Formality ECO User Guide

Version T-2022.03-SP4, September 2022

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

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 includes running Fusion Compiler or Design 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.

Feedback

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
  - Addition or removal of primary ports, datapath or control path changes.
- Preserves scan-chain: Formality ECO preserves scan chains of O-NET (See Scan Chain Preservation inCreate ECO Patch)
- Preserve clock-tree: Formality ECO attempts to reuse the clock of the O-NET as much as possible to minimize clock-tree disruption (See Clock Tree Preservation inCreate ECO Patch)
- Supports ECOs on low-power (UPF) designs (See UPF Support)
- Supports ECOs on designs with retimed components (Does not support if the ECO is in the fan-in or fan-out range of the retimed design portions)
- Supports SVF checkpoint flows using the Design Compiler Graphical and Fusion Compiler tools (See SVF Checkpoint Support)

The Formality ECO tool has the following advantages:

- Targeted-Synthesis: Synthesize only ECO-regions but with full constraints. Delivers fast turnaround time (TAT) without compromising on patch quality. (See ECO RTL Flow Using Targeted Synthesis)
- Maintains the multibit register mapping of the original design and supports multibit patch-in place (See Multibit Register Support).

- Generates an ECO patch for any Design Compiler or Fusion Compiler optimization
- Uses the Design Compiler or Fusion Compiler tool so that timing and other important considerations are used during ECO RTL synthesis
- No restrictions on ECO-synthesis: Formality ECO allows Fusion Compiler (or Design Compiler) to synthesis ECO-regions without any requirements to replay or mimic exactly the same optimizations as the original netlist. No intervention is required to add any new constraints to the synthesis recipe.

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.

Note:

The content of this document and the licensing requirements assume these versions or newer of the following Synopsys products:

- Formality version T-2022.03
- Design Compiler version S-2021.06-SP4
- Fusion Compiler version S-2021.06-SP4

Formality ECO Classifications

The Formality ECO tool consists of the following flows:

- Formality ECO RTL flow
- Formality ECO netlist flow

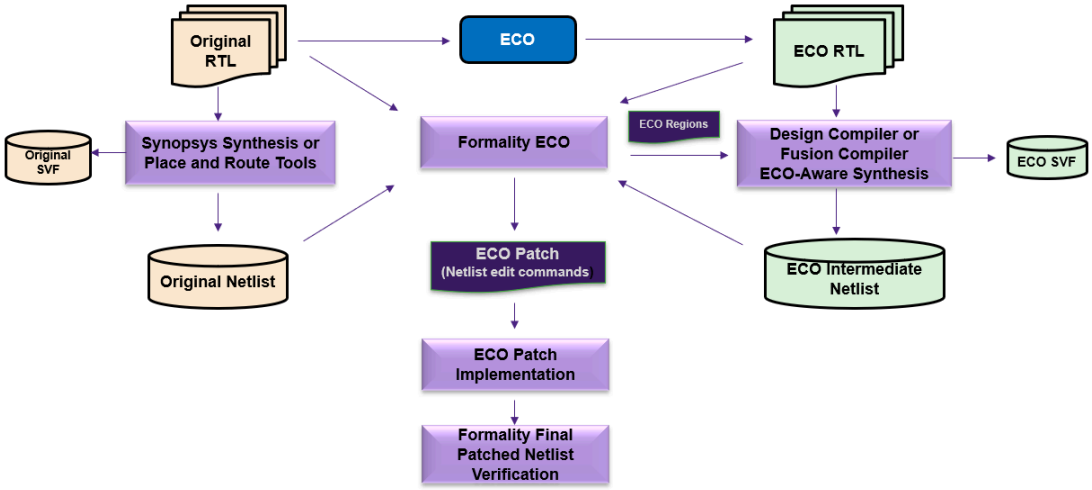
2

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



ECO RTL Flow Prerequisites

The Formality ECO RTL flow assumes the following criteria:

- The original netlist is synthesized using Design Compiler or Fusion Compiler.
- Ensure that the original netlist verifies successfully against the original RTL using the appropriate SVF guidance before proceeding through the remaining steps.
- Ensure that the RTL files, netlist, and SVF files are synchronized.

Feedback

- The original SVF file is available.
- The ECO RTL is simulated and considered to be the new golden source code.

Note:

It is recommended to use the latest version of Design Compiler and Fusion Compiler tools for ECO synthesis. You need not redo the original synthesis, it can be performed with older synthesis versions.

Table 1 Terminologies and Descriptions in the Formality ECO RTL Flow

Terminology	Description
ortl	Container with the original RTL
ertl	Container with the ECO RTL
r	Standard reference container
i	Standard implementation container
O-RTL	Original RTL (pre-ECO RTL), referred by `set_orig_reference` in the ECO RTL flow
E-RTL	ECO RTL (post-ECO RTL), referred by `set_eco_reference` in the ECO RTL flow
O-SVF	Original SVF file
O-NET	Original netlist to be patched, referred by `set_orig_implementation` in the ECO RTL flow
E-SVF	ECO SVF refers to the SVF generated from the E-RTL synthesis step
ECO region SVF	Contains the guidance information only for the ECO portion of the design
Patched orig SVF	SVF that is generated when there is a checkpoint in the design
E-NET	Intermediate ECO netlist generated during E-RTL synthesis referred by `set_eco_implementation` in the ECO RTL flow
P-NET	Final ECO patched netlist

Feedback

Formality ECO Input Requirements

The Formality ECO flows require the following input sources:

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

ECO RTL Flow Summary

The Formality ECO RTL flow requires the original RTL, the original netlist, original SVF, and the ECO RTL as inputs. The tool compares the ECO RTL against the original RTL to find the regions of change in the design.

The Design Compiler or Fusion Compiler tool synthesizes the ECO RTL to create an intermediate ECO implementation. ECO synthesis is done with the same script or constraints as the original synthesis. This ensures that timing and area quality of results (QoR) are considered by the Fusion Compiler or Design Compiler tool during ECO compile, thus providing the best quality patch.

Using the latest targeted synthesis technology, only the ECO portion of the E-RTL is synthesized instead of the entire design. This saves runtime during ECO synthesis especially if the full ECO synthesis runtime is very long. After the intermediate ECO netlist is synthesized, the Formality ECO tool automatically generates netlist edit commands (the patch). This Formality ECO patch is used by downstream tools (Design Compiler, Fusion Compiler, IC Compiler, and IC Compiler II) to modify the original netlist (O-NET) to generate the final patched netlist (P-NET). Formality ECO can also generate the supplemental SVF file to verify the P-NET against the ECO-RTL (See Miscellaneous).

The recommended flow is to take the final patched netlist through the IC Compiler II tool to leverage its incremental place and route capabilities and generate the final ECO netlist. This final ECO netlist should be verified against the E-RTL using the Formality tool and yield a successful verification result.

ECO RTL Flow Using Targeted Synthesis

The full ECO synthesis flow (legacy Formality ECO flow) requires complete E-RTL synthesis to generate the E-RTL ECO region gates. This can be time consuming for designs that have a long synthesis runtime.

In the targeted synthesis flow, only the ECO portion of the E-RTL is synthesized instead of the entire design. This provides excellent runtime improvement over the full ECO

Feedback

synthesis flow. You can choose between complete synthesis or targeted synthesis based on the options you use with the `set_fm_eco_mode` command in the synthesis script.

To enable targeted synthesis, use the `set_fm_eco_mode` command with the appropriate options in the synthesis script. The flow triggers targeted synthesis with the `set_verification_top` command in Design Compiler and the `set_top_module` command in Fusion Compiler. Synthesis performs all the steps in the flow and writes out the synthesized netlist.

Any remaining steps in the synthesis Tcl script, after the `set_verification_top` stage (in Design Compiler) and the `set_top_module` stage (in Fusion Compiler), are ignored after the flow is invoked during targeted synthesis using the `set_fm_eco_mode` command. The `set_fm_eco_mode` command includes options to add or tweak the constraints for compile.

For details on the`set_fm_eco_mode` command, see The set_fm_eco_mode Command in Design Compiler and Fusion Compiler in the Miscellaneous section.

For targeted synthesis, use the O-NET in the form of DDC in Design Compiler and NDM in Fusion Compiler. You can still use a Verilog netlist when running the full E-RTL synthesis flow.

The following example shows the `set_fm_eco_mode` command usage for the Design Compiler tool:

```
set_fm_eco_mode
# Set Formality Auto ECO Mode

 -region  <file_name>       (Name of FRD file)
  [-netlist <file_name]>  (O-NET DDC file)

 [-compile_options string]  (<same compile options as the first
 compile_ultra used for O-NET generation>)
 [-pre_compile_script <file_name>] (TCL script to be sourced prior to
 compile to add any additional settings which are not saved in the O-NET
 ddc)
  [-post_compile_script <file_name>] (TCL script to be sourced after
 compile, like reporting commands etc.)


Library and tool setup
…..
Read the design
Elaborate the design
set_verification_top
```

Note:

Ensure that there are no commands between the `elaborate` and `set_verification_top` commands in the script.

Feedback

The following example shows the `set_fm_eco_mode` command usage for the Fusion Compiler tool:

```
set_fm_eco_mode
# Set Formality Auto ECO Mode
  -region  <file_name>              (Name of FRD file)
  [-netlist <path>/<nlib_file_name>:<block_name>/<label>] (O-NET NDM
 file)
  [-compile_options string]  (<to the same stage as the O-NET was written
 out, like <-to initial_opto>)
[-pre_compile_script <file_name>] (TCL script to be sourced prior to
 compile   to add any additional settings which are not saved in the
 O-NET NDM )
[-post_compile_script <file_name>] (TCL script to be sourced after
 compile, like reporting commands)


Library and tool setup
….

Read the design
set_top_module
```

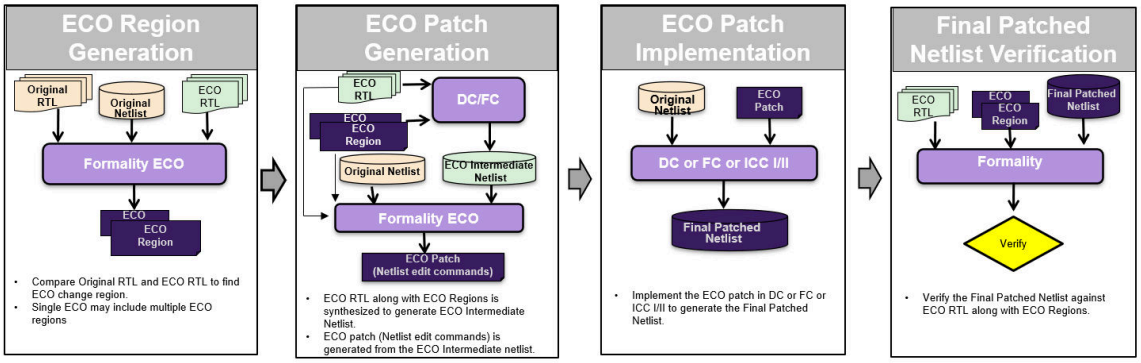
Note:

If you observe any issues using the targeted synthesis flow, switch back to the traditional full ECO synthesis flow.

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. ECO Region Generation

   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. Match the ECO regions: Identify the portion of the original netlist that needs to be rectified.

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

2. ECO Patch Generation

   a. Synthesize the ECO RTL using targeted synthesis: ECO synthesis preserves ECO regions in the intermediate ECO netlist.

   b. Create ECO Patch: Generates the patch or netlist edit commands from Formality ECO.

   c. Confirm ECO Patch (Optional): Applies the generated patch to the original netlist (within the Formality tool) and verifies it against the ECO RTL before implementing the patch.

3. ECO Patch Implementation

   Take the patch or the netlist edit commands generated by Formality-ECO into Design Compiler, Fusion Compiler, IC Compiler, or IC Compiler II tool to implement the gate changes on the original netlist and write out the final patched netlist.

4. Final Patched Netlist Verification

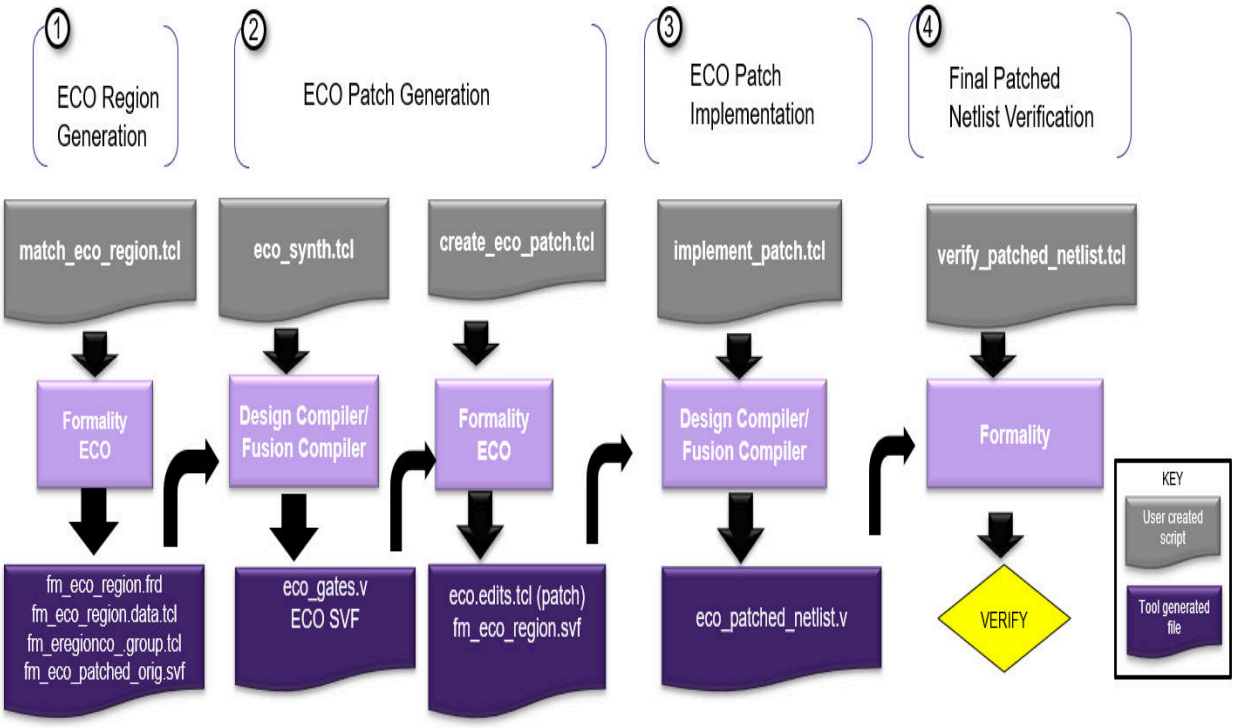
   Confirm the final patched netlist generated by the Design Compiler, Fusion Compiler, IC Compiler, or IC Compiler II tool against the ECO RTL.

Feedback

ECO RTL Flow Scripts

Figure 3 shows the Tcl scripts you have to create and run in the Formality ECO RTL flow to generate the final patched netlist using the Design Compiler, Fusion Compiler, and Formality ECO tools:

Figure 3



1. ECO Region Generation (`match_eco_region.tcl`)
2. ECO Patch Generation
   a. Synthesize the ECO RTL using targeted synthesis (`eco_synth.tcl`)
   b. Create ECO patch (`create_eco_patch.tcl`)
   c. ECO patch verification within Formality (`confirm_patch.tcl`) --Optional
3. ECO Patch Implementation (`implement_patch.tcl`)
4. Final Patched Netlist Verification (`verify_patched_netlist.tcl`)

Feedback

ECO Region Generation

The ECO Region Generation step creates ECO regions. 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.

See Also

- ECO Patch Generation
- ECO Patch Implementation
- Final Patched Netlist Confirmation

User Setup for match_eco_region

All the appropriate setup required for successful verification of the original netlist against the original RTL should get applied to all the applicable containers.

For example, any constraints like `set_constant`, `set_user_match`, `set_dont_verify_points`, `set_black_box`, and so on should get applied to all the applicable containers such as `ortl`, `ertl`, `r`, and `i`.

The Formality ECO tool attempts to automatically transfer the setup from `r` to `ortl` and `ertl` and checkpoint verifications if any.

The following types of setups are automatically transferred:

- `set_constant`
- `set_dont_verify_points`
- `set_dont_match_points`
- `set_black_box`
- `set_cutpoint`

Note:

These are the most common setups that cause issues in Formality ECO flow. Any unsupported setup still has to be applied to all the containers manually.

See Also

- SVF Checkpoint Support
- UPF Support

Feedback

- Match ECO Region Script
- Debugging Tips

SVF Checkpoint Support

When using Fusion Compiler, the SVF file may contain checkpoints for retiming. These checkpoints are intermediate netlists that the Formality tool uses for verification. The Formality ECO flow is designed to automatically handle checkpoints to preserve the continuity of ECO verification. Checkpoints in Design Compiler flow are also treated in the same manner.

Design-based checkpoint (DBC) technology improves the runtime of preverification by verifying only the DesignWare retimed block from the checkpoint. This greatly simplifies the Formality ECO flow as there is no requirement to patch the checkpoint when DBC is successful.

Note:

For designs where DBC does not work (not applicable or fails) then ECOs can still be done. Formality ECO automatically patches the checkpoint netlist during the `match_eco_region` step and generates a patched original SVF that should be used instead of the original SVF.

There are two scenarios that occur in the DBC Formality ECO flow:

1. Scenario1: When DBC is deployed and passes:

   If the checkpoint is due to the presence of DesignWare components and it succeeds using DBC verification, the fm_eco_patched_orig.svf patched original SVF file is not produced. This is because Formality ECO does not need to patch the checkpoint netlist when checkpoint verification succeeds using this method. There is no special Formality ECO setup required when DBC passes.

2. Scenario2: When DBC Verification cannot be used or DBC is deployed but fails:

   For the designs where the checkpoint is due to the presence of user hierarchy retiming, DBC verification cannot be used. So, the checkpoint needs to be patched and Formality ECO produces the fm_eco_patched_orig.svf file. In the final confirmation step, the fm_eco_patched_orig.svf file is used instead of the original SVF file.

See Also

- User Setup for match_eco_region
- UPF Support

Feedback

- Match ECO Region Script
- Debugging Tips

UPF Support

The Formality ECO flow supports functional ECOs on low-power (UPF) designs. The UPF is constrained during the ECO steps.

Formality ECO does not support UPF aware patches. That is, a patch cannot be generated for any power-intent changes or for ECOs that cross power-domain boundaries.

O-RTL to O-NET verification must pass without the `load_upf` command and by using the `constrain_low_power_intent` on the O-NET.

UPF needs to be constrained in all the ECO steps. If a design uses the UPF low-power flow, do not include any `load_upf` commands in the Formality ECO Tcl script and use the `constrain_low_power_intent` command. This command constrains isolation cells and retention cells in the design. Usually, only the netlists have the low power cells, but if there are instantiated low power cells in RTL, you need to constrain the RTL as well.

Note:

Additional user setup might be needed to constrain away the low power intent if the netlist has power switches as shown in the following example.

The following example shows the usage of the `constrain_low_power_intent`:

```
constrain_low_power_intent $ref
constrain_low_power_intent $impl

constrain_low_power_intent ertl:/WORK/<DESIGN_NAME> ; If RTL has
 instantiated low power cells
constrain_low_power_intent ortl:/WORK/<DESIGN_NAME> ; If RTL has
 instantiated low power cells
```

See Also

- User Setup for match_eco_region
- SVF Checkpoint Support
- Match ECO Region Script
- Debugging Tips

Feedback

Match ECO Region 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 to load it later in this script.
## See Miscellaneous section for more details.

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 as DDC for Design Compiler and
## NDM for Fusion Compiler in Targeted Synthesis flow

read_ddc -i ./DC/orig.gates.ddc
set_top top

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

## For designs with UPF flow,donot read the UPF
## 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
```

Feedback

```
## 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
## Check the section "User setup for match_eco_region" for more details
# 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


## 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
## More details on these commands can be
## found in the Miscellaneous section
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
## which will be used later in the flow
##    a.) fm_eco_region.frd
##    b.) fm_eco_region.group.tcl
##    c.) fm_eco_region.data.tcl
##    d.) fm_eco_patched_orig.svf; Please check the section
## "SVF Checkpoint Support" for more details

write_eco_regions -replace

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

quit
```

Note:

- For the targeted synthesis flow, load the O-NET as DDC for Design Compiler and NDM for Fusion Compiler. Make sure to use the same O-NET DDC or NDM throughout the flow for reader consistency, that is, during

Feedback

the `match_eco_region`, targeted ECO synthesis, `create_eco_patch`, and patch implementation steps.

- For the full synthesis flow, load O-NET as Verilog for both Design Compiler and Fusion Compiler.

  For the Full Synthesis flow scripts, see Full Synthesis Flow Scripts in the Miscellaneous section.

See Also

- User Setup for match_eco_region
- SVF Checkpoint Support
- UPF Support
- Debugging Tips

Debugging Tips

Ensure that the following conditions are met after running the `match_eco_region` 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.

You can confirm whether ECO changes reported by the tool are expected based on the information from the `match_eco_region` log file.

```
Find ECO regions: Structural analysis found 1 ECO origins in 2 hier.
 blocks
Find ECO regions: ECO'd hier. blocks:
  ertl:/WORK/top/U1
  ertl:/WORK/top/U2
```

`ECO'd hier. blocks` refers to the hierarchical designs with ECO changes.

- Make sure the fm_eco_region.frd, fm_eco_region.group.tcl, and fm_eco_region.data.tcl files are generated.
- If there are problems with generating the ECO region files, replace the `preverify` command with low-effort verification and ensure that there are no failures. Resolve failures if there are any before proceeding ahead.

Feedback

See Also

- User Setup for match_eco_region
- SVF Checkpoint Support
- UPF Support
- Match ECO Region Script

ECO Patch Generation

This step consists of ECO synthesis and creating an ECO patch. Optionally, you can also verify the correctness of the patch separately within the Formality tool.

See Also

- ECO Region Generation
- ECO Patch Implementation
- Final Patched Netlist Confirmation

ECO Synthesis

The ECO synthesis Tcl script sets up the Design Compiler or Fusion Compiler Tcl session to synthesize the E-RTL as indicated using the ECO region information created in the previous step.

Use the same synthesis script that was used during the original synthesis.

If the ECO is purely combinational and impacts bits of a multibit bank, then a good tip is to disable multibit optimization during ECO synthesis. Doing this can result in a better quality patch where the multibit register impacted by the ECO is patched in place in the O-NET.

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 <FRD_file>` command. You must invoke this command before the `analyze`, `elaborate`, or `read_file` command in your script.

Choosing to perform full synthesis or targeted synthesis is based on the options you use with the `set_fm_eco_mode` command.

Example 2 shows the Fusion Compiler script. See Table 1 for the terminologies used in the script.

Feedback

Example 2 Fusion Compiler Script

```
## Use the ECO mode
## Recommendation is to use O-NET in the form of NDM
## Check the Miscellaneous section for set_fm_eco_mode options
set_fm_eco_mode \
-region            ./ECO_WORK/fm_eco.frd \
-netlist           ./ONET.nlib:design_top \
-compile_options {-to initial_opto}

set_svf eco_ts_flow.svf

set search_path "./libs ./eco_rtl . ./libs"

set ref_path "./libs"
set tech_lib_file "${ref_path}/xxx_tech.ndm"
set ref_libs "${ref_path}/xxx_hvt.ndm ${ref_path}/xxx_tech.ndm"

create_lib -use_technology_lib $tech_lib_file -ref_libs $ref_libs
 ENET.nlib

analyze -format verilog "rtl1.v rtl2.v rtl3.v design_top.v"
elaborate design_top
set_top_module design_top

## After set_top_module, control goes back to set_fm_eco_mode,
## Additional options with set_fm_eco_mode address the remaining script
## See the ECO RTL Flow Using Targeted Synthesis and
## Miscellaneous sections for details
## The gates.eco.v netlist is generated at the end of the targeted
 synthesis flow
```

Example 3 shows Design Compiler script. SeeTable 1 for the terminologies used in the script.

Example 3 Design Compiler Script

```
## Use the ECO mode
## Recommendation is to use ONET in the form of DDC
## Check Miscellaneous section for the options of set_fm_eco_mode
set_fm_eco_mode \
-region ./ECO_WORK/fm_eco.frd \
-netlist ./design_top.ddc \
-compile_options {-scan -gate_clock -no_autoungroup }

set search_path ". ./eco_rtl ./lib"
set target_library “some_tech_lib.db”
set synthetic_library “dw_foundation.sldb”
set link_library " * $target_library $synthetic_library"

## This is the E-SVF
set_svf  ./eco_ts_flow.svf
```

Feedback

```
# ECO RTL
analyze -format verilog "rtl1.v rtl2.v rtl3.v design_top.v"
elaborate design_top
current_design design_top
set_verification_top

## After set_verification_top, control goes back to set_fm_eco_mode,
##Additional options with set_fm_eco_mode will address the remaining
 script
## See the ECO RTL Flow Using Targeted Synthesis and
## Miscellaneous sections for details
```

Note:

Use the same synthesis script that was used during the original synthesis.

Targeted synthesis gets triggered with the `set_top_module` command in Fusion Compiler and the `set_verification_top` command in Design Compiler. The control then goes back to the `set_fm_eco_mode` command.

All steps in the script after these are ignored. For additional plug-ins with the `set_fm_eco_mode` command, or to add any constraints or commands, see the ECO RTL Flow Using Targeted Synthesis and Miscellaneous sections.

The control is returned to the main script after `set_top_module` or `set_verification_top` command; so all user script commands after these commands get executed with full synthesis.

For full synthesis scripts for Design Compiler and Fusion Compiler, see Full Synthesis Flow Scripts in the Miscellaneous section.

For debugging, ensure the following after running the ECO synthesis script:

- Synthesis tool does not report any error after using the `fm_eco_region.frd` script.
- Check the synthesis log to confirm that all the ECO regions are successfully accepted during ECO synthesis. ECO synthesis should stop if any region creation issues exist.

```
Formality Auto Eco Regions Summary
----------------------------------
Number of Regions (Success) : 7
Number of Regions (Fail)    : 0
Total Number of Regions     : 7
```

- E-NET and E-SVF are generated properly.

See Also

- Create ECO Patch
- User Setup for the create_eco_patch Command

- Scan Chain Preservation
- Clock Tree Preservation
- Multibit Register Support
- Confirm ECO Patch

Create ECO Patch

Creating the ECO patch script sets up the Formality ECO tool to match the E-RTL and E-NET and generate the Formality ECO patch (netlist edit commands). This script requires multiple SVF files:

- Supplemental SVF
- Original SVF
- ECO synthesis SVF

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. See Generating Supplemental SVF for ECO Changes in the Miscellaneous section. It generates the ECO region SVF which needs to be used during the final patched netlist verification step.

See Also

- ECO Synthesis
- User Setup for the create_eco_patch Command
- Scan Chain Preservation
- Clock Tree Preservation
- Multibit Register Support
- Confirm ECO Patch

User Setup for the create_eco_patch Command

All the appropriate setups required for successful verification of the ECO netlist against the ECO RTL should get applied to all the applicable containers involved in the `create_eco_patch` step.

For example, any constraints such as `set_constant`, `set_user_match`, `set_dont_verify_points`, `set_black_box`, and so on should get applied to all the applicable containers such as `r`, `i`, and `onet`.

Feedback

The Formality ECO tool attempts to automatically transfer the setup from `i` to `onet`. The following types of setup are automatically transferred:

- `set_constant`
- `set_dont_verify_points`
- `set_dont_match_points`
- `set_black_box`
- `set_cutpoint`

These are the most common setups that cause issues in the Formality ECO flow. Any unsupported setups still have to be applied to all the containers manually.

Note:

This applies to full synthesis flow only. Any constraints on ONET in the `match_eco_region` step are not automatically transferred to ONET in the `create_eco_patch` step unless they get automatically transferred from the `i` to the `onet` container in the `create_eco_patch` step. You need to manually provide it.

See Also

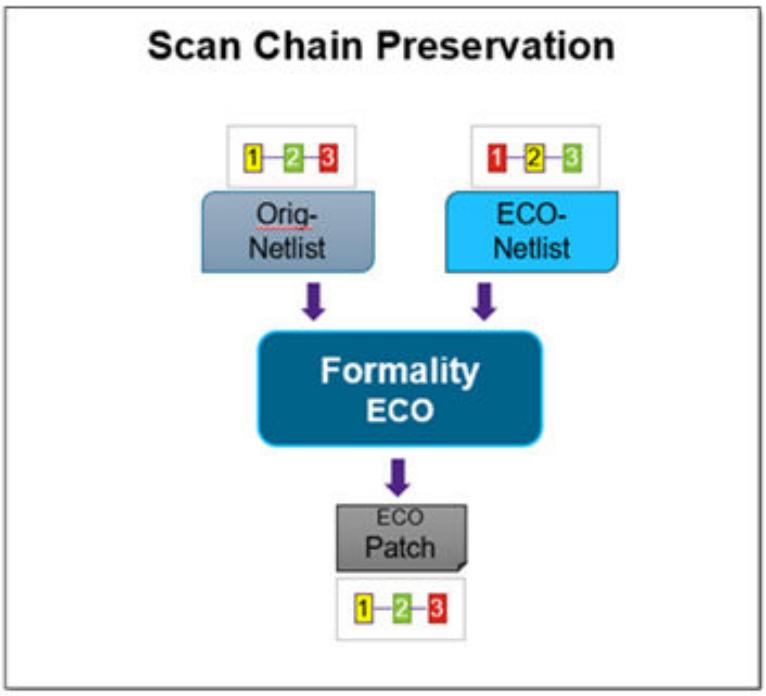
- ECO Synthesis
- Create ECO Patch
- Scan Chain Preservation
- Clock Tree Preservation
- Multibit Register Support
- Confirm ECO Patch

Scan Chain Preservation

The Formality ECO tool preserves the existing scan chain order. 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 (using the `report_eco_impact` command) so that the DFT scan chain can be manually edited.

Feedback

Figure 4 Scan Chain Preservation



See Also

- ECO Synthesis
- Create ECO Patch
- User Setup for the create_eco_patch Command
- Clock Tree Preservation
- Multibit Register Support
- Confirm ECO Patch

Clock Tree Preservation

The Formality ECO tool attempt to reuse the clock of the O-NET as much as possible to minimize clock-tree disruption as long as ECO changes do not impact the load enable condition of the registers.

See Also

- ECO Synthesis
- Create ECO Patch

Feedback

Multibit Register Support

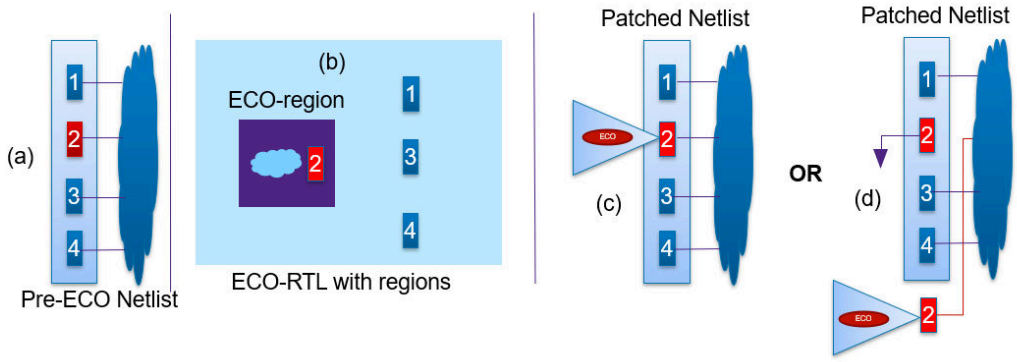
The Formality ECO tool maintains the multibit register mapping of the original design. Also, it can perform multibit-patch-in-place. If there are multibit registers in the ONET and the ECO has impacted at least some bits of the multibit registers, then the Formality ECO tool attempts to patch-in-place the multibit registers of the ONET.

Note:

This is not always possible, for example, when the ECO impacts the load enable of only a few of the bits of the multibit register.

Multibit patch in place is only possible when multibit banking is disabled during ECO synthesis.

Figure 5 Multibit Register Support



In Figure 5, the ECO impacts only 1 bit of the bank (figure a). The ECO RTL has ECO regions that contain only that register (figure b). Registers in the ECO region are not banked with registers outside the region.

The Formality ECO tool can patch in place (figure c) only if you confirm that the register in question has identical shared-logic (such as `enable`, `clock`, `reset` and so on) with other flip-flops in the bank.

For cases, where you cannot patch in place, the tool creates a new register while keeping the original bank intact (figure d).

Feedback

Additional setup files are written out during the `create_eco_patch` step for multibit designs during the Create ECO Patch 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

Example 4 shows Create ECO Patch script. See Table 1 for the terminologies used in the script.

Example 4

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

## Expand the ECO region information into a directory structure on disk
set_eco_data fm_eco.frd

## Make sure to set_svf to eco_change.svf, original SVF
## and the SVF created during Targeted Synthesis

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

## Append Targeted Synthesis SVF
set_svf -append eco_ts_flow.svf

read_db "some_tech_lib.db"
## Read E-RTL
## Do not use the previously generated ERTL container
## See “Miscellaneous” section for more details
read_verilog -r ./eco_rtl/top.v
set_top top

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

## For designs with UPF flow, donot read the UPF
## Uncomment the following line for UPF low-power flow
# constrain_low_power_intent $impl

## Reload O-NET in separate container
## For Targeted Synthesis Flow, load O-NET as ddc
## for Design Compiler and NDM for Fusion Compiler
read_container -container onet ./ECO_WORK/onet.fsc

## Perform any setup needed to verify E-RTL and E-NET
```

Feedback

```
## Refer to the section above "User Setup for create_eco_patch"

# 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


current_container r
source $fm_work_path/fm_eco_frd/ECO_1/fm_ECO_region.group.tcl

## For debugging purposes replace "match" with the lines below.
## The resulting verification of E-RTL and E-NET should have no failures.
## This will also perform the same function as "match".
##
# set verification_effort_level super_low
# verify
## Set up containers, source the data file to read ECO regions
## Create patch file, and create ECO region SVF file

if { [ match ] } {
set_eco_ref r
set_eco_imp i
set_orig_imp onet
source $fm_work_path/fm_eco_frd/ECO_1/fm_ECO_region.data.tcl

#### Generates fm_eco_region.patch.tcl, fm_eco_region.svf
## For multibit designs, additional files fm_eco_region.confirm_setup.tcl
## and fm_eco_region.netlist_flow_setup.tcl will be generated

create_eco_patch -replace

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

## Reload a fresh copy of the original netlist container, and apply the
## patch to get the final ECO Tcl script (netlist edit commands)
remove_container i

## Reload (or create again) the original netlist container
read_container -i ./ECO_WORK/onet_imp.fsc -replace

## Source the FM patch just created
source ./fm_eco_region.patch.tcl

## Write out final ECO Tcl script (netlist edit commands) for use in ICC
## (ICC2/DC/FC)
write_edits -replace eco.edits.tcl
quit
```

Feedback

Note:

1. Usage of SVF files in `create_eco_patch` is slightly different between the targeted synthesis flow and the full synthesis flow.
   - For the targeted synthesis flow, you need to use the eco_change.svf, original SVF, and ECO synthesis SVF.
   - For full synthesis flow, you just need ECO synthesis SVF.
2. For the targeted synthesis flow, load O-NET as DDC for Design Compiler and NDM for Fusion Compiler.

   For full synthesis flow, load O-NET as Verilog for both Design Compiler and Fusion Compiler.

   For UPF designs, you might need to add additional options while loading the O-NET NDM in Formality. See the miscellaneous section under “Loading O-NET NDM in Formality” for details.
3. For the full synthesis script, see Full Synthesis Flow Scripts in the Miscellaneous section.

For debugging, ensure that the following conditions are met after running the `create_eco_patch` script:

- No errors are reported in the Create ECO Patch script log.
- Make sure the eco.edits.tcl, fm_eco_region.patch.tcl, and fm_eco_region.svf are generated properly.

  The eco.edits.tcl is the final patch (netlist edit commands) to be given to the IC Compiler, IC Compiler II, Design Compiler, and Fusion Compiler tool to implement the patch.

  The fm_eco_region.patch.tclinternal patch file can be used to modify the original netlist within the Formality tool 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

  The fm_eco_region.svf file contains SVF guidance information for the affected ECO regions only.
- For multibit designs, additional setup files are generated to be used in the next steps in the flow. The fm_eco_region.confirm_setup.tcl file should be included during the next step of confirming the patch. The fm_eco_region.netlist_flow_setup.tcl file should be used in the ECO netlist flow
- If there are problems with generating these files, replace the `match` command with the low effort `verify` command and investigate any failing compare points, especially

for setup issues. Ensure that the verification does not find any failing points before continuing with the ECO RTL flow.

- Use `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
  - Clock gates impacted by patch

Note:

Always use the latest versions of the Formality tool as there are continuous improvements with patch size.

See Also

- ECO Synthesis
- Create ECO Patch
- User Setup for the create_eco_patch Command
- Scan Chain Preservation
- Clock Tree Preservation
- Confirm ECO Patch

Confirm ECO Patch

This step is optional.

The confirm ECO patch script confirms whether the Formality internal 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: Supplemental SVF (`eco_change.svf`), Original SVF, ECO region SVF. 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.

See Generating Supplemental SVF for ECO Changes for details.

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.

Feedback

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 5.

If the *Create ECO Patch* 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 5 shows the script to confirm the ECO patch. See Table 1 for the terminologies used in the script.

Example 5 Confirm ECO Patch Script

```
## Confirms the correctness of the ECO patch before implementing
## the patch. Using this script is optional.
set synopsys_auto_setup true

## Expand the ECO region information into a directory structure on disk
set_eco_data fm_eco.frd

## 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)

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

## Read O-NET or read O-NET container created previously
## For Targeted Synthesis Flow, load O-NET as ddc
## for Design Compiler and NDM for Fusion Compiler

# 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
```

Feedback

```
current_container r
source $fm_work_path/fm_eco_frd/ECO_1/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


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

match
verify

quit
```

Note:

For the targeted synthesis flow, load O-NET as DDC for Design Compiler and NDM for Fusion Compiler.

For the full synthesis flow, ONET can be used as Verilog.

For UPF design, you may need to add additional switches while loading O-NET NDM in Formality. See Loading O-NET NDM in Formality ECO in the Miscellaneous section for details. For the full synthesis script, see Full Synthesis Flow Scripts in the Miscellaneous section.

For debugging, if patch confirmation fails, then go back to the previous step `create_eco_patch`, and verify ERTL to ECO Netlist. Debug the failures if ERTL to ENET verification fails.

Feedback

See Also

- ECO Synthesis
- Create ECO Patch
- User Setup for the create_eco_patch Command
- Scan Chain Preservation
- Clock Tree Preservation
- Multibit Register Support

ECO Patch Implementation

Take the eco.edits.tcl Tcl patch generated from the Create ECO Patch step and apply it to the O-NET in the Design Compiler, Fusion Compiler, or IC Compiler II tool to generate the final patched netlist.

Example 6 shows the script to implement the patch in Design Compiler. See Table 1 for the terminologies used in the script.

Example 6 Design Compiler Script

```
set search_path ". ./netlist ./lib"
set target_library "some_tech_lib.db"
set synthetic_library "dw_foundation.sldb"
set link_library " * $target_library $synthetic_library"

## Read the ONET ddc and link the design
read_ddc ./onet.ddc
link
## Source the ECO patch generated from Formality-ECO flow
source eco.edits.tcl
## Write out the final patched ONET
write_file -format ddc -hierarchy -output gates-patched.ddc
write_file -format verilog -hierarchy -output gates-patched.v
quit
```

Example 7 shows the script to implement the patch in Fusion Compiler. See Table 1 for the terminologies used in the script.

Example 7 Fusion Compiler Script

```
….
open_block $DESIGN_NAME/initial_opto

current_lib
current_block
```

Feedback

```
## Source the ECO patch generated from Formality-ECO flow
source ../eco.edits.tcl
## Write out the final patched ONET
save_block -label gates-patched
write_file -format verilog -hier -output gates-patched.v


quit
```

See Also

- ECO Region Generation
- ECO Patch Generation
- Final Patched Netlist Confirmation

Final Patched Netlist Confirmation

This step verifies the final patched netlist from the IC Compiler, IC Compiler II, Design Compiler, or Fusion Compiler tool against the ECO RTL.

Use the `set_svf` command to read the eco_change.svf file and the original SVF, and the `set_svf -append` command to read the ECO region SVF.

Follow these considerations during Fusion Compiler or Design Compiler synthesis:

- If the tool created the fm_eco_patched_orig.svf file due to checkpoints, use it instead of the original SVF file.
- Always include the `source fm_eco_region.group.tcl` command on the reference container (ECO RTL) before running the `match` or `verify` command. Also include the fm_eco_region.confirm_setup.tcl file if it exists.

The following example script verifies the final patched netlist from the IC Compiler, IC Compiler II, Fusion Compiler, or Design Compiler tool against the ECO RTL

Example 8 shows the script to verify the final patched netlist (P-NET). See Table 1 for the terminologies used in the script.

Example 8 Confirm Final Patched Netlist (P-NET) against ECO RTL

```
## FM ECO Tcl Script to Verify Final Patched Netlist from ICC2
set synopsys_auto_setup true

## Expand the ECO region information into a directory structure on disk
set_eco_data fm_eco.frd

## Use fm_eco_patched_orig.svf if created during match_eco_region script
## due to one or more guide_checkpoint guidance in original SVF.
```

Feedback

```
## Otherwise, use original SVF. Include supplementary SVF (if needed).
set_svf eco_change.svf
if {[file exists fm_eco_patched_orig.svf]} {
set_svf -append fm_eco_patched_orig.svf
} else {
set_svf -append ./DC/orig.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

## Read patched O-NET from ICC (ICC2/DC/FC)
read_verilog -i gates-patched.v
set_top top

## Apply ECO region group to E-RTL
current_container r
source $fm_work_path/fm_eco_frd/ECO_1/fm_ECO_region.group.tcl

## For low-power flow, load original reference and 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
quit
```

Feedback

See Also

- ECO Region Generation
- ECO Patch Generation
- ECO Patch Implementation

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

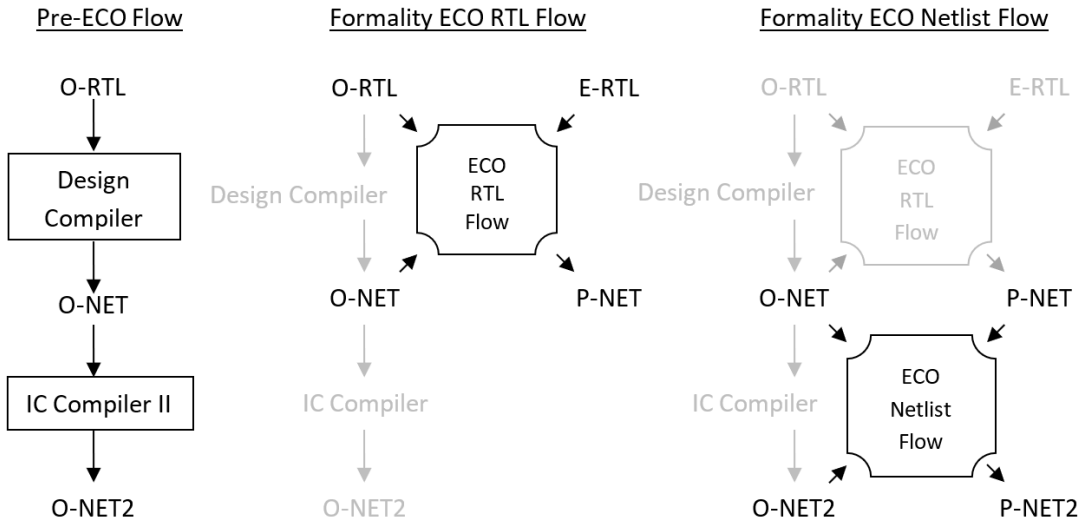
Note:

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 functionally equivalent to the post ECO netlist. This process is accomplished without the intermediate use of synthesis.

Figure 6 illustrates how the Formality ECO netlist flow fits into the Formality ECO flow:

Figure 6 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

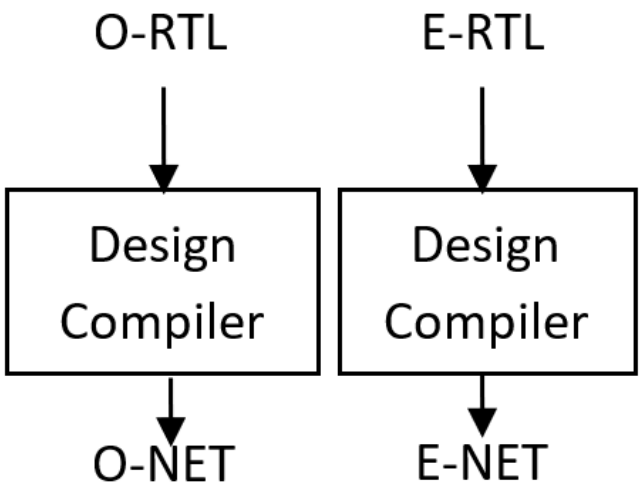
Netlists resulting from flows unrelated to the Formality ECO RTL flow are not recommended for the Formality ECO netlist flow.

Figure 7 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 7 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

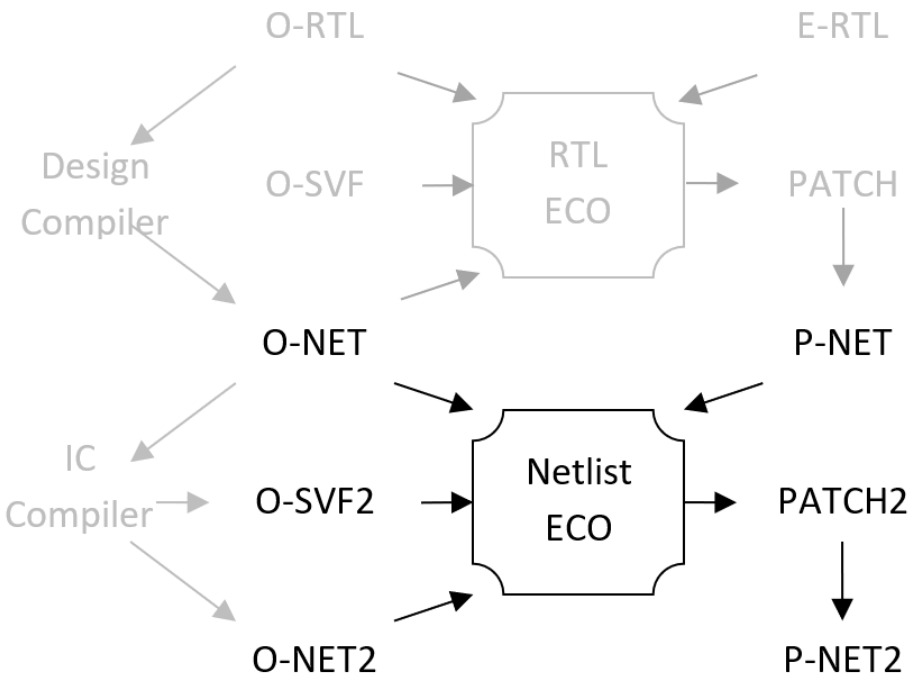
The Formality ECO netlist flow is similar to the RTL flow. However, there is no synthesis involved. So, there is no ECO region group script. The region generation and the patch generation steps can be combined into one script. Next, implement the patch (netlist edit commands) and generate the final patched optimized netlist (P-NET2). Finally, verify the final patched optimized netlist (P-NET2) to be functionally equivalent to P-NET.

Optionally, you can verify the correctness of the patch within Formality by applying it to Optimized Original Netlist (O-NET2) and verifying against P-NET.

Figure 8 illustrates the typical Formality ECO netlist flow. See Table 2 for terminologies used in the ECO netlist flow.

Feedback

Figure 8 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 fm_eco_region.netlist_flow_setup.tcl setup file during the Create ECO Patch 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.

Note:

Additional user setup might be needed to constrain away the low power intent if the netlist has power switches.

See Also

- Confirm ECO Patch Script for the Netlist Flow
- Create ECO Patch Script for the Netlist Flow
- ECO Patch Implementation for the Netlist Flow
- Final Patched Netlist Verification 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 single script identifies ECO regions and generates the internal Formality ECO patch.

Example 9 shows the script to Create the ECO Patch Script for the Netlist flow. See Table 2 for the terminologies used in the script.

Example 9 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 reference design, O-NET created
## by Design Compiler  or Fusion Compiler
read_verilog -netlist -r O-NET.v
set_top top_of_design

## Donot load the UPF files
## Uncomment the following line if using UPF flow
# constrain_low_power_intent $ref
```

Feedback

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

## Read in original implementation design ,O-NET2
## created by IC Compiler or  IC Compiler II
read_verilog -netlist –i O-NET2.v
set_top top_of_design

## Write out original implementation container for later use
write_container -i ./ECO_WORK/O-NET2.fsc

## Donot load the UPF files
## 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

## Donot load the UPF files
## 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
## When there is no original SVF, no need to read the original reference
## again in 'ONET' container and set the command set_orig_reference to
 'r'
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 fm_eco_region.netlist_flow_setup.tcl]} {
```

Feedback

```
source ./fm_eco_region.netlist_flow_setup.tcl
}

## Find the regions to be patched
match_eco_regions

## Generate internal Formality ECO patch file named
## fm_eco_region.patch.tcl

create_eco_patch -replace

## Report patch size information
report_eco_impact -all > report_eco_impact.txt q

##  Reload a fresh copy of the original implementation O-NET2 and apply
 the
## internal patch fm_eco_region.patch.tcl to generate the netlist edit
## commands eco.edits.tcl
remove_container i
read_container -i ./ECO_WORK/O-NET2.fsc

## Source the internal FM patch just created
source ./fm_eco_region.patch.tcl

## Write out final ECO Tcl script for use in ICC or ICCII
write_edits -replace eco.edits.tcl
```

For debugging, check the following after executing the ‘Create ECO Patch’ script:

- Ensure that verification of O-NET and O-NET2 is successful before proceeding
- Make sure the eco.edits.tcl and fm_eco_region.patch.tcl files are generated properly.
  - The eco.edits.tcl is the final patch or netlist edit commands to be given to ICC, ICCII to generate the final patched optimized netlist (P-NET2).
  - The fm_eco_region.patch.tcl internal patch file generated can be used to modify the original reference design (O-NET2) within the Formality tool to compare it against P-NET. This is not the final netlist patch file for the IC Compiler or IC Compiler II tool.
- The fm_eco_region.confirm_setup.tcl file might be created for handling multibit designs downstream.
- The netlist flow does not generate a formality_eco_region.group.tcl file. This is created only in the Formality ECO RTL flow.

Feedback

- The script in Example 9 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
- ECO Patch Implementation for the Netlist Flow
- Final Patched Netlist Verification for the Netlist Flow

Confirm ECO Patch Script for the Netlist Flow

The Formality ECO Tcl script in Example 10 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 10 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
```

Feedback

```
## Read in Formality ECO patch file to modify O-NET2 to be functionally
## equivalent to P-NET
current_container i
source fm_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 fm_eco_region.confirm_setup.tcl]} {
  preverify
  source ./fm_eco_region.confirm_setup.tcl
}

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

quit
```

Check whether the verification is successful after you run the confirm script.

It is recommended that you verify the final patched netlist P-NET2 generated from IC Compiler or IC Compiler II against P-NET.

See Also

- ECO Netlist Flow Summary
- Create ECO Patch Script for the Netlist Flow
- ECO Patch Implementation for the Netlist Flow
- Final Patched Netlist Verification for the Netlist Flow

ECO Patch Implementation for the Netlist Flow

Apply the ECO patch file or the netlist edit commands (eco.edits.tcl) to the original implementation design O-NET2 and generate the final patched optimized netlist (P-NET2).

Feedback

Example 11 Patch Implementation for Fusion Compiler or IC Compiler II

```
……..
open_block $DESIGN_NAME/initial_opto

current_lib
current_block

## Source the ECO patch generated from Formality-ECO flow
source ../eco.edits.tcl
## Write out the final patched optimized netlist P-NET2
save_block -label initial_opto_P-NET2
write_file -format verilog -hierarchy -output P-NET2.v

quit
```

See Also

- ECO Netlist Flow Summary
- Create ECO Patch Script for the Netlist Flow
- Confirm ECO Patch Script for the Netlist Flow
- Final Patched Netlist Verification for the Netlist Flow

Final Patched Netlist Verification for the Netlist Flow

This step verifies the final patched optimized netlist (P-NET2) from the IC Compiler, IC Compiler II, Design Compiler, or Fusion Compiler tool against the P-NET.

Example 12 Final Patched Netlist Verification 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 (P-NET)
read_verilog -netlist -r P-NET.v
set_top top_of_design

## Read final patched optimized netlist P-NET2 from
##ICC (ICC2/DC)
read_verilog -i P-NET2.v
set_top top_of_design

## 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 fm_eco_region.confirm_setup.tcl]} {
 preverify
source ./fm_eco_region.confirm_setup.tcl
}

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

quit
```

See Also

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

A

Miscellaneous

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

See Also

- Reading Designs in the Formality ECO Flow
- Container Reference used in the Formality ECO flow
- Loading O-NET NDM in Formality ECO
- E-RTL Container Generation Recommendation in the Formality ECO flow
- Options to be Used With the match_eco_region Command
- ECO Synthesis Using Full Synthesis Flow
- The set_fm_eco_mode Command in Design Compiler and Fusion Compiler
- Full Synthesis Flow Scripts
- Generating Supplemental SVF for ECO Changes

Reading Designs in the Formality ECO Flow

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
```

Container Reference used in the Formality ECO flow

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`

For example, to specify the top-level design within the `ortl` container 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.

Loading O-NET NDM in Formality ECO

The O-NET NDM generated by the Fusion Compiler tool contains saved UPF objects.

For the Formality ECO flow, use these options while loading the O-NET NDM:

```
read_ndm
  -no_upf                    #Prevents UPF sourcing
  -format non_pg_netlist     #If NDM has power supplies in it
  -preserve_supply_constant #To create drivers for supply constants
```

E-RTL Container Generation Recommendation in the Formality ECO flow

The `guide_hier_map` guidance gets applied to the reference container during elaboration in Formality (`set_top` stage).

The `guide_hier_map` command is different during ECO synthesis and original synthesis. So, avoid reusing the previously generated E-RTL containers and regenerate the E-RTL container at each stage in the Formality ECO flow using that SVF information of that stage.

Options to be Used With the match_eco_region Command

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 design. 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`

ECO Synthesis Using Full Synthesis Flow

The following example shows the usage of the `set_fm_eco_mode` command in the Design Compiler tool:

```
set_fm_eco_mode
# Set Formality Auto ECO Mode

 -region  <file_name>       (Name of FRD file)

Library and tool setup
…..
Read the design
Elaborate
set_verification_top
## Remainder of the customer synthesis script
….
```

The following example shows the usage of the `set_fm_eco_mode` command in the Fusion Compiler tool:

```
set_fm_eco_mode
# Set Formality Auto ECO Mode
  -region  <file_name>        (Name of FRD file)

-pre_region_script <tcl script>  (If needed, see appendix for details)

Library and tool setup
```

```
….

Read the design
set_top_module
## Remainder of customer synthesis script
…..
```

Note:

Unlike the targeted synthesis flow, control is returned to the synthesis script after the `set_top_module` stage.

The set_fm_eco_mode Command in Design Compiler and Fusion Compiler

set_fm_eco_mode

This command enables Formality automatic ECO mode in Fusion Compiler or Design Compiler.

Syntax

```
set_fm_eco_mode
   -region frd_file_name
   [-netlist onet_file_name]
   [-compile_options compile_options]
   [-output enet_name]
   [-pre_region_script file_name]
   [-pre_compile_script file_name]
   [-post_compile_script file_name]
```

Data Types

```
frd_file_name       string
onet_file_name      string
compile_options     string
output              string
enet_name           string
file_name           string
```

Arguments

`-region` **`frd_file_name`**

Specifies the path and name of the FRD (Formality Region Data) file which has ECO region data from Formality. This data is processed immediately after the `set_verification_top` command in Design Compiler or the `set_top_module` command in Fusion Compiler. For the full synthesis flow, control is returned to the customer script after the ECO region data is

processed. For targeted synthesis, the remainder of the customer script after the `set_verificaiton_top` or `set_top_module` command is ignored after the ECO region data is processed, as targeted synthesis takes over using information supplied with the other `set_fm_eco_mode` command options. The `-region` option is required for both the full synthesis and targeted synthesis auto ECO flows.

`-netlist` **`onet_file_name`**

Specifies the path and name of the O-NET (original netlist) which was synthesized from the O-RTL (original RTL). This option is required for the targeted synthesis flow.

For Design Compiler, the format is path_to/onet_file.ddc. The path is not needed if the onet_file.ddc can be found in the search path for the Design Compiler session.

For Fusion Compiler, the format is path_to/onet_nlib_name:onet_block_name/label. The path is not needed if the onet_nlib_name can be found in the search_path of the Fusion Compiler session. The onet_block_name is not needed if it matches the onet_nlib_name. The label is not needed if the onet_block_name is not saved with a label.

`-compile_options` **`compile_options`**

Specifies the compile options to be passed to ECO targeted synthesis.

For Design Compiler, use the same options that were used for the first `compile_ultra` command in the O-NET script, for example, `-compile_options {-scan -no_seq_output_inversion}`. If no compile options are specified, targeted synthesis uses the `-scan` option by default.

For Fusion Compiler, use the same stage as the O-NET, for example, `-compile_options {-to initial_opto}`. If no compile options are specified, targeted synthesis uses the `-to final_opto` option by default.

`-output` **`enet_name`**

Specifies what name to write out for the E-NET (ECO netlist) block. This is used in targeted synthesis flow only.

For Design Compiler, the format is `path_to` or `enet_name` with no extension. For example, the `-output my_eco` option writes out my_eco.ddc and my_eco.v for the resulting targeted synthesis E-NET. If this option is not specified, Design Compiler defaults to writing out gates.eco.ddc and gates.eco.v in the run directory for the resulting E-NET.

For Fusion Compiler, the format is path_to/enet_nlib_name:enet_block_name/label.

Note:

If specified, the `nlib` must already exist. Typically, this is either the O-NET `nlib`, or the local `nlib` created by the targeted synthesis run.

If the `-output` option is not specified, Fusion Compiler targeted synthesis defaults to writing out the resulting E-NET as `gates_eco` in the O-NET `nlib`, and gates.eco.v in the run directory.

If only the `enet_block_name` is specified as `-output my_eco`, Fusion Compiler targeted synthesis writes out the resulting E-NET as `my_eco` in the O-NET nlib, and my_eco.v in the run directory.

If enet_block_name/label is specified as `-output my_eco/my_label`, Fusion Compiler targeted synthesis writes out the resulting E-NET as `my_eco` or `my_label` in the O-NET nlib and my_eco_my_label.v in the run directory.

If enet_nlib_name:enet_block_name is specified as `-output my_local_nlib/my_eco`, Fusion Compile targeted synthesis writes out the resulting E-NET as `my_eco` in my_local_nilb and my_eco.v in the run directory.

If enet_nlib_name:enet_block_name/label is specified as `-output my_local_nlib/my_eco/my_label`, Fusion Compiler targeted synthesis writes out the resulting E-NET as `my_eco` or `my_label` in my_local_nilb and my_eco_my_label.v in the run directory.

`-pre_region_script` **`file_name`**

Specifies the path and name of the script to be executed before ECO region creation. This can be used in the Fusion Compiler full synthesis flow if you have constraints declared on lower level cells that are moved into ECO regions and these constraints cause errors when they are sourced after the `set_top_module` command triggers ECO region creation. Move these constraints into a Tcl script and use this option to source those constraints before ECO region creation takes place, for example, `-pre_region_script my_lower_cell_constraints.tcl`. This option is used in full and targeted synthesis flows for the Fusion Compiler tool.

`-pre_compile_script` **`file_name`**

Specifies the path and name of the script to be executed before compile. This is an optional argument and can be used as needed, for example, `-pre_compile_script pre_compile_report.tcl`. This option is used only with the targeted synthesis flow.

`-post_compile_script` **`file_name`**

Specifies the path and name of the script to be executed after compile. This is an optional argument and can be used as needed, for example,

`-post_compile_script post_compile_report.tcl`. This option is used only with the targeted synthesis flow.

Description

The `set_fm_eco_mode` command enables the Formality ECO flow which streamlines the process for automatic ECOs. The flow has two flavors, the full synthesis flow, and the targeted synthesis flow. If you chose the `-netlist` option, you enable the targeted synthesis flow. The `set_fm_eco_mode` command automatically enables the guide hierarchical map flow, which requires the `set_verification_top` command in Design Compiler and the `set_top_module` command in Fusion Compiler. The `set_fm_eco_mode` command also modifies some tool settings in Design Compiler and Fusion Compiler to ensure successful completion of the Formality ECO flow including stopping the failure to create ECO regions.

The Formality ECO flow uses the same synthesis scripts as the original RTL synthesis flow, with the addition of the `set_fm_eco_mode` command. The command must be invoked as the first step in the script before any designs are read and before the `set_svf` command is invoked.

The Formality ECO flow needs an FRD (Formality Region Data) file, generated as part of the Formality pre-synthesis step. This file encapsulates all the region data for the Formality ECO flow.

Use the `-pre_region_script` option to apply any settings that need to be done before region creation in the Formality ECO flow.

Use the `-pre_compile_script` option to apply any settings that need to be done before compile in the Formality ECO flow.

Use the `-post_compile_script` option to apply any settings that need to be done after compile in the Formality ECO flow.

Examples

This command triggers the full Formality ECO flow with the eco.frd region file:

```
prompt> set_fm_eco_mode -region ./eco.frd
```

The following command triggers the targeted synthesis flow with the eco.frd region file and the O-NET from O-RTL synthesis:

```
prompt> set_fm_eco_mode -region ./eco.frd
 -netlist ./onet_path/compile_initial_opto.ddc
```

Feedback

Full Synthesis Flow Scripts

This section provides examples for all the scripts used in the full synthesis ECO flow:

- Match ECO region script
- ECO synthesis scripts for Design Compiler and Fusion Compiler
- Create ECO patch script
- Confirm ECO patch script

Example 13 shows the match ECO region script:

Example 13 Match ECO Region 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 to load it later in
## this script. See appendix for more details

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

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

```
## For designs with UPF flow,donot read the UPF
## 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
## Check the section "User setup for match_eco_region" for more details
# 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

## 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
## More details on these commands can be
## found in the appendix section
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
## which will be used later in the flow
##    a.) fm_eco_region.frd
##    b.) fm_eco_region.group.tcl
##    c.) fm_eco_region.data.tcl
##    d.) fm_eco_patched_orig.svf; Please check the section "Designs
##with checkpoints" for more details

write_eco_regions -replace

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

Example 14 and Example 15 show the ECO synthesis scripts for the Fusion Compiler and Design Compiler tools.

Example 14 Fusion Compiler Script

```
## Use the ECO mode
set_fm_eco_mode \
-region            ./ECO_WORK/fm_eco.frd


set_svf eco_full_synth_flow.svf

set search_path "./libs ./eco_rtl . ./libs"

set ref_path "./libs"
set tech_lib_file "${ref_path}/xxx_tech.ndm"
set ref_libs "${ref_path}/xxx_hvt.ndm ${ref_path}/xxx_tech.ndm"

create_lib -use_technology_lib $tech_lib_file -ref_libs $ref_libs
 ENET.nlib

analyze -format verilog "rtl1.v rtl2.v rtl3.v design_top.v"
elaborate design_top
 set_top_module design_top
 ….
 compile_fusion {-to initial_opto}
 save_block -as ENET
 save_lib
```

Example 15 Design Compiler Script

```
## Use the ECO mode
set_fm_eco_mode \
-region ./ECO_WORK/fm_eco.frd

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_full_synth_flow.svf
## Ensure that Design Compiler creates guide_hier_map SVF guidance
set_app_var hdlin_enable_hier_map true

# ECO RTL
analyze -format verilog "rtl1.v rtl2.v rtl3.v design_top.v"
elaborate design_top
current_design design_top

## Ensure that you use the set_verification_top command
set_verification_top
…..
compile_ultra -scan -gate_clock -no_autoungroup
```

```
change_names -rules verilog -hierarchy
write_file -format verilog -hierarchy -output eco_netlist.v
write_file -format ddc -hierarchy -output eco_netlist.ddc
```

Example 16 shows the create ECO patch script.

Example 16 Create ECO Patch Script

```
set synopsys_auto_setup true

## Read only E-SVF for the Full Synth Flow
set_svf ./eco_full_synth_flow.svf

read_db "some_tech_lib.db"

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

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

## For designs with UPF flow,donot read the UPF
## 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
## Refer to the section above “User Setup for create_eco_patch”

# 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

current_container r
source fm_eco_region.group.tcl

## For debugging purposes replace "match" with the lines below.
## The resulting verification of E-RTL and E-NET should have no failures.
## This will also perform the same function as "match".
##
# set verification_effort_level super_low
# verify
## Set up containers, source the data file to read ECO regions
## Create patch file, and create ECO region SVF file
```

```
if { [ match ] } {
  set_eco_ref r
  set_eco_imp i
  set_orig_imp onet
  source fm_eco_region.data.tcl

## Generates fm_eco_region.patch.tcl, fm_eco_region.svf
## For multibit designs, additional files
##fm_eco_region.confirm_setup.tcl andfm_eco_region.netlist_flow_setup.tcl
##will be generated
 create_eco_patch -replace

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

## Reload a fresh copy of the original netlist container, and apply the
## patch to get the final ECO Tcl script (netlist edit commands)
remove_container i

## Reload (or create again) the original netlist container
read_container -i ./ECO_WORK/onet_imp.fsc -replace

## Source the FM patch just created
source ./fm_eco_region.patch.tcl

## Write out final ECO Tcl script (netlist edit commands) for use in ICC
## (ICC2/DC/FC)
write_edits -replace eco.edits.tcl

quit
```

Example 17 shows the confirm ECO patch script.

Example 17 Confirm ECO Patch Script

```
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)
 set_svf eco_change.svf
if {[file exists fm_eco_patched_orig.svf]} {
set_svf -append fm_eco_patched_orig.svf
} else {
  set_svf -append ./DC/orig.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

## 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


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

match
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}
```