog -r ./eco_rtl/top.v
set_top top

## Read E-NET
read_verilog -i ./eco_gates.v
set_top top

## Uncomment the following line for UPF low-power flow
# constrain_low_power_intent  $impl

## Reload O-NET in separate container
read_container -container onet ./ECO_WORK/onet.fsc

## Perform any setup needed to verify E-RTL and E-NET
## Apply the necessary setup to all applicable containers
## Include commands such as set_constant, set_user_match, set_dont_verify
# set_constant -type port r:/WORK/top/test_en 0
# set_constant -type port i:/WORK/top/test_en 0
# set_constant -type port onet:/WORK/top/test_en 0

# set_dont_verify r:/WORK/test/foo_reg
# set_dont_verify i:/WORK/test/foo_reg
```

Feedback

```
# set_dont_verify onet:/WORK/test/foo_reg

## Check that setup is applied to all needed containers
report_setup_status -all

set_eco_ref r
set_eco_imp i
set_orig_imp onet;

source fm_eco_region.data.tcl
current_container r
source fm_eco_region.group.tcl

## Perform "match" (or replace with "verify" for debugging purposes)
## to find ECO regions, create patch file, and create ECO region SVF file

if { [ match ] } {
  ## Generate fm_eco_region.patch.tcl and fm_eco_region.svf
  create_eco_patch -replace

  ## Report information about the ECO patch
  report_eco_impact -all > report_eco_impact.txt
}

quit
```

In this patch generation step, you use the `set_svf` command to read the E-SVF because it came out of the Design Compiler tool during E-RTL synthesis. The E-SVF is applied only to the E-RTL. Do not use the previously generated `ertl`.

If a design uses the UPF low-power flow, do not include any `load_upf` commands in the script. The RTL does not have any UPF constructs. For the netlist, use the `constrain_low_power_intent` command to constrain isolation and retention cells for ECOs.

Note:

The ECO RTL flow does not support low-power designs using power switches.

This script verifies the ECO netlist and creates several files including the internal ECO region patch file, the ECO region SVF file, and sometimes additional setup files for multibit designs. There are two setup files generated for multibit designs during this step:

- fm_eco_region.confirm_setup.tcl, which should be included during the next step of confirming the patch
- fm_eco_region.netlist_flow_setup.tcl, which should be used in the ECO netlist flow

The fm_eco_region.svf file contains SVF guidance information for the affected ECO regions only. If there are problems with generating these files, replace the `match` command with the `verify` command and investigate any failing compare points,

Feedback

especially for setup issues. Ensure that this verification succeeds before continuing with the ECO RTL flow.

The post-synthesis script in Example 3 includes the optional use of the `report_eco_impact` command in match or verify mode after successfully generating an ECO patch. The `report_eco_impact` command reports the following:

- Impact of the ECO on the scan chain
- Registers that are left without readers after the ECO
- ECO patch size

Confirm the following after running the post-synthesis script:

- No errors are generated by the `create_eco_patch` command
- fm_eco_region.patch.tcl file is generated
- fm_eco_region.svf file is generated

The fm_eco_region.patch.tcl patch file generated in this step modifies the original netlist to compare it against the E-RTL. This is not the final netlist patch file for the IC Compiler, IC Compiler II, Fusion Compiler, or Design Compiler tool.

See Also

- ECO RTL Flow Patch Generation Scripts
- Match ECO Regions Script
- ECO Synthesis Script
- Confirm ECO Patch Script

Confirm ECO Patch Script

The confirm ECO patch script confirms whether the Formality ECO patch is correct by verifying the E-RTL against the O-NET that is patched inside the Formality tool before exporting the ECO netlist patch to IC Compiler, IC Compiler II, Fusion Compiler, or Design Compiler.

This script requires multiple SVF files. The O-SVF file in this script might include a supplementary SVF file that corrects line number changes between the O-RTL and the E-RTL. The Formality ECO tool names objects such as adders, subtractors, multipliers, and other operators based on their line number in the RTL. It is important to maintain synchronicity between the O-RTL and the E-RTL because SVF guidance information relies on the accuracy of these object names.

To create this supplemental file, use the `fm_eco_to_svf` utility. Redirect the output of this command to the eco_change.svf file. For example, you can generate the supplementary file as follows:

```
fm_eco_to_svf ./rtl  ./eco_rtl  > eco_change.svf
```

Note:

For more information, see Generating Supplemental SVF for ECO Changes.

If the Formality ECO tool created the fm_eco_patched_orig.svf file due to checkpoints, use this file instead of the original SVF file along with the eco_change.svf file.

Use the `set_svf -append` command to read in the ECO region SVF file. The ECO region SVF is applied to the E-RTL inside the ECO regions.

If a design uses the UPF low-power flow, read in both the reference and implementation UPF files using the `load_upf` commands as indicated in the example script in Example 4.

If the post-synthesis step created the fm_eco_region.confirm_setup.tcl file for handling multibit designs, source this setup after running the `preverify` command and before running the `match` or `verify` command.

Example 4 shows the script to confirm the ECO patch. See Table 1 for the terminologies used in the script.

Example 4 Confirm ECO Patch Script

```
## Generates "eco.edits.tcl" file to patch O-NET using IC Compiler,
## IC Compiler II, Fusion Compiler, and Design Compiler
set synopsys_auto_setup true

## Use fm_eco_patched_orig.svf if created during the create ECO regions
## script due to one or more guide_checkpoint guidance in original SVF
## Otherwise, use original SVF. Include supplementary SVF (if needed)

if {[file exists fm_eco_patched_orig.svf]} {
   set_svf "fm_eco_patched_orig.svf eco_change.svf"
} else {
   set_svf "./DC/orig.svf eco_change.svf"
}

## Read in ECO region SVF file (generated at the same time as ECO patch)
set_svf -append "fm_eco_region.svf"

## Read libraries
read_db "some_tech_lib.db"

## Read E-RTL
## Do not use the previously generated ERTL container
read_verilog -r eco_rtl/top.v
set_top top
```

Feedback

```
## Read O-NET or read O-NET container created previously
# read_verilog -i DC/orig_gates.v
# set_top top
read_container -i ./ECO_WORK/onet.fsc

## Apply ECO region group to E-RTL and internal patch to O-NET
current_container r
source fm_eco_region.group.tcl

current_container i
source fm_eco_region.patch.tcl

## For UPF low-power flow load the original reference and
## original implementation UPF files
# load_upf -r reference.upf
# load_upf -i implementation.upf

## Perform any necessary setup to verify E-RTL and patched O-NET
## Include commands such as set_constant, set_user_match,
## set_dont_verify.
# set_constant -type port r:/WORK/top/test_en 0
# set_constant -type port i:/WORK/top/test_en 0

# set_dont_verify r:/WORK/test/foo_reg
# set_dont_verify i:/WORK/test/foo_reg

## Check that setup is applied to all needed containers
report_setup_status -all

## Automatic setup for multibit designs
if {[file exists fm_eco_region.confirm_setup.tcl]} {
  preverify
  source ./fm_eco_region.confirm_setup.tcl
}

match
verify

## Write out final ECO Tcl script for use in IC Compiler (IC Compiler II
 or Design Compiler or Fusion Compiler)
write_edits -replace eco.edits.tcl

quit
```

See Also

- ECO RTL Flow Patch Generation Scripts
- Match ECO Regions Script

- ECO Synthesis Script
- Create ECO Patch Script

3

Formality ECO Netlist Flow

The Formality ECO netlist flow requires the following inputs:

- A pre-ECO netlist
- A post-ECO netlist (previously patched)
- The target netlist

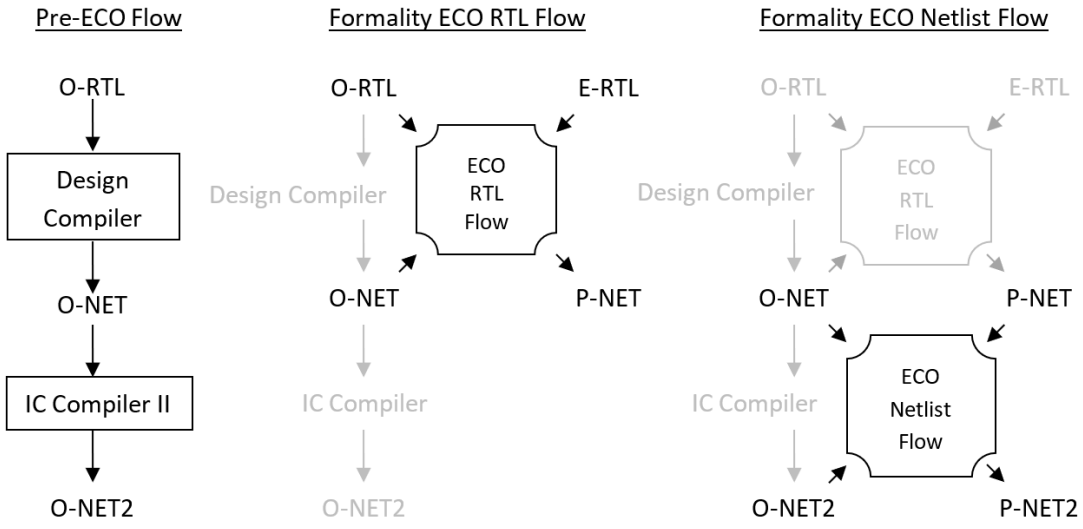
Note:

The target netlist should be functionally equivalent to the pre ECO netlist.

The Formality ECO netlist flow produces a Tcl patch file that can be applied to the target netlist to make it functionality equivalent to the post ECO netlist. This process is accomplished without the intermediate use of synthesis.

Figure 4 illustrates how the Formality ECO netlist flow fits into the Formality ECO flow:

Figure 4 Formality ECO Flow Representation



The pre ECO flow shows a simplified design flow before any ECO changes. This flow begins with the original RTL (O-RTL). The Design Compiler tool produces the original

netlist (O-NET), and the IC Compiler or IC Compiler II tool produces the optimized netlist (O-NET2). The netlist O-NET2 is functionally equivalent to the O-RTL.

The Formality ECO RTL flow begins with a new version of the RTL that contains changes for the ECO (E-RTL). The ECO RTL flow produces a Tcl patch script, which implements the E-RTL changes on O-NET to become the patched netlist (P-NET).

The Formality ECO netlist flow propagates the patch from the previous step to produce a patched optimized netlist (P-NET2) derived from the optimized netlist (O-NET2).

The goal of the Formality ECO netlist flow is to create the P-NET2 patched netlist without having to restart the entire ECO process using E-RTL. This is convenient when you do not have access to the O-RTL or E-RTL and need to start the process using the synthesized netlist. Also, this saves time in creating an ECO on the final netlist.

You can use the Formality ECO netlist flow when the P-NET is generated using the Formality ECO RTL flow, or when the P-NET is generated manually. Using the Formality ECO RTL flow results in generating additional setup files for the Formality ECO Netlist flow, for example in handling multibit designs.

Table 2 includes the terminologies used in the Formality ECO netlist flow with their descriptions.

Table 2 Netlist Flow Terminologies and Descriptions

Terminology	Description
O-RTL	Original RTL
E-RTL	Version of the RTL containing changes for the ECO
O-NET	Original netlist
P-NET	Patched netlist
O-NET2	Optimized netlist
P-NET2	Patched optimized netlist

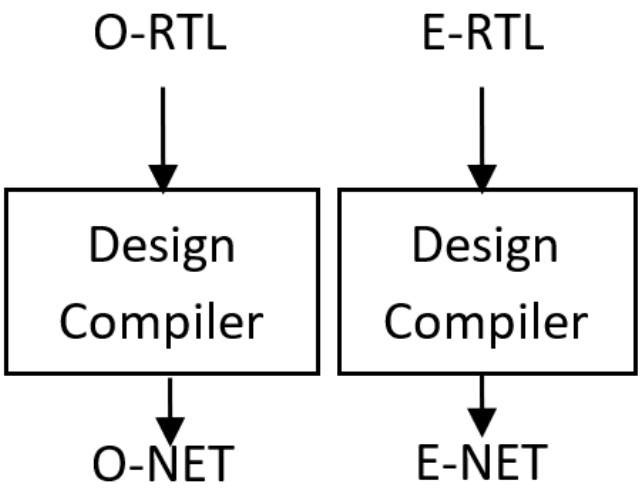
Netlists resulting from flows unrelated to the Formality ECO RTL flow are not recommended for the Formality ECO netlist flow.

Figure 5 illustrates an ECO flow that is not recommended because the netlists are synthesized from two different RTL sources.

Feedback

- The O-NET and E-NET are synthesized from two different RTL sources, and are not ideal for use in the Formality ECO netlist flow.
- Sequential, multibit, and datapath differences impact the patch and verification of the patched netlist.

Figure 5 Unsupported ECO Netlist Flows



Note:

Use the Formality ECO RTL flow or the Formality interactive ECO (Formality Ultra) flow instead to generate the ECO patch, which modifies the O-NET to be functionally equivalent to the E-RTL.

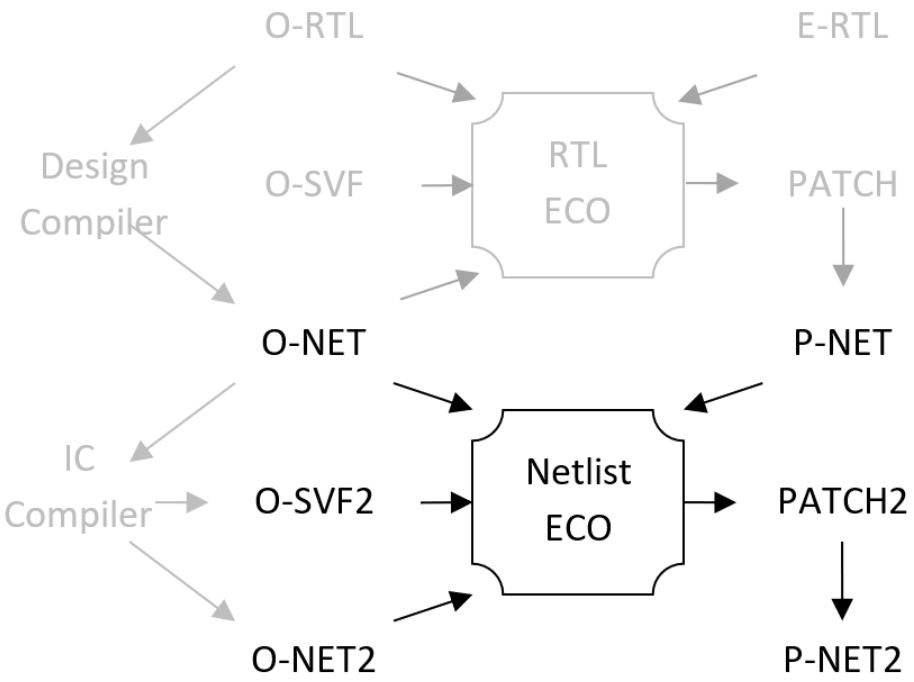
ECO Netlist Flow Summary

The Formality ECO netlist flow is similar to the RTL flow. However, there is no synthesis involved. You can generate the internal ECO patch using just one Formality ECO session. A second Formality ECO session confirms the functionality of the internal patch and generates the final ECO Tcl script for the IC Compiler II tool. This set of Tcl commands modifies the target netlist O-NET2 inside the IC Compiler II tool to be functionally equivalent to P-NET.

Figure 6 illustrates the typical Formality ECO netlist flow. See Table 2 for terminologies used in the ECO netlist flow..

Feedback

Figure 6 Formality ECO Netlist Flow



The netlist flow is based on the successful verification of O-NET2 against O-NET before proceeding forward to creating a patch (PATCH2). PATCH2 is applied to O-NET2 to create P-NET2.

In most cases, there is no SVF file (O-SVF2) from the IC Compiler II tool. If the verification from O-NET to O-NET2 passes without the SVF file, then the SVF is not necessary during the ECO netlist flow.

The Formality ECO Tcl script might need setup information to perform the following tasks:

- disable scanning
- set up clock gating
- perform additional setup required

If the Formality ECO RTL flow is used to generate the P-NET, it sometimes produces a formality_eco_region.netlist_flow_setup.tcl setup file during the post-synthesis step. This setup file involves multibit designs and should be sourced during the ECO netlist flow just before the `match_eco_regions` command.

The netlist flow issues a formality_eco_region.confirm_setup.tcl setup file for handling multibit designs. Source this setup file during the Confirm ECO Patch step after running the `preverify` command and before running the `match` or `verify` command.

For low-power designs (UPF flow), the Formality ECO netlist flow uses the `constrain_low_power_intent` command to constrain isolation cells and retention cells, allowing Formality ECO to work on the design. The netlist flow does not currently support low-power designs with power switches.

See Also

- Confirm ECO Patch Script for the Netlist Flow
- Create ECO Patch Script for the Netlist Flow

Create ECO Patch Script for the Netlist Flow

The following Formality ECO Tcl script illustrates how the Formality ECO netlist flow is used to generate an internal Formality ECO patch file. This patch file is later confirmed and converted into a Tcl patch file for use in the IC Compiler or IC Compiler II tool. This single script identifies ECO regions and generates the internal Formality ECO patch.

Example 5 Create ECO Patch Script for the Netlist Flow

```
## Read the SVF from IC Compiler or IC Compiler II only if necessary
## to verify O-NET and O-NET2 are equivalent
# set_svf O-SVF2

## Read in libraries
read_db -technology_library my_library.db

## Read in original, pre-ECO, netlist (O-NET) created by Design Compiler
## or Fusion Compiler
read_verilog -netlist –r O-NET.v
set_top top_of_design

## Uncomment the following line if using UPF flow
# constrain_low_power_intent $ref

## Write out reference container before any SVF modification
write_container -replace -r O-NET.fsc

## Read in original, pre-ECO, netlist (O-NET2) created by IC Compiler or
 IC Compiler II
read_verilog -netlist –i O-NET2.v
set_top top_of_design

## Uncomment the following line if using UPF flow
# constrain_low_power_intent $impl
```

```
## Read in ECO netlist (P-NET) that has the patch already applied to it
read_verilog -netlist -container P-NET P-NET.v
set_top top_of_design

## Uncomment the following line for UPF flow
# constrain_low_power_intent P-NET:/WORK/top_of_design

## Reload O-NET into container that is not modified by SVF processing
read_container -container O-NET O-NET.fsc

## Perform any setup needed to verify O-NET and O-NET2
## Apply the necessary setup to all applicable containers
## Include commands such as set_constant, set_user_match, set_dont_verify
## Examples:

# set_constant -type port r:/WORK/top/test_en 0
# set_constant -type port i:/WORK/top/test_en 0
# set_constant -type port ONET:/WORK/top/test_en 0
# set_constant -type port PNET:/WORK/top/test_en 0

## Ensure that the verification of O-NET and O-NET2 succeeds before
 continuing
if { ![verify] } {
   echo “”
   echo “Cannot continue due to failing verification…”
   return }

## Generating the patch requires the following containers
set_orig_reference O-NET
set_orig_implementation i
set_eco_reference P-NET
set_eco_implementation P-NET
## Perform any necessary setup for multibit
## The Formality ECO RTL flow might have produced the following setup
 file
if {[file exists Formality_eco_region.netlist_flow_setup.tcl]} {
   source ./Formality_eco_region.netlist_flow_setup.tcl
}

## Find the regions to be patched
match_eco_regions

## Generate internal Formality ECO patch file named
 Formality_eco_region.patch.tcl
create_eco_patch -replace

## Report patch size information
report_eco_impact -all > report_eco_impact.txt
quit
```

The following Formality ECO commands use container names or specified designs within a container:

- `set_orig_reference`
- `set_orig_implementation`
- `set_eco_reference`
- `set_eco_implementation`

To specify the top-level design within `ortl` as the original reference design, use the `set_orig_reference ortl` command. To specify a certain subdesign within the `ortl` container, use the `set_orig_reference ortl:/WORK/subdesign_name` command.

Check the following after executing the generate script:

- Verification is successful
- The formality_eco_region.confirm_setup.tcl file might be created for handling multibit designs downstream
- The internal formality_eco_region.patch.tcl patch script is generated. This is later used to modify O-NET2 in the Formality tool to be functionally equivalent to P-NET.

Using the `write_eco_regions` command for the netlist flow is optional and can be used to break the script into two parts. The first part executes the `match_eco_regions` command and the second part executes the `create_eco_patch` command.

The netlist flow does not generate a formality_eco_region.group.tcl file. This is created only in the Formality ECO RTL flow.

Example 5 includes the optional use of the `report_eco_impact` command, which reports the following:

- impact of the ECO on the scan chain
- registers left without readers after the ECO
- the ECO patch size

See Also

- ECO Netlist Flow Summary
- Confirm ECO Patch Script for the Netlist Flow

Feedback

Confirm ECO Patch Script for the Netlist Flow

The Formality ECO Tcl script in Example 6 is used to confirm the internal Formality ECO patch. The confirm step verifies the correctness of the internal Formality ECO patch and writes out the ECO edit Tcl file, which is the netlist patch for the IC Compiler, IC Compiler II, Design Compiler, or Fusion Compiler tool. This script creates PATCH2, which is the set of Tcl commands needed to modify O-NET2 to become P-NET2.

When generating the internal Formality ECO patch earlier, the tool might create a formality_eco_region.confirm_setup.tcl setup file for handling multibit designs. Source this setup file during this confirm step after running the `preverify` command and before running the `match` or `verify` command.

Example 6 Confirm ECO Patch Script for the Netlist Flow

```
## Read in SVF from IC Compiler only if necessary to verify that O-NET
## and O-NET2 are equivalent
# set_svf O-SVF2

## Read in libraries
read_db -technology_library my_library.db

## Read in the ECO netlist that was previously patched in Design
 Compiler
## or Fusion Compiler
read_verilog -netlist -r P-NET.v
set_top top_of_design

## Read in original netlist (pre-ECO) from IC Compiler or IC Compiler II
read_verilog -netlist -i O-NET2.v
set_top top_of_design

## Read in Formality ECO patch file to modify O-NET2 to be functionally
## equivalent to P-NET
current_container i
source Formality_eco_region.patch.tcl

## Uncomment the next two lines if needed for UPF flow
# load_upf -r O-NET.upf
# load_upf -i O-NET2.upf

## Preform any setup needed for verification
# set_constant $ref/…
# set_constant $impl/…

## Check if setup file was created when Formality ECO patch was generated
## This setup helps with processing multibit designs
if {[file exists Formality_eco_region.confirm_setup.tcl]} {
  preverify
  source ./Formality_eco_region.confirm_setup.tcl
}
```

Feedback

```
## Verification of the patched netlist O-NET2 against P-NET
## should succeed
if { ![verify] } {
   echo ""
   echo "Cannot continue due to failing verification…"
   return }

## Write out ECO Edit File for guiding IC Compiler or IC Compiler II in
## modifying O-NET2 to become P-NET2. This Tcl file is PATCH2.
write_edits -replace eco.edits.tcl

quit
```

Check the following conditions after you run the confirm script:

- Verification is successful
- The eco.edits.tcl file is generated

The eco.edits.tcl file is the final Formality ECO netlist patch file PATCH2. It contains the set of Tcl commands needed to modify O-NET2 inside the IC Compiler, IC Compiler II, Design Compiler, or Fusion Compiler tool to become P-NET2.

It is recommended that you verify P-NET2 from IC Compiler or IC Compiler II against P-NET to confirm the accuracy of the changes. While performing the verification, include the formality_eco_region.confirm_setup.tcl setup file if it is created during the netlist flow.

See Also

- ECO Netlist Flow Summary
- Create ECO Patch Script for the Netlist Flow

A

Miscellaneous

This topic includes links to all custom scripts applicable to the RTL and netlist flows.

See Also

- Formality ECO RTL Flow Using Wrapper Script
- Post ECO Verification
- Generating Supplemental SVF for ECO Changes

Formality ECO RTL Flow Using Wrapper Script

The purpose of the wrapper script is to automatically generate the scripts needed for the ECO flow. This script is available under <installation>/fm_auto_eco_wrapper.pl and is used to simplify the setup and use of the Formality ECO RTL flow. These scripts are executed from the same directory as the wrapper script.

Create a minimal set of Formality Tcl scripts and use a configuration file to point to them. The wrapper script then generates all necessary Tcl scripts and guides you through the ECO RTL flow.

Note:

This method does not support the UPF low-power design flow.

The wrapper takes a configuration file that lists the location of the template scripts as input. These template scripts are used to generate other scripts that need to be executed to generate the ECO patch. The wrapper script outlines each step that you need to execute to generate the patch.

The following are the advantages of the wrapper script:

- You have the final control over the scripts generated.
- The wrapper hides the details of intermediate files and commands required to generate the patch.
- You can use the `-debug` option to save session files, reports, containers and so on during each run.

Create the configuration file as follows:

```
FM_ENGINE = "<path>"
ORIG_SCRIPT = "<path>"
ERTL_LOAD_SCRIPT = "<path>"
ECO_SCRIPT = "<path>"
CONFIRM_SCRIPT = "<path>"
```

Note:

The use of white space around names is optional.

Table 3 Configuration File Elements and Descriptions

Element	Description
FM_ENGINE	This is the path to the fm_shell executable. It is recommended to use the newest version of Formality ECO available. FM_ENGINE = "/u/snps/release/formality/bin/fmeco_shell"
ORIG_SCRIPT	This is the path to the Formality script that is set up to verify the O-RTL with the O-NET. The script must have a separate line for the set_svf <original_svf>, match, and verify commands so that the wrapper can use it as a template to generate other subscripts as follows:
ERTL_LOAD_SCRIPT	This is the path to a Formality Tcl script that is used to read and set the E-RTL as the top design using the r container. ERTL_LOAD_SCRIPT= "ECO/fm_eco_rtl_load.tcl"
ECO_SCRIPT	This is the path to a Formality Tcl script required to verify the E-RTL to the E-NET. This script has to be created before synthesizing the E-RTL. This script loads the E-RTL as the reference and the E-NET as the implementation design. It must contain the set_svf <eco_svf>, match, and verify commands though the eco_svf file does not exist (The E-NET and eco_svf files are generated during synthesis). ECO_SCRIPT = "ECO/fm.tcl"
CONFIRM_SCRIPT	This is the path to a Formality Tcl script required to verify the E-RTL to O-NET. The SVF files needed in this script are the original SVF file and any supplementary SVF files needed to fix line number changes, or other changes, incurred by the ECO RTL modifications (See Generating Supplemental SVF for ECO Changes). The automated wrapper modifies this script to verify the E-RTL against the patched O-NET to confirm that the ECO patch rectifies the O-NET.

Feedback

Note:

Ensure that the scripts can run under the same directory where the wrapper script is invoked. All intermediate data files and scripts are stored in the ./ECO_WORK directory that is created automatically.

See Also

- Wrapper Script and Output

Wrapper Script and Output

Example 7 shows how to run the wrapper script:

Example 7 Wrapper Script

```
(unix) % cat config
## Specify the path to the Formality executable
FM_ENGINE = /u/formal/release/formality/bin/fmeco_shell

## Original FM Tcl script including using original SVF file
ORIG_SCRIPT = fm.orig.tcl

## New FM Tcl script that would be needed to read in the ECO RTL into the
 "r" container and perform "set_top"
ERTL_LOAD_SCRIPT=fm_ertl_load.tcl

## FM Tcl script for validating intermediate ECO RTL synthesis results
ECO_SCRIPT = fm.eco.tcl

## FM Tcl script for verifying ECO RTL versus the original netlist
CONFIRM_SCRIPT=fm.confirm.tcl

(unix) % fm_auto_eco_wrapper.pl -c config
```

Example 8 shows the output of the wrapper script in Example 7.

Example 8 Wrapper Script Output

```
Reading config file ... done
Generating scripts ...

See "auto_eco.wrapper.output" with instructions on how to generate the
 ECO-patch:

----------- auto_eco.wrapper.output -----------------

Run the following scripts:

[1] Generate ECO-region data:
```

```
   % /u/formal/release/formality/bin/fmeco_shell -work ./.TEMP -f
 ECO_WORK/fm_pre_synth.tcl | tee ECO_WORK/fm_pre_synth.tcl.log

[2] Synthesize E-RTL:
    a. Add the following line to E-RTL DC synthesis script right after
 design is linked
       source ECO_WORK/fm_eco_region.group.tcl
    b. Synthesize to generate the E-Netlist

[3] Confirm (or edit if needed) that 'ECO_WORK/fm_post_synth.tcl' loads
 the [E-Netlist + SVF] generated in Step-2-b.

[4] Verify E-NET + Generate the ECO-patch:
   % /u/formal/release/formality/bin/fmeco_shell -work ./.TEMP -f
 ECO_WORK/fm_post_synth.tcl | tee ECO_WORK/fm_post_synth.tcl.log

[5] Confirm that the ECO-patch is correct:
   % /u/formal/release/formality/bin/fmeco_shell -work ./.TEMP -f
 ECO_WORK/fm_confirm_patch.tcl | tee ECO_WORK/fm_confirm_patch.tcl.log

If Verification in Step[5] SUCCEEDED, then use the generated
 "./fm_eco_edits.tcl" ECO-patch file for this ECO
```

1. Step1 runs the initially generated Formality ECO Tcl script as instructed by the wrapper script output log. This script creates containers and sets things up in memory between the original RTL, the ECO RTL, and the original netlist for three-way matching. It generates ECO region information for the Design Compiler tool.

   ECO regions are matched boundaries that the Formality ECO tool detects across the original RTL, ECO RTL, and original netlist designs that isolate the ECO changes. The logic cones outside the ECO regions are equivalent and do not need a patch.

   Review the transcript log during this pre-synthesis step. The Formality ECO tool indicates the number of ECO regions to replace in the original netlist. It also identifies rewiring ECOs where no new logic is needed. If there are no matched ECO regions and only rewiring ECOs, then the ECO synthesis and verification steps are optional and can be skipped. The transcript log indicates that you can proceed directly to the final confirm script.

   If the original SVF contains one or more accepted `guide_checkpoint` guidance commands, the Formality ECO tool creates a new fm_eco_patched_orig.svf SVF file along with the data and group region scripts while running the `write_eco_regions` command contained in the pre synthesis script. This SVF file is used automatically in the wrapper script instead of the original SVF file in the confirm step and final verification step of the flow.

2. Step 2 modifies the original Design Compiler Tcl script to synthesize the ECO RTL. This is done by inserting the source ECO_WORK/fm_eco_region.group.tcl command as indicated by the wrapper script log file just after the Design Compiler link command. Remove all incremental `compile_ultra` commands. Remove options for register

replication and any manual commands to create checkpoints. Run the Design Compiler tool using the modified Design Compiler Tcl script.

3. Step 3 visually confirms that the post-synthesis script loads the ECO RTL and the newly created ECO netlist and SVF files. Edit the Formality Tcl script if required.
4. Step 4 uses the automatically generated post-synthesis script to verify the ECO netlist and create files including the Formality ECO patch, the SVF file for the ECO regions, and sometimes setup files dealing with patching multibit designs.

   Sometimes, there are two setup files for multibit designs generated by ECO during this step. The fm_eco_region.confirm_setup.tcl file is used automatically by the wrapper script for *confirming* the patch. The fm_eco_region.netlist_flow_setup.tcl file should be used in the Formality ECO netlist flow.
5. Step 5 confirms the accuracy of the Formality ECO patch and writes out the final IC Compiler, IC Compiler II, or Design Compiler patch. This confirm patch script verifies the ECO RTL against the changes to the original netlist using the Formality ECO region patch file, the original SVF, the supplementary SVF for line number changes, and the newly created region SVF generated from the post synthesis script. After successful verification, this script generates the final fm_eco_edits.tcl ECO patch script for the IC Compiler, IC Compiler II, or Design Compiler downstream tools. This netlist patch or ECO patch file contains the netlist edit commands required to change the original netlist to make it functionally equivalent to the ECO RTL.

See Also

- Formality ECO RTL Flow Using Wrapper Script

Post ECO Verification

To verify the patched or ECO netlist from the IC Compiler, IC Compiler II, or Design Compiler tool against the ECO RTL, copy and modify the confirm script from …/ECO_WORK/fm_confirm_patch.tcl. Comment out the `source fm_eco_region.patch.tcl` and `write_edits -replace` lines in the eco.edits.tcl file.

Use the `set_svf` command to read in the original SVF and the `eco_change.svf` and `set_svf -append` commands to read the ECO patch SVF. If the Formality ECO tool creates the fm_eco_patched_orig.svf file to handle checkpoints, use it instead of the original SVF file. Always include the `source fm_eco_region.group.tcl` command in the reference container (ECO RTL) before running the match or verify command.

Example 9 shows a script to verify the final patched netlist from the IC Compiler, IC Compiler II, or Design Compiler tool:

Example 9 Script to Verify the Final Patched Netlist

```
## Formality Tcl Script to Verify Final Patched Netlist from IC Compiler,
 IC Compiler II, or Design Compiler

set synopsys_auto_setup true

## Use fm_eco_patched_orig.svf if created during Pre-Synthesis script due
## to one or more guide_checkpoint guidance in original SVF.  Otherwise,
## use original SVF. Include supplementary SVF (if needed).

if {[file exists fm_eco_patched_orig.svf]} {
   set_svf “fm_eco_patched_orig.svf eco_change.svf”
} else {
   set_svf “pe_orig.svf eco_change.svf”
}

## Read in ECO region SVF file (generated at the same time as ECO patch)
set_svf -append ECO_WORK/fm_eco_region.svf

## Read libraries
read_db "some_tech_lib.db”

## Read E-RTL
read_verilog -r ./eco_rtl/pe.rtl.v
set_top top

## Read patched O-NET from IC Compiler, IC Compiler II, or Design
 Compiler
read_verilog -i gates-patched.v
set_top top

## Apply ECO region group to E-RTL
current_container r
if {[file exists ECO_WORK/fm_eco_region.group.tcl]} {
  source ./ECO_WORK/fm_eco_region.group.tcl
}

## Perform any necessary setup to verify E-RTL and patched O-NET
## Include commands such as set_constant, set_user_match,
 set_dont_verify.
# set_constant -type port r:/WORK/top/test_en 0
# set_constant -type port i:/WORK/top/test_en 0

# set_dont_verify r:/WORK/test/foo_reg
# set_dont_verify i:/WORK/test/foo_reg

## Check that setup is applied to all needed containers
report_setup_status -all
```

```
## Automatic setup for multibit designs
if {[file exists fm_eco_region.confirm_setup.tcl]} {
  preverify
  source ./fm_eco_region.confirm_setup.tcl
}


verify

quit
```

Generating Supplemental SVF for ECO Changes

From the Formality application tree, use the `fm_eco_to_svf` program to create a supplemental SVF file to correlate line number differences between the ECO RTL and the original RTL. The Formality and Design Compiler tools use RTL line numbers to name objects such as multipliers, adders, subtractors, and other objects. So, the SVF and ECO RTL must have line number correlation to avoid inconclusive verifications.

The `fm_eco_to_svf` program is located in the following directory:

```
$SYNOPSYS/<PLATFORM>/fm/bin/fm_eco_to_svf
```

Using the program, specify the original RTL file and the ECO RTL file with the same name. Alternatively, specify the directories that contain the original and the ECO RTL files. The program finds matching file names and compares the contents of the files to generate supplemental SVF commands indicating changes in line numbers.

You must run this program for each modified RTL file, and compile the changes in an SVF file. In the following example, the name of the resulting SVF file is eco_change.svf. The first command creates the file, and the next command appends to the SVF file. Use the following syntax to run the program:

```
$ fm_eco_to_svf original/my_design.v eco/my_design.v > eco_change.svf
$ fm_eco_to_svf original/my_design_2.v eco/my_design_2.v >>
 eco_change.svf
```

The generated SVF file contains the `guide_eco_change` commands that describe the location of each modification in the RTL. Single lines are represented by a single line number, and multiple lines are represented by two line numbers that indicate the first line and the last line of the modified region.

The following examples show how line numbers are indicated. The commands identify the changes to the `mydsgn.v` design.

To insert lines 4 and 5 in the modified RTL, use the following example:

```
guide_eco_change -file {mydsgn.v} -type {insert} -original {4} -eco {4
 5}
```

To delete line 7 in the original RTL, use the following example:

```
guide_eco_change -file {mydsgn.v} -type {delete} -original {7} -eco {8}
```

To replace lines 12 through 14 in the original RTL with lines 13 and 14 in the modified RTL, use the following example:

```
guide_eco_change -file {mydsgn.v} -type {replace} -original {12 14} -eco
 {13 14}
```