cadence®

Innovus Foundation Flows Guide

Product Version 21.10
May 2021

Contents

About This Manual

This manual describes the tasks in the recommended Foundation Flow using the

Cadence® Innovus™ Implementation System software to implement a block or flat chip from a completed floorplan, while meeting timing and physical design requirements and resolving signal integrity (SI) problems.
The Innovus family encompasses the following products:

- Innovus Implementation System L

- Innovus Implementation System XL

- NanoRoute® Ultra SoC Routing Solution

- Virtuoso® Digital Implementation

- Innovus Timing System L

- Innovus Timing System XL

- Innovus Power System L

- Innovus Power System XL

- First Innovus™ L

- First Innovus XL

- First Innovus GXL

Note: This document applies to the Innovus user interface.

For information about these products, see the "Getting Started" chapter of the *Innovus User Guide*.

For more information about the Innovus family of products, see the following documents. You can access these and other Cadence documents with the Cadence Help documentation system.

Innovus Product Documentation

- *What's New in Innovus*
  Provides information about new and changed features in this release of the Innovus family of products.

- *Innovus Text Command Reference*
  Describes the Innovus text commands, including syntax and examples.

- *Innovus Menu Reference*
  Provides information specific to the forms and commands available from the

Innovus graphical user interface.

- *Innovus Database Access Command Reference*
  Lists all of the Innovus database access commands and provides a brief description of syntax and usage.

- *Innovus User Guide*
  Describes how to install and configure the Innovus software, and provides strategies for implementing digital integrated circuits.

- *Mixed Signal Interoperability Guide*
  Describes the digital mixed-signal flow.

- README file
  Contains installation, compatibility, and other prerequisite information, including a list of Cadence Change Requests (CCRs) that were resolved in this release. You can read this file online at downloads.cadence.com.

Innovus Stylus Common UI Product Documentation

- *Innovus Stylus Common UI Quick Start Guide*
  Provides information on starting with the Stylus Common User Interface.

- *What's New in Innovus Stylus Common UI*
  Provides information about new and changed features in this release of the Innovus family of products.

- *Innovus Stylus Common UI User Guide*
  Describes how to install and configure the Innovus Stylus Common UI software, and provides strategies for implementing digital integrated circuits.

- *Innovus Stylus Common UI Text Reference Manual*
  Describes the Innovus Stylus Common UI text commands, including syntax and examples.

- *Innovus Stylus Common UI Menu Reference*
  Provides information specific to the forms and commands available from the Innovus Stylus Common UI graphical user interface.

- *Stylus Common UI Database Object Information*
  Provides information about Stylus Common UI database objects.

- *Innovus Stylus Common UI Mixed Signal (MS) Interoperability Guide*
  Describes the digital mixed-signal flow using Innovus Stylus Common UI.

For a complete list of documents provided with this release, see the Cadence Help online documentation system.

1

Overview

In this section we cover a basic introduction to how foundation flows are implemented in Innovus.

- Introduction to Foundation Flow

- Foundation Flow Overview Diagram

- Before You Begin

- Input for Running Foundation Flows

- Additional Inputs for Running Foundation Flows

- Flow Variations

- About the Flow Documentation

Introduction to Foundation Flow

This guide describes the tasks in the recommended flow using the Innovus Implementation System (Innovus) software to implement a block or a flat chip from a completed floorplan, while meeting timing and physical design requirements and resolving signal integrity (SI) problems. The guide also includes Low Power and Hierarchical flows.

For designs with multiple operating modes or corners that require optimization, and for more accurate timing, the flow supports multi-mode multi-corner (MMMC) timing analysis and on-chip variation (OCV). From EDI 10.1, the flow takes advantage of the Innovus software multiple-CPU processing capabilities to accelerate the design process and runs features in a multi-thread or distributed process-mode.

For the main design tasks the flow uses the Innovus super commands. For example, `place_opt_design`, `routeDesign`, and `optDesign` with as few non-default options as possible.

When the tasks described in this guide are complete, you should have a flat design that is ready for DRC and LVS checks and other sign-off tasks.

This flow is a recommended starting point for you to work with designs for a block or a flat chip. Ultimately, your design might have a different final set of commands, owing to your specific customization for your design or technology needs.

Foundation Flow Overview Diagram

A completed floorplan is a prerequisite for a flat implementation flow. At the end of each main task you optimize the step and the task is complete when violations are fixed and the timing is acceptable. The flow is complete and the design is ready for sign-off tasks when the design meets timing and physical design requirements.

* At any point in the flow, go back to an earlier step to resolve the unsatisfactory timing.
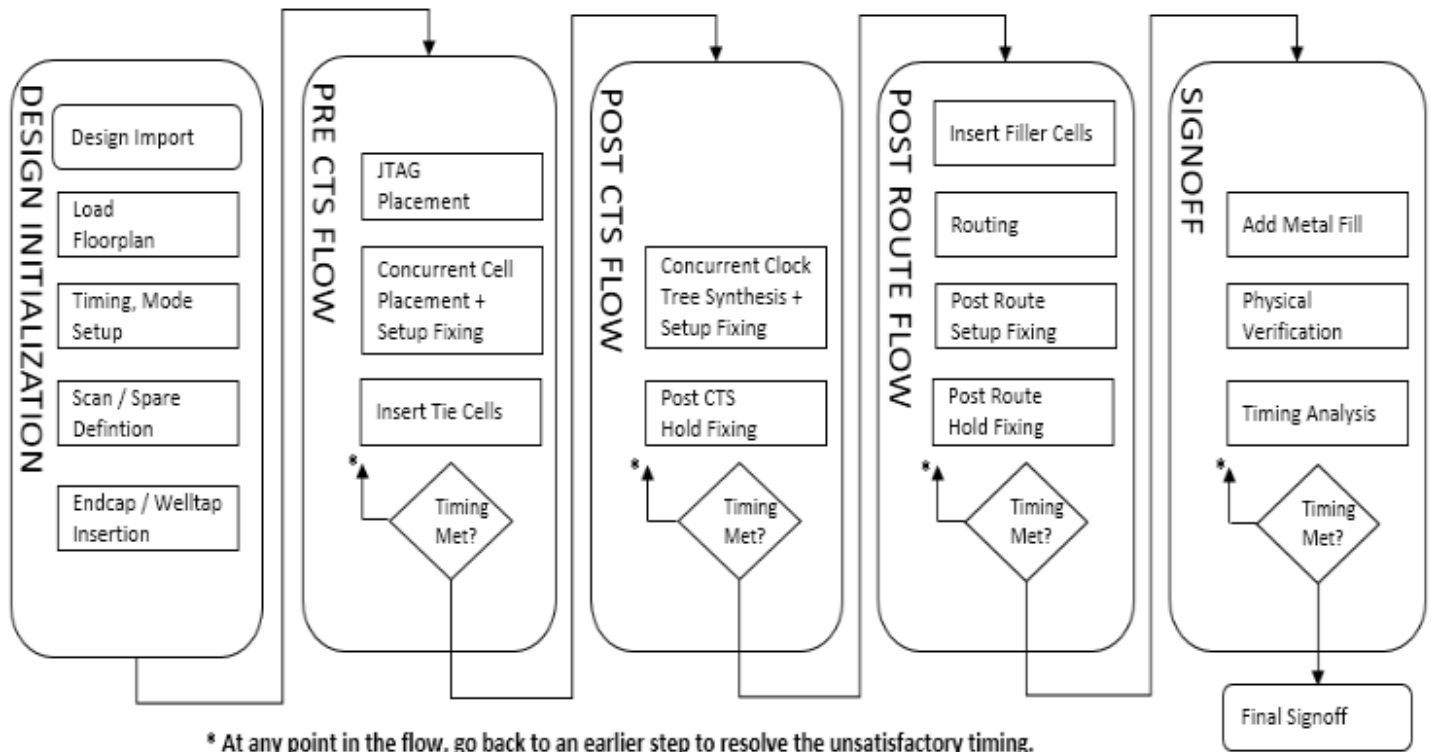
Before You Begin

Make sure the following tasks are complete before starting this flow:

- Floorplanning, including the following tasks:
  - Hard block placement
  - Region definition
  - I/O pad placement
  - Creation of placement and routing blockages
- Creating the Clock Tree specification file

- Creating the RC scale factors

For more information, see the following sections of the *Innovus User Guide*

- Floorplanning the Design

- Synthesizing Clock Trees

- RC Extraction

Input for Running Foundation Flows

The foundation flows are driven by a combination of TCL scripts and Makefiles. The command scripts are provided in the software when you install the tool. The user provides library data and options to customise the flow by means of TCL variables. The flow expects the library data to be in a file called setup.tcl. A full description of the required and optional variables is provided later in this guide.

Specify the Innovus configuration file (setup.tcl) with the following information:

- Timing libraries (*.lib)

- LEF libraries (*.lef)

- Timing constraints (*.sdc)

- Capacitance table or QRC technology file

- SI libraries (*.cdb or *.udn) (this is recommended but not required)

- User-defined ILMs

- Verilog netlist (*.v)

- Floorplan file (*.fp or *.def)

- Clock Tree Specification file

- Scan Chain information (*.tcl or *.def)

- GDS Layer map file (*.gds)

Note: The Flat Implementation Flow assumes that timing and routing constraints are feasible.

Additional Inputs for Running Foundation Flows

In addition, depending on the design and technology, you may need to supply values for the following items:

- RC scaling factors
- OCV derating factors
- Metal fill parameters
- Filler cell names
- Tie cell names
- Welltaps and Endcaps
- Clock gating cell names
- Spare instance names
- JTAG instance names
- JTAG rows
- LSF queue
- Dont Use Cells
- Delay Cells
- CTS Cells

Flow Variations

Based on the timing model you use, there are three variations of the flat implementation flow:

- MMMC: the flow uses the MMMC timing throughout the flow. You may use the MMMC flow in designs that require optimization at multiple operating modes or at multiple corners.

The following two are supported for backward compatibility but are not recommended:

- Default: the flow uses the default timing model until the final Timing Analysis step, when it changes to MMMC timing. Though the default flow is a non-MMMC flow, it changing to MMMC

timing for the final timing analysis step gives the most accurate sign-off timing.

- Postroute MMMC: the flow uses the default timing model for the Preroute portion of the flow and MMMC timing after Routing. You may use the Postroute MMMC flow in designs that require improved co-relation with third-party tools.

Note: Because MMMC timing incorporates timing information for more modes and corners, the postroute MMMC and MMMC flows may incur memory and run-time degradation.

About the Flow Documentation

This guide includes flow charts for the overall Flat Implementation Flow and for each of the major design tasks, along with brief explanations. It also includes the following additional information in the Cadence Innovus documentation:

- Command syntax and parameter descriptions in the Innovus Text Command Reference document.

- Task information in the Innovus User Guide

2

TCL Variables

In this section we cover the Tool Command Language (Tcl) variables and descriptions for foundation flows.
The section covers:

- Installing Foundation Flows

- Starting Foundation Flows

Installing Foundation Flows

All of the Innovus Foundation Flows rely on the Tcl scripts to keep the flows free from errors and easy to manage. These Tcl scripts are sourced from the main flow script.
You can download the scripts from the software by using one of the following methods:

- On the main Innovus menu, select Tools - Foundation Flow Templates on the main Innovus menu.

- Use the following text command:

  ```
  writeFlowTemplate [-directory directoryName]
  ```

This will create the following directories:

- SCRIPTS (the Foundation Flow scripts)

- TEMPLATES (Configuration File templates)

- EXAMPLES (working design example and sample configuration files)

Starting Foundation Flows

The Foundation Flow requires a configuration file (setup.tcl) that defines the necessary information to execute the flow including pointers to design data and command option information. The full list of variables used to control the flow can be found in the documentation or in the examples in the TEMPLATES directory in the software.
In addition, some of the variable values can be quickly populated from an existing design in the following way:

1. Load a design (initial netlist, constraints, floorplan)

2. Run the flow wizard (*Flows > Foundation Flow Wizard*) OR enter the command:
   `generateFFSetupFile`

3. This will generate the following files: setup.auto.tcl and innovus_config.auto.tcl

4. Respectively copy to setup.tcl and innovus_config.tcl.

5. (optional) Review and edit the files, if necessary.
   Or: Copy the template files from TEMPLATES/Innovus and edit them manually.

The following Tcl scripts are described in this guide:

- Describing setup.tcl
  Defines the variables to use in the flow. This script is unique for each design and is the only script you must edit. It is the source of all flow input, and is a superset of the configuration file that includes design data, library information, and flow control.

- Describing innovus_config.tcl
  This is the Innovus Foundation Flow configuration file. It contains the necessary flow options to drive the Flat and/or Hierarchical implementation flows. This file is optional and is intended to support design projects where the `setup.tcl` is shared between team members and defines common design data such as library, timing, and technology information and the `innovus_config.tcl` is a local file that contains flow related information that is unique to a particular block or run.

- Example Settings for Each Script
  Contains the Metal Fill rules.

See also,

- Describing setup.tcl

- Defining Extraction Options

- Describing innovus_config.tcl

- Example Settings for Each Script

Describing setup.tcl

Defines the variables to use in the flow. This script is unique for each design and is the only script that is required. It contains variable settings that define the necessary information to drive the foundation flow; including design data, library information, and flow control.
The following variables are defined in setup.tcl. They are put into an array named vars.
To use them enter:

```
set vars variable_name value
```

Note: The vars in the column Variable Name in italics are actually placeholders and can be substituted with your own values. For example in *rc_max*,cap_table the value *rc_max* is a placeholder and can be a value you want to specify in cap_table.

You can also specify a `view_definition_tcl` file in place of the MMMC vars. For this, you must define either of the following:

`vars(view_definition_tcl)`

OR

`vars(library_sets)`, `vars(rc_corners)`, `vars(delay_corners)`, `vars(constraint_modes)` and `vars(setup/hold_analysis_view)` variables.

Note: The selections mentioned above cannot be mixed.

The following types of variables are covered in this section:

- Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

- Defining Hierarchical Partition Information

- Defining Library and Technology File

- Example of Library and Technology File

- Defining RC Corner Information

- Example of RC Corner Information

- Defining Delay Corner Information

- Example of Delay Corner Information

- Defining Constraint Modes and Analysis Views

- Example of Constraint Modes and Analysis Views

- Defining OpenAccess

- Mailing Results and Updates

- Miscellaneous Variables

Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

Variable Name	Value Type	Usage Description
fp_file	*string*	Specify the name and location of the Innovus floorplan file. Do not use the variable if specifying the def_files variable.
def_files	*name*	Specify the name of the initial DEF file. The fp_file is the alternate recommended file.
cts_spec	*list*	This file is optional. If omitted Innovus will use the SDC to build the clock tree.
max_route_layer	*integer*	Specify the number of routing layers
process	*integer*	Specify the process node between min=10, max=250 (nm)
scan_def	*name*	This is optional, specify the name of the Scan DEF file
dbs_dir	*string*	Specify the string of the database directory (the default is DBS)
fp_tcl_file	*text*	script to build floorplan (overrides fp_file)
fp_tcl_proc	*text*	TCL procedure to build floorplan (overrides fp_file, fp_tcl_file)
rpt_dir	*string*	Specify the string of the report directory (the default is RPT)
<step>,rpt_dir	*string*	Specify the report directory for a specific step. If this is not set, vars(rpt_dir) is used.
script_root	*string*	Specify the location of SCRIPTS directory (the default is SCRIPTS)
design	*name*	Name of the design (required)
add_tracks	*boolean*	Re-generate tracks as against using what is in the incoming floorplan. The default value is false.
netlist	*string*	Pointer to netlist file (required)

version	*integer*	Innovus version (9.1.2, 9.1.3, 10.1.0, ...)
cell_check_early	*float*	Specify for derate cell library checks
cell_check_late	*float*	Specify for derate cell library checks
honor_pitch	*boolean*	An option for addTracks command (enabled by vars (add_tracks)
design_root	*name*	Name of the root directory of the design (required)
critical_range	*float*	Specify critical range for TNS optimization

For other `setup.tcl` variables, see:

Defining Library and Technology File

Defining RC Corner Information

Defining Delay Corner Information

Defining Constraint Modes and Analysis Views

Mailing Results and Updates

Miscellaneous Variables

For `Innovus_config.tcl` variables see Describing Innovus_config.tcl

For examples see Example Settings for Each Script

Defining Hierarchical Partition Information

Defining Hierarchical Partition Information

Varible Name	Value Type	Usage Description
`set vars(placement_based_ptn) 1`	*float*	Specify the option to run placement based partition flow (feed through insertion/pin assignment). *Default:* 0 (to run route based partition flow)
`set vars(insert_feedthrough) 1`	*float*	Specify the option to run feedthrough buffer insertion. *Default:* 1
`set vars(abutted_design) 1`	*float*	Specify the option to support channelless design. *Default:* 0
`set vars(use_proto_net_delay_model) 1`	*float*	Specify the option to use pico-second per micro model for quick timing estimation. *Default:* 0
`set vars(use_flexmodels) 1`	*float*	Specify this variable to enable FlexModel flow to reduce netlist size. *Default:* 0
`set vars(flexmodel_as_ptn) 1`	*float*	Specify the option to specify one FlexModel per partition in a design. *Default:* 1
`set vars(budget_mode) giga_opt`	*float*	Specify this variable to enable timing budgeting with GigaOptvirtual. Possible values are `giga_opt` and `proto_net_delay_model`. *Default:* `giga_opt`

For other `setup.tcl` variables, see:

Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

Defining Library and Technology File

Defining RC Corner Information

Defining Delay Corner Information

Defining Constraint Modes and Analysis Views

Defining OpenAccess

Mailing Results and Updates

Miscellaneous Variables

For `edi_config.tcl` variables see Describing innovus_config.tcl

For examples see Example Settings for Each Script

Defining Library and Technology File

Variable Name	Value Type	Usage Description
`lef_files`	*list*	Specify the list name of the LEF file
`library_sets`	*list*	Specify one or more library set names for slow and fast processes. The library_sets must be one of the Constraint Modes and a list member.
`enable_ldb`	list	Pre-compiles .lib to .ldb.
`library_set`,`timing`	*list*	Specify the .lib files
`library_set`,`si`	*list*	Specify the .cdb files (optional)
`library_set`,`aocv`	*list*	Points to the AOCV tables which are required for AOCV analysis

Also see Example of Library and Technology File

For other `setup.tcl` variables, see:

Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

Defining RC Corner Information

Defining Delay Corner Information

Defining Constraint Modes and Analysis Views

Defining OpenAccess

Mailing Results and Updates

Miscellaneous Variables

For `edi_config.tcl` variables see Describing innovus_config.tcl

For examples see Example Settings for Each Script

Example of Library and Technology File

Example of Library and Technology File

Variable Name	Value Type
`set vars(library_sets)`	`slow fast`
`set vars(slow,timing)`	`libs/slow.lib libs/rams_slow.lib`
`set vars(fast,timing)`	`libs/fast.lib libs/rams_fast.lib`
`set vars(slow,si)`	`libs/slow.cdb`
`set vars(fast,si)`	`libs/fast.cdb`

Defining RC Corner Information

Variable Name	Value Type	Usage Description
`rc_corners`	*list*	Specify the RC corner names
`rc_corner,cap_table`	*list*	Specify the cap table per RC corner
`rc_corner,T`	*list*	Specify the temperature
`rc_corner,qx_tech_file`	*list*	Specify the QRC technology file
`rc_corner,qx_lib_file`	*list*	Specify the QRC library file
`rc_corner,qx_conf_file`	*list*	Specify the QRC configuration file

rc_corner,*xcap_factor*	*list*	Specify the scaling factor for coupling capacitance (post-route)
rc_corner,*scale_factor*	*float**	Specify scaling factors; valid factors are: pre_route_res_factor, pre_route_clk_res_factor pre_route_cap_factor, pre_route_clk_cap_factor, post_route_res_factor, post_route_clk_res_factor, post_route_cap_factor, post_route_clk_cap_factor, post_route_xcap_factor Note: *post_route* factors can take a triplet for low medium high effort scale factors.
rc_corner,pre_route_res_factor	*float*	Specify the scaling factor for resistance (pre-route) Note: Has replaced variable def_res_factor **
rc_corner,pre_route_clk_res_factor	*float*	Specify the scaling factor for clock resistance (pre-route) Note: Has replaced variable def_clk_res_factor **
rc_corner,pre_route_cap_factor	*float*	Specify the capacitance scaling factor (pre-route) Note: Has replaced variable def_cap_factor **
rc_corner,pre_route_clk_cap_factor	*float*	Specify the scaling factor for clock capacitance (pre-route) Note: Has replaced variable def_clk_cap_factor **
rc_corner,post_route_res_factor	*float*	Specify the scaling factor for resistance (post-route) Note: Has replaced variable det_res_factor **

rc_corner,`post_route_clk_res_factor`	*float*	Specify the scaling factor for clock resistance (post-route) Note: Has replaced variable det_cap_factor **
rc_corner,`post_route_cap_factor`	*float*	Specify the capacitance scaling factor (post-route) Note: Has replaced variable det_cap_factor **
rc_corner,`post_route_clk_cap_factor`	*float*	Specify the scaling factor for clock capacitance (post-route) Note: Has replaced variable det_clk_cap_factor **
rc_corner,`post_route_clk_res_factor`	*float*	Specify the resistance scaling factor (post-route) Note: Has replaced variable det_clk_res_factor **
rc_corner,`post_route_xcap_factor`	*float*	Specify the scaling factor for coupling capacitance (post-route) Note: Has replaced variable xcap_factor **
rc_corner,`scale_tcl`	*file*	Pointer to a file with update_rc_corner command that sets the scale factors appropriately for the RC corner.

Note: ** For backward compatibility, the script retains both the variables.

Also see Example of RC Corner Information

For other `setup.tcl` variables, see:

Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

Defining Library and Technology File

Defining Delay Corner Information

Defining Constraint Modes and Analysis Views

Defining OpenAccess

Mailing Results and Updates

Miscellaneous Variables

For `edi_config.tcl` variables see Describing innovus_config.tcl

For examples see Example Settings for Each Script

Example of RC Corner Information

Variable Name	Value Type
`set vars(rc_corners)`	`worst best`
`set vars(worst,cap_table)`	`tech/worst/capTable`
`set vars(worst,T)`	125
`set vars(best,cap_table)`	`tech/best/capTable`
`set vars(best,T)`	0

Defining Delay Corner Information

Variable Name	Value Type	Usage Description
`delay_corners`	*list*	Specify the delay corner names
`delay_corner,library_set`	string	Delay corner library set name
`delay_corner,rc_corner`	string	Delay corner RC corner name
`delay_corner,derate_factor`	float	Specify derating factors*. Note: The valid factors are listed below this table.
`delay_corner,cell_worst_late`	float	Specify the setup for derating of cells
`delay_corner,cell_worst_early`	float	Specify the hold derating for cells
`delay_corner,cell_best_late`	float	Specify the setup for derating of cells
`delay_corner,cell_best_early`	float	Specify the hold derating for cells
`delay_corner,wire_worst_late`	float	Specify the setup derating for nets

delay_corner,wire_worst_early	float	Specify the hold derating for nets
delay_corner,wire_best_late	float	Specify the setup derating for nets
delay_corner,wire_best_early	float	Specify the hold derating for nets
delay_corner,derate_tcl	float	Allows the user to have a delay_corner.derate.tcl file for each delay corner instead of having to set one file per delay in the setup.tcl. Foundation flow also has an analogous *rc*,scale_tcl variable for RC corners called *rc_corner*,scale_tcl
delay_corner,*cell_check_early*	float	Specify the check early for cells
delay_corner,*cell_check_late*	float	Specify the check late for cells
set vars(*delay_corner*,*late_opcond_library*)	*list*	Specify the internal library name for the library in which the late operating condition is defined. This is not the library file name.
set vars(*delay_corner*,*early_opcond_library*)	*list*	Specify the internal library name for the library in which the early operating condition is defined. This is not the library file name.
set vars(*delay_corner*,*late_opcond*)	*list*	Specify the operating condition to use for calculating late arrival times at a single delay corner.
set vars(*delay_corner*,*early_opcond*)	*list*	Specify the operating condition to use for calculating early arrival times at a single delay corner.

*Derating Factors

The valid derating factors are described below:

Variable Name	Usage Description
data_cell_late	Specify setup derating of cell
data_cell_early	Specify hold derating of cell

`data_net_late`	Specify setup derating of net
`data_net_early`	Specify hold derating of net
`clock_cell_late`	Specify hold derating of clock cell
`clock_cell_early`	Specify setup derating of clock cell
`clock_net_late`	Specify hold derating of clock net
`clock_net_early`	Specify setup derating of clock net
`set_timing_derate`	Specify after CTS in case you did not define derating on initialization

Support for Power Domain - Delay Corner Binding Via

Variable Name	Usage Description
`set vars(library_sets)`	List of library sets
`set vars(delay_corners)`	List of delay corners
`set vars(power_domains)`	List of power domains to bind
`set vars(dc,power_domains)`	List of power domains
`set vars(dc,ls,power_domains)`	List of power domains. This allows bind to a specify delay corner AND library set

Support for Power Domain - Per Delay Corner AND Per Library Set

Variable Name	Usage Description
`set vars(library_sets)`	List of library sets
`set vars(delay_corners)`	List of delay corners
`set vars(power_domains)`	List of power domains to bind
`set vars(dc1,pd1,early_library_set)`	List of delay corners, power domains, and early library sets
`set vars(dc1,pd1,late_library_set)`	List of delay corners, power domains, and late library sets
`set vars(dc2,pd2,library_set)`	List of delay corners, power domains, and library sets

Also see Example of Delay Corner Information

For other `setup.tcl` variables, see:

Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

Defining Library and Technology File

Defining RC Corner Information

Defining Constraint Modes and Analysis Views

Defining OpenAccess

Mailing Results and Updates

Miscellaneous Variables

For `edi_config.tcl` variables see Describing innovus_config.tcl

For examples see Example Settings for Each Script

Example of Delay Corner Information

Variable Name	Value Type
`set vars(worst,pre_route_cap_factor)`	1.08
`set vars(worst,pre_route_res_factor)`	1.12

Defining Constraint Modes and Analysis Views

Variable Name	Value Type	Usage Description
`constraints_modes`	list	Specify the constraint mode names
constraint_mode`,pre_cts_sdc`	list	Specify the Pre CTS SDCs
constraints_mode`,post_cts_sdc`	list	Specify the Replacement Post CTS SDCs (optional)
constraints_mode`,incr_cts_sdc`	list	Specify the Incremental Post CTS SDCs (optional)
`analysis_views`	list	Specify the analysis view names
analysis_view`,delay_corner`	string	Specify the delay corners
analysis_view`,constraint_mode`	setup_func	Specify the constraint mode
`setup_analysis_views`	list	Specify the setup analysis views
`hold_analysis_views`	list	Specify the hold analysis views
`default_setup_view`	*list*	Specify the default setup views
`default_hold_view`	*list*	Specify the default hold views
step`,active_setup_views`	*list*	Specify the active setup views
step`,active_hold_views`	*list*	Specify the active hold views
step`,setup_analysis_views`	*list*	Specify the setup analysis views
step`,hold_analysis_views`	*list*	Specify the hold analysis views

See also Example of Constraint Modes and Analysis Views

For other `setup.tcl` variables, see:

Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

Defining Library and Technology File

Defining RC Corner Information

Defining Delay Corner Information

Defining OpenAccess

Mailing Results and Updates

Miscellaneous Variables

For `edi_config.tcl` variables see Describing edi_config.tcl

For examples see Example Settings for Each Script

Example of Constraint Modes and Analysis Views

Variable Name	Value Type
`set vars(constraint_modes)`	`func_mode test_mode`
`set vars(func_mode,pre_cts_sdc)`	`sdc/functional.sdc`
`set vars(test_mode,pre_cts_sdc)`	`sdc/test_mode.sdc`
`set vars(func_mode,post_cts_sdc)`	`sdc/postcts.sdc`
`set vars(delay_corners)`	`slow_worst fast_best`
`set vars(slow_worst,library_set)`	`slow`
`set vars(slow_worst,rc_corner)`	`worst`
`set vars(fast_best,library_set)`	`fast`
`set vars(fast_best,rc_corner)`	`best`
`set vars(slow_worst,clock_cell_early)`	0.95
`set vars(fast_worst,clock_cell_late)`	1.05

`set vars(analysis_views)`	`func_slow_worst func_fast_best test_slow_worst test_fast_best`
`set vars(func_slow_worst,delay_corner)`	`slow_worst`
`set vars(func_slow_worst,constraint_mode)`	`func_mode`
`set vars(func_fast_best,delay_corner)`	`fast_best`
`set vars(func_fast_best,constraint_mode)`	`func_mode`
`set vars(test_slow_worst,delay_corner)`	`slow_worst`
`set vars(test_slow_worst,constraint_mode)`	`test_mode`
`set vars(test_fast_best,delay_corner)`	`fast_best`
`set vars(test_fast_best,constraint_mode)`	`test_mode`
`set vars(setup_analysis_view)`	`func_slow_worst test_slow_worst`
`set vars(hold_analysis_view)`	`func_fast_best test_fast_best`
`set vars(active_setup_views)`	`func_slow_worst`
`set vars(active_hold_views)`	`func_fast_best test_fast_best`
`set vars(default_setup_view)`	`func_slow_worst`
`set vars(default_hold_view)`	`func_fast_best`

Defining OpenAccess

Defining OpenAccess

Varible Name	Value Type	Usage Description
`oa_ref_lib`	*string*	Specify the list of OpenAccess reference libraries
`oa_abstract_name`	name	Specify the list of OpenAccess abstract name(s) for the libraries
`oa_layout_name`	*string*	Specify the list of OpenAccess layout view(s) for the libraries
`dbs_format`	*string*	Specify the format for the intermediate database saves. Use the value oa for OpenAccess. oa_design_lib and oa_design_cell are also required in this situation.
netlist_type	*string*	Specify the format of the initial database. Valid options are verilog (default) or oa. If oa is specified the following three oa_design_* variables are also required.
`oa_design_lib`	*string*	Specify the OpenAccess design library
`oa_design_cell`	strings	Specify the OpenAccess design cell to read and write
`oa_design_view`	strings	Specify the OpenAccess design view to import at the start of the flow.
oa_fp	*string*	Specify the OpenAccess floorplan support.

When running the flow using OpenAccess libraries there are two ways in which to initalise the flow. With a netlist and floorplan or with an already initialised OA view.

If the design database needs to interoperable with virtuoso it may be necessary to use the setOaxMode command. This should be included in a plug script.

For other `setup.tcl` variables, see:

Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

Defining Library and Technology File

Defining RC Corner Information

Defining Delay Corner Information

Defining Constraint Modes and Analysis Views

Mailing Results and Updates

Miscellaneous Variables

For `edi_config.tcl` variables see Describing innovus_config.tcl

For examples see Example Settings for Each Script

Mailing Results and Updates

Variable Name	Value Type	Usage Description
mail,to	*text*	Specify the address
mail,steps	*text*	Specify the procedure

For other `setup.tcl` variables, see:

Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

Defining Library and Technology File

Defining RC Corner Information

Defining Delay Corner Information

Defining Constraint Modes and Analysis Views

Defining OpenAccess

Miscellaneous Variables

For `edi_config.tcl` variables see Describing edi_config.tcl

For examples see Example Settings for Each Script

Miscellaneous Variables

Variable Name	Value Type	Usage Description
`flat`	*[off, partial, full]*	Controls the level of script unrolling
`verbose`	*boolean*	Adds comments when true
`tags,verbose`	*boolean*	Adds tag comments when true
`tags,verbosity_level`	*[low	high]*	When high adds comments for ALL possible tags in each flow script
`log_dir`	name	Directory for log files
`plug_dir`	string	Directory for plug-ins
`skew_buffers`	*list*	Specify list of buffers to use during useful skew
`set vars(enable_flexilm)`	*boolean*	Supports FlexILM for preCTS top-level timing closure. *Default:* ILM based flow

For other `setup.tcl` variables, see:

Defining Floorplanning, Scan, CTS, SDC, and Routing Layer

Defining Library and Technology File

Defining RC Corner Information

Defining Delay Corner Information

Defining Constraint Modes and Analysis Views

Defining OpenAccess

Mailing Results and Updates

For `edi_config.tcl` variables see Describing innovus_config.tcl

For examples see Example Settings for Each Script

Defining Extraction Options

Variable Name	Value Type	Usage Description
relative_c_thresh	*float*	Ratio of coupling cap to total cap
total_c_thresh	*float*	Total cap threshold for coupling
coupling_c_thresh	*float*	Coupling cap threshold
qrc_config_file	file	Specify configuration files for QRC extraction
qrc_layer_map	file	Specify layer map for QRC extraction
qrc_library	directory	Specify library for QRC extraction

For other `edi_config.tcl` variables, see

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Hier ILMs

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Describing innovus_config.tcl

This is an optional configuration file intended to support design projects where the `setup.tcl` is shared between team members and defines common design data such as library, timing, and technology information and the `innovus_config.tcl` is a local file that contains flow related information that is unique to a particular block or run.

The following variables are put in an array named vars and defined in `innovus_config.tcl`.

To use them enter:

```
set vars variable_name value
```

Note: The vars in the column Variable Name in italics are actually placeholders and can be substituted with your own values. For example in *rc_max*,cap_table the value *rc_max* is a placeholder and can be a value you want to specify in cap_table.

The following types of variables are covered in this section:

- Defining Noise Settings

- Defining Power and Ground Nets

- Defining Multi-Threading or Distributed Processing

- Defining CCOpt Variables

- Example of Distributed Processing

- Defining LP Flow Variables

- Defining Optional Variables

- Defining Tie and Filler Cells

- Defining Welltap and Endcap Specifications

- Defining Welltap and Endcap Specifications (Per power domain basis)

- Reporting Power

- Defining Hier ILMs

- Defining Flow Control

- Defining Command Mode Variables

- Defining ccopt_design for Clock Tree Construction

- Defining CCOpt Top and Bottom Layers

Defining Noise Settings

Variable Name	Value Type	Usage Description
delta_delay_threshold	*float*	Noise delay threshold

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Hier ILMs

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining Power and Ground Nets

Variable Name	Value Type	Usage Description
power_nets	*list*	List of power nets
ground_nets	*list*	List of ground nets

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Noise Settings

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining Multi-Threading or Distributed Processing

The variables can be one of the following: lsf, rsh, local, or custom. Depending on the value of the distribution, additional variables might be required.

Variable Name	Value Type	Usage Description
`distribute`	-	Specify the distribution style [local, custom, rsh, lsf]
`custom,script`	script	Specify the custom grid submission script
distribute,queue	*string*	Specify the LSF Queue (lsf)
distribute,resource	*string*	Specify the LSF Resources (lsf)
distribute,args	*string*	Specify the LSF Args (lsf)
distribute,`host_list`	*list*	Specify the list of hosts (rsh)
`distribute_timeout`	seconds	Tells how long to wait for a distributed host before it fails
`local_cpus`	*integer*	Specify the number of threads
`remote_hosts`	*integer*	Specify the number of hosts

`cpus_per_remote_host`	*integer*	Specify the number of CPUs per remote host

See also Example of Distributed Processing

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining LP Flow Variables

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Hier ILMs

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining CCOpt Variables

The following variables are added to Foundation Flow to enable Clock Current Optimization. CCOpt runs clock tree synthesis (CTS) concurrently with placement and physical optimization and uses a timing window driven engine to optimize timing paths and clocks simultaneously.

Variable Name	Description
`cts_engine`	Enables Clock Tree Synthesis, Clock Current Optimization and Clock Current Optimization enabled CTS (cts, ccopt, and ccopt_cts).
`ccopt_executable`	Enables the path to the CCOpt executable. This is optional.
`cts_buffer_cells`	List of usable buffers (required).

`cts_inverter_cells`	List of usable inverters (required).
`cts_use_inverters`	Enables the use of inverters (true or false).
`cts_target_slew`	Enables maximum clock tree transition time
`cts_target_skew`	Enables initial skew target.
`cts_io_opt`	Enables IO timing optimization (on, off, secondary).
`ccopt_effort`	Defines the effort level (low, medium, high).
`pre_ccopt_tcl`	Enables the native pre-ccopt plug-in.
`post_ccopt_tcl`	Enables the native post-ccopt plug-in.
`clk_tree_top_layer`	Enables the top tree preferred layer.
`clk_tree_bottom_layer`	Enables the bottom tree preferred layer.
`clk_tree_ndr`	Enables the NDR for tree nets.
`clk_leaf_top_layer`	Enables the top leaf preferred layer.
`clk_leaf_bottom_layer`	Enables the bottom leaf preferred layer
`cts_leaf_ndr`	Enables the NDR for leaf nets.
`clk_tree_shield_net`	Enables the shielding net (ccopt only).
`clock_gate_cells`	This is an existing variable for CTS but is also used for ccopt_design
`update_io_latency`	This is an existing variable for CTS but is also used for ccopt_design

Example of Distributed Processing

Example of Distributed Processing

Example 1

Variable Name	Example Value
`set vars(distribute)`	*custom*

`set vars(local_cpus)`	4
`set vars(remote_hosts)`	*4*
`set vars(custom,script)`	*/grid/sfi/farm/bin/bsub <run limit>* *<project name> <resource string> <queue name>*

Example 2

Variable Name	Example Value
`set vars(distribute)`	*lsf*
`set vars(local_cpus)`	*2*
`set vars(remote_hosts)`	4
`set vars(lsf,queue)`	*nx64*
`set vars(lsf,resource)`	*<resource string>* The execution host on which you want to run the job.
`set vars(lsf,args)`	*<run limit> <project name>*

Defining LP Flow Variables

Variable Name	Value Type	Usage Description
`cpf_timing`	boolean	Specifies the CPF timing.
`cpf_file`	*cpf_file_name*	Specifies the name of the CPF file. **IEEE1801 Mapping:** `vars(ieee1801_file)`
`resize_shifter_and_iso_insts`	boolean	Allows level shifter and resizing of isolation cell
`opconds`	list	Specifies the conditionals list.
opcond,`voltage`	voltage	Specifies the voltage.
opcond,`temperature`	temp	Specifies the temperature.
opcond,`process`	process scale factor	Specifies the process.

opcond, library_file	file	Specifies the library file.
cpf_isolation	boolean	Specifies the CPF isolation cells. **IEEE1801 Mapping:** vars(ieee1801_isolation)
cpf_keep_rows	boolean	Specifies the CPF rows to keep. **IEEE1801 Mapping:** vars(ieee1801_keep_rows)
cpf_level_shifter	boolean	Specifies the CPF level shifters. **IEEE1801 Mapping:** vars(ieee1801_level_shifter)
cpf_power_domain	boolean	Specifies the CPF power domains. **IEEE1801 Mapping:** vars(ieee1801_power_domain)
cpf_power_switch	boolean	Specifies the CPF power switches. **IEEE1801 Mapping:** vars(ieee1801_power_switch)
cpf_state_retention	boolean	Specifies the CPF state retentions. **IEEE1801 Mapping:** vars(ieee1801_state_retention)

Foundation Flow supports the following scenarios:

Using the variable cpf_file displays the results in FF/view_definition.tcl

1. Variables vars(cpf_file) + vars(cpf_timing) will set the init_cpf_file but not the init_mmmc_file. The variable init_design will be executed which means that the timing will be derived from CPF.

read_power_intent –cpf will be executed.

commit_power_intent will be executed.

The tcl file view_definition.tcl will be created to define RC Corners and to update the Delay Corners.

2. Variables vars(cpf_file) + !vars(cpf_timing) will not set the init_cpf_file but will set the init_mmmc_file. The variable init_design will be executed which means that the timing will be

derived from `mmmc_file`.

`read_power_intent -cpf` will be executed.

`commit_power_intent` will be executed.

3. No CPF file. `init_cpf_file` will not be set but `init_mmmc_file` will be set and `init_design` will be executed which means that timing will be derived from `mmmc_file`.

So, when `cpf_file` is set and `cpf_timing` is NOT defined, we default to (1) (assuming the CPF has full timing). The user must specify `vars(cpf_timing) false` to get flow (2)

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Hier ILMs

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining Optional Variables

Variable Name	Value Type	Usage Description	Command/Option Affected
`assign_buffer`	*boolean*	Adds buffers for assign statements. The default is False.	None

`buffer_name`	*list*	Specifies the list of assigned buffers.	None
`buffer_tie_assign`	*boolean*	Specifies buffer tie high/low to assign nets	None
`clock_buffer_cells`	*list*	Specifies the list of clock buffer cells.	`setCCOptMode -cts_buffer_cells` during cts step
`clock_inverter_cells`	*list*	Specifies the list of clock inverter cells.	`setCCOptMode -cts_inverter_cells` during cts step
`clock_gating_cells`	*list*	Specifies the list of clock gating cells.	`setCCOptMode -cts_clock_gating_cells` during cts step
`cts_cells`	*list*	Specifies the CTS buffer cells.	`specifyClockTree -buffer` (legacy CTS) during cts step
`delay_cells`	*list*	Specifies the list of delay cell names. This variable ensures that these cells are set to 'setDontUse' false during hold fixing.	`setDontUse <cell> <false>` during hold fixing steps
`dont_use_list`	*list*	Specifies the list of dont use cells	This file is sourced during the init step, so it should contain `setDontuse` commands.
`dont_use_file`	*file*	Specifies the file of `setDontUse` commands created by the user.	`setDontUse <cell> <true>` during the init step
`filler_cells`	*list*	Specifies the list of filler cell names.	`setFillerMode -core` during route step
`gds_files`	list	Specifies the GDS file name	`streamOut -merge` during the signoff step
`gds_layer_map`	*file*	Specifies GDSII to LEF layer mapping file	None

`jtag_cells`	*list*	Specifies the list of JTAG instances.	`specifyJtag -inst` during place step
`jtag_rows`	*integer*	Specifies the number of rows for JTAG placement.	`specifyJtag -inst` during place step
`size_only_file`	*list*	Specify instances that can be re-sized only during optimization.	`setOptMode -sizeOnlyFile <file>` during the placement step, when `place_opt_design` is enabled. Otherwise, it is during the `prects` step.
`spare_cells`	*list*	Specifies the list of spare cells instances.	`specifySpareGate -inst` during place step
`tie_cells`	*list*	Specifies the list of tie cells [tiehi tielo]	`setTieHiLoMode -cells` during the placement step
`time_design_options`	*list*	Specifies the list of following additional valid `timeDesign` options: `vars(time_design_options,setup)` and `vars(time_design_options,hold)`.	`timeDesign`
`use_list`	*list*	Specifies the list of cells to use regardless of whether they are set *dont use* in the library	`setDontUse <cell> <false>` during the init step

Examples

The variable vars (power_analysis_view) is the analysis_view that is used for power optimization. It is one of the defined views in the setup.tcl. It is used in the init step:

```
set_power_analysis_mode -analysis_view $vars(power_analysis_view)
```

Following is an example where report power is called at the end of each step and when vars (report_power) is true:

```
report_power -view $vars(power_analysis_view) -outfile
$vars(rpt_dir)/$vars(step).power.rpt
```

References

For other `edi_config.tcl` variables, see:

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Hier ILMs

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For more examples, see Example Settings for Each Script

valid timeDesign options

Defining Tie and Filler Cells

Variable Name	Value Type	Usage Description
`tie_cells`	*list*	Specify the list of tie cells
`filler_cells`	*list*	Specify the list of filler cells
`tie_cells, max_distance`	*vars*	Specify the distance between tie-cell and tie-pins less than given value. A float value (microns).
`tie_cells, max_fanout`	*vars*	Specify the number of tie-pins a tie-net can drive. Zero means no-limit (integer)

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Optional Variables

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Hier ILMs

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining Welltap and Endcap Specifications

The max_gap and cell_interval parameters are mutually exclusive, the user has to define any one of these parameters to add welltap cells.

Variable Name	Value Type	Usage Description
`welltaps`	*name*	Specify the well tap cells
`welltaps,`checkerboard	*boolean*	Specify *true or false in* the checkerboard in well tap cells.
`welltaps,`max_gap	*float*	Specify the maximum distance from the right edge of one well-tap cell to the left edge of the following well-tap cell in the same row
`welltaps,`cell_interval	*float*	Specify the maximum distance from the center of one well-tap cell to the center of the following well-tap cell in the same row

welltaps,verify_rule	*float*	Specify the verify rules between welltap cells
pre_endcap	*cell_name*	Specify the Pre Endcap cells
post_endcap	*cell_name*	Specify the Post Endcap cells

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Hier ILMs

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining Welltap and Endcap Specifications (Per power domain basis)

Defining Welltap and Endcap Specifications (Per power domain basis)

Note: This is optional.

Variable Name	Value Type	Usage Description

domain,welltaps	*name*	Specify the well tap cells in the domain
domain,checkerboard	*boolean*	Specify *true or false in* the checkerboard in well tap cells
domain,max_gap	*float*	Specify the maximum distance from the right edge of one well-tap cell to the left edge of the following well-tap cell in the same row
domain,cell_interval	*float*	Specify the maximum distance from the center of one well-tap cell to the center of the following well-tap cell in the same row
domain,pre_endcap	*cell_name*	Specify the Pre Endcap cells
domain,post_endcap	*cell_name*	Specify the Post Endcap cells

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Reporting Power

Defining Hier ILMs

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Reporting Power

Reporting Power

Variable Name	Value Type	Usage Description
activity_file	*string*	Specify the name of the activity file
activity_file_format	*enum*	Specify the format of the activity file TCF | SAF | VCD
report_power	*boolean*	Choose True or False depending upon whether you want power reported
power_analysis_view	*list*	Specifies the default setup view

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Defining Hier ILMs

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining Hier ILMs

Variable Name	Value Type	Usage Description
ilm_non_sdc_file	*ilm*	Allows support for interactive Non-SDC constraint support in ILMs
ilm_list	ilm	Specifies ILMs for bottom up hierarchical design
lef_file	*ilm*	Specifies ILMs for bottom up hierarchical design
ilm_dir	*ilm*	Specifies ILMs for bottom up hierarchical design
pre_cts_ilm_sdc	mode	Specifies ILMs for bottom up hierarchical design

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Flow Control

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining Flow Control

These variables are optional and are specific to the design.

Variable Name	Value Type	Usage Description
`activity_file`	*string*	Specify the name of the activity file
`activity_file_format`	*enum*	Select the format of the activity type TCF | SAF | VCD
`report_power`	*boolean*	Choose True or False depending upon whether you want power reported
`power_analysis_view`	*list*	Specifies the default setup view
`enable_ocv`	*enum*	Specify when to enable OCV (list of enum values). The options are: {false | pre_postcts | pre_postroute | pre_postroute_si | pre_prects | pre_signoff}
`enable_cppr`	*enum*	Specify during enable CPPR. The options are: {setup, hold, both, none}
`enable_ss`	*enum*	Specify if/when to enable signalStorm delay calculation. The options are: {pre_place | pre_prects | pre_postcts, | pre_postroute | pre_postroute_si | pre_signoff | false}
`enable_si_aware`	boolean	Specify whether to enable si-aware postroute optimization Note: This will disable the postroute_si and postroute_si_hold steps as the are unnecessary when this is enabled)
`fix_hold`	enum	Controls the points of the flow where hold optimization is enabled. The options are: false | postcts | postroute | postroute_si
`catch_errors`	*string*	Specify to save an immediate database on errors in catch
`save_on_catch`	*string*	Saves the Innovus database when the catch is activated because of an error

antenna_diode	*string*	Cell for antenna fixing
save_constraints	*string*	Save constraints along with design database
abort	boolean	Abort the codegen if an error is found. The default is True.
report_run_time	boolean	Specify to track and report CPU time for each step
useful_skew	*boolean*	Specify to enable useful skew optimization
postroute_spread_wires	*boolean*	Enable postroute wire spreading
signoff_extraction_effort	*string*	Specify the signoff extraction effort
postroute_extraction_effort	enum	Specify level of extraction effort. The default is medium. This variable is a flow control variable related to the type of extraction post route.
hier_flow_type		Enables the new two pass hierarchical flow. The valid types are 1pass (default) and 2pass
enable_aocv		Enables the new AOCV timing feature setAnalysisMode -aocv
rc_corner,atf_file		A CCOpt technology file (atf) needed for the new command ccopt_design
delay_corner,power_domains		Defines the binding between delay corner and power domains. It is used with the vars(cpf_timing) false flow.
absolute_lib_path		Saves absolute paths to library files and is an option to saveDesign
relative_path		Saves relative paths to data files and is is an option to saveDesign
cts_engine		Switches on ccopt features. The valid values are [*cts, ccopt,* and *ccopt_cts*]

For other edi_config.tcl variables, see

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Hier ILMs

Defining Command Mode Variables

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining Command Mode Variables

In this section we provide a list of recommendations for the Command Mode variables. The placement variables assume that all I/Os are placed in legal locations, as they are not moved by default.
The set optimization options should be persistent throughout the flow. Preserve the assertions on hierarchical ports (enable for hierarchical flows so that module ports that are constrained are not replicated).

Note: Values in italic text are recommended but are not default values.

- For congested designs, set the congestion effort to high instead of to medium.

Variable Name	Default
place_io_pins	*false*
clock_gate_aware	*true*
congestion_effort	*auto*
in_place_opt	*false*
no_pre_place_opt	*false*

place_opt_design	*true*

- For timing effort:

Variable Name	Default
high_timing_effort	*false*

- In hierarchical flow, the final full flat timing analysis is done with the original constraints and it does not take into account the block level constraint files (that may have been modified). Use `-preserveAssertions` {true | false} to avoid issues related to buffer removal. It may be beneficial to run the first pass of hold fixing with setup degradation disallowed to see how many hold violations are left, and then check if the violations are real and decide how to proceed.

- To enable leakage power optimization throughout the flow, use `-leakagePowerEffort` {`low` | `high`}. An high effort may impact timing and will have an impact on run-time.

- For pre-route optimization, set critical range to the percentage of sub-critical paths to be optimized, for example, 0.2 or 0.5.

Variable Name	Default
preserve_assertions	false
leakage_power_effort	none
dynamic_power_effort	none
fix_hold	true
fix_hold_ignore_ios	false
fix_hold_allow_tns_degradation	true

The high effort hold fixing should improve the quality of results at the expense of runtime.

- Set CTS options; the recommendation is to route the clock nets during CTS. If some signals should not be routed, set the clock nets option to false and use `RouteClkNet` true in the spec file.

Variable Name	Default	Usage Description
route_clock_nets	*true*	Route all clock nets during CTS
clock_eco	*false*	Run ckECO [postcts, postroute, both, none]

`clock_gate_clone`	*false*	Choose True or False depending upon whether you want to clone the gates

- Set the routing options. The recommended options for multicut via insertion and litho driven routing incur run-time penalties but yield advantages so they are included here. The `-routeWithLithoDriven` option is recommended for 45 nm but may also be used at 65 nm.

  The `-drouteUseMultiCutViaEffort` option can be set to high if needed; expect an average of 5 percent coverage improvement. Expect an average 10 percent run-time degradation as compared to the medium effort. Setting the value to high also affects TNS. Consider disabling this option during early trials.

  The flow default is to disable multi-cut via insertion. For 90nm and below, it is recommended to insert multi-cut vias using `set vars(multi_cut_effort)` medium. This will come with a runtime and memory penalty and will also affect QoR. The `set vars(multi_cut_effort)` can be set to high if needed; expect an average of 5 percent coverage improvement with an average 10 percent runtime hit versus medium and will also affect TNS.

Variable Name	Default	Usage Description
`multi_cut_effort`	*low*	Insert multi-cut vias during routing
`litho_driven_routing`	*false*	Litho-driven routing option
`in_route_opt`	*false*	Performs timing optimization after track assignment

- Set the noise analysis options. Existing CeltIC users should preserve existing settings if applicable. Consult the SI Analysis application note for more information on these settings. You can get the application note from your Cadence Support Representative.

Variable Name	Default	Usage Description
`delta_delay_threshold`	*undefined*	Specify the delay noise threshold
`celtic_settings`	*undefined*	Specify the Celtic settings
`total_c_thresh`	*undefined*	Specify the total cap threshold filter
`relative_c_thresh`	*undefined*	Specify the relative threshold filter
`coupling_c_thresh`	*undefined*	Specify the coupling threshold filter

si_analysis_type	*enum*	Specify the analysis type (default or pessimistic)
acceptable_wns	float	Specify to define the target slack for SI optimization

For other `edi_config.tcl` variables, see

Defining Extraction Options

Defining Noise Settings

Defining Power and Ground Nets

Defining Multi-Threading or Distributed Processing

Defining LP Flow Variables

Defining Optional Variables

Defining Tie and Filler Cells

Defining Welltap and Endcap Specifications

Defining Welltap and Endcap Specifications (Per power domain basis)

Reporting Power

Defining Hier ILMs

Defining Flow Control

For `setup.tcl` variables see Describing setup.tcl

For examples see Example Settings for Each Script

Defining ccopt_design for Clock Tree Construction

The following vars are used for defining ccopt_design for Clock Tree Construction.

Variable Name	Value Type	Usage Description
cts_inverter_cells	list	Specify the CTS inverter cells
cts_buffer_cells	list	Specify the CTS buffer cells
clock_gate_cells	list	Specify the clock gate cells
cts_use_inverters	boolean	Specify true or false

update_io_latency	boolean	Specify true or false
cts_target_skew	float	Specify the skew
cts_target_slew	float	Specify the slew
cts_io_opt	enum	Specify on | off | secondary
ccopt_effort	enum	Specify low | medium | high

Defining CCOpt Top and Bottom Layers

The following vars define top and bottom layers for clock tree and leaf nets, non default rules, and clock shielding net (ccopt only).

Variable Name	Value Type	Usage	Description
clk_tree_top_layer	string	setCTSMode | set_ccopt_mode	Specifies the preferred top routing layer for non-leaf clock nets.
clk_tree_bottom_layer	string	setCTSMode | set_ccopt_mode	Specifies the preferred bottom routing layer for non-leaf clock nets.
clk_leaf_top_layer	string	setCTSMode | set_ccopt_mode	Specifies the preferred top routing layer for leaf clock nets.
clk_leaf_bottom_layer	string	setCTSMode | set_ccopt_mode	Specifies the preferred bottom routing layer for leaf clock nets.
clk_tree_ndr	string	setCTSMode | set_ccopt_mode	Specifies the non default rule for non-leaf clock nets.
clk_leaf_ndr	string	setCTSMode | set_ccopt_mode	Specifies the non default rule for leaf clock nets.
clk_tree_shield_net	string	setCTSMode | set_ccopt_mode	Specifies the global net to use for shielding the clock network.

Note: Leaf nets are the nets connected to the clock pins of the leaf cell (flop/latches). Non-leaf nets are the nets that are not leaf nets; and they form the main part of the clock tree from the root down till the leaf nets.

Example Settings for Each Script

Each step of the script has settings as shown in the example below which tells the user which variables affect the script generation.

- **Place**

    ```
    set vars(step) place
    ```

    These variables affect this step:

    - – vars(congestion_effort)

    - – vars(clock_gate_aware)

    - – vars(place_io_pins)

    - – vars(in_place_opt)

    - – vars(preserve_assertions)

    - – vars(leakage_power_effort)

    - – vars(dynamic_power_effort)

    - – vars(clock_gate_aware)

    - – vars(critical_range)

- **PreCTS**

    ```
    set vars(step) prects
    ```

    These variables affect this step:

    - – vars(process)

    - – vars(preserve_assertions)

    - – vars(leakage_power_effort)

    - – vars(dynamic_power_effort)

    - – vars(clock_gate_aware)

    - – vars(critical_range)

    - – vars(useful_skew)

    - – vars(skew_buffers)

- **CTS**

    ```
    set vars(step) cts
    ```

These variables affect this step:

- – vars(process)

- – vars(route_clock_nets)

- – vars(litho_driven_routing)

- – vars(multi_cut_effort)

- **PostCTS**

  set vars(step) postcts

  These variables affect this step:

  - – vars(process)

  - – vars(enable_cppr)

  - – vars(clock_gate_clone)

    See Sample Single Script Flow for an example implementation.

3

Plug-in Variables and Tags

Tags and plug-ins provide an easy method to both expand upon and customise the basic flows. In this section we describe the pre-defined variables for plug-ins and tags. The section covers:

- Defining Plug-in Variables

- Commands that can be Tagged

Defining Plug-in Variables

Plug-in variables are included to allow you to customize the flow. For each step in the flow, there are plug-ins before and after the main step command. For example, in the placement step, there is a plug-in before and after the `place_opt_design` command. The plug-ins are called `pre_`*step*`_tcl` and `post_`*step*`_tcl`.

So, in the case of the placement step, the plug-in would be called `vars(pre_place_tcl)` and *vars(post_place_tcl)*.
These variables point to a user generated TCL file containing Innovus commands which are run in addition to the basic flow commands. The following plug-ins are supported.

Keep in mind that there is no error checking but errors that occur will be caught and the tool will exit gracefully and save databases, where appropriate.

Note: The plug-in script should not include the `restoreDesign` command. The foundation flow automatically restores the design from the previous step.

Plug-in Variable Name	Value Type	Usage Description
`always_source_tcl`	*string*	The contents of the file will be sourced at the start of each stage of the flow
`pre_init_tcl`	*string*	The content of the file will be sourced before loading the design

post_init_tcl	*string*	The content of the file will be sourced after loading the design
pre_place_tcl	*string*	The content of the file will be sourced after loading the design but before running the placement step. Before running the command `place_opt_design`
place_tcl	*string*	When specified, this file will replace the default placement commands
post_place_tcl	*string*	The content of the file will be sourced after running the placement step but before exiting the script. After running the command `place_opt_design`
pre_prects_tcl	*string*	The content of the file will be sourced before loading the CTS
post_prects_tcl	*string*	The content of the file will be sourced after pre-CTS optimization.
cts_tcl	*string*	When specified, this file will replace the default `ccopt_design` command
pre_cts_tcl	*string*	The content of the file will be sourced after placing the design but before running designing the clock. Before running the command `ccopt_design`
post_cts_tcl	*string*	The content of the file will be sourced after designing the clock. After running the command `ccopt_design` Specify the directory `PLUG/post_cts.tcl`
pre_postcts_tcl	*string*	The content of the file will be sourced before post-CTS optimization
post_postcts_tcl	*string*	The content of the file will be sourced after post-CTS optimization
pre_postcts_hold_tcl	*string*	The content of the file will be sourced before post-CTS Hold Fixing
post_postcts_hold_tcl	*string*	The content of the file will be sourced after before post-CTS Hold Fixing

`pre_route_tcl`	*string*	The content of the file will be sourced after loading the CTS but before routing. Before running the command `routeDesign`
`post_route_tcl`	*string*	The content of the file will be sourced after routing. After running the command `routeDesign`
`pre_postroute_tcl`	*string*	The content of the file will be sourced before `postRoute` Hold Fixing
`post_postroute_tcl`	*string*	The content of the file will be sourced after `postRoute` Hold Fixing
`postroute_spread_wires`	boolean	The content of the file will be sourced after loading the post routing design and spreading the wires
`pre_signoff_tcl`	*string*	The content of the file will be sourced before final static timing analysis (STA)
`post_signoff_tcl`	*string*	The content of the file will be sourced after before final static timing analysis (STA)
`metal_fill_tcl`	*string*	An example is provided in the TEMPLATES directory
`final_always_source_tcl`	*string*	Specify the path to plug-in file. Runs after every flow.
`metalfill_tcl`	*string*	When specified, this file will replace the default metal fill commands
`metalfill`	*string*	Specify variable to define if and when to insertion `metalfill` (`pre_postroute`, `pre_postroute_si`, `pre_signoff`)
`pre_model_gen_tcl`	*string*	The content of the file will be sourced before running `FlexModel` model generation
`pre_assign_pin_tcl`	*string*	The content of the file will sourced before running partition pin assignment

Plug-in Usage:

- To enable a plug-in, define the appropriate plug-in variable in the setup.tcl file. The software sources the plug-in automatically at the appropriate time in the flow. For example:

  ```
  set vars(post_cts_tcl) PLUG/post_cts.tcl
  ```

- To use the `post_cts_tcl` plug-in to adjust clock latencies after CTS.

Commands that can be Tagged

Below is a list of commands that can be tagged with the following tags:

- `-pre_tcl` *filename* => Script to be run before the tagged command

- `-post_tcl` *filename* => Script to be run after the tagged command

- `-replace_tcl` *filename* => Script to be run to replace the tagged command

- `-skip` *true | false* => Skip the tagged command

These variables are structure *step,commmand,tag*
They should be added to your `setup.tcl` or `edi_config.tcl`

See Also,

Tags for Innovus Flow

Tags for Innovus Flow

Tags provides a more granular approach to flow customization for customers who need it. Each step of the flow has a list of tags and they are generated by `gen_setup.tcl` in a file called `edi_tags.auto.tcl`. Each command has four tags that you need to define. The types of tags are:

- A variable to skip the command entirely

- A script to completely replace the command

- Scripts to run before the command

- Scripts to run after the command

The following list lists the tags for the Innovus foundation flow Tags for Flow

This section also contains these examples:

- Example 1:

- Example 2:

- Example 3:

Example 1:

```
set vars(prects,opt_design,post_tcl) file
```

```
set var(flat) full
```

Where *file* contains:

```
 saveDesign $vars(dbs_dir)/prects1.enc


 setClockDomains -fromType register -toType register


 optDesign -preCTS incr


 ...


 optDesign -preCTS -outDir RPT -prefix prects


 # <begin tag prects,opt_design,post_tcl>


 saveDesign $vars(dbs_dir)/prects1.enc


 setClockDomains -fromType register -toType register


 optDesign -preCTS incr


 # <end tag prects,opt_design,post_tcl>


 ...
```

By setting, `vars(flat)` to `full`, the contents of the tag script get imported into the run script to create a fully flattened script. Otherwise, there would simply be source *file*.

Example 2:

By default, `time_design` for place, cts, and route are skipped. If you want to enable timing reports for any of them, just set `vars(step,time_design,skip) true`.

```
set vars(cts,time_design,skip)       true

...

setCTSMode -routeClkNet true

setNanoRouteMode -routeWithLithoDriven true

createClockTreeSpec

specifyClockTree -update "AutoCTSRootPin * RouteClkNet YES"

ccopt_design -skipTimeDesign -outDir RPT

...

# <begin tag cts,time_design,skip>

# timeDesign -postcts -prefix cts -outDir RPT

# <end tag cts,time_design,skip>
```

Example 3:

Steps to create directories inside the RPT directory where reports can be appended while running the flow:

Create a `final_always_source_tcl` plug-in:

`set vars(final_always_source_tcl) file`

Where file:

Creates the directory you want.

For Example:

```
file mkdir $vars(rpt_dir)/$vars(step)
```

Moves all relevant reports into this directory

For Example:

```
foreach file glob $vars(rpt_dir) $vars step*
file move $file $vars(rpt_dir) $vars(step)
```

Add a final timeDesign --reportOnly

For Example:

```
timeDesign --reportOnly --prefix $vars(step) --outDir $vars(rpt_dir)/$vars(step)
```

This has the advantage of being able to easily change the prefix as well.

See also, Plug-in Variables and Tags

4

Using the Wizard

Innovus provides a foundation flow wizard and code generator to enable you to quickly generate scripts to load design data, setup timing environment, and execute the recommended implementation flow from Placement through Signoff.

The Innovus wizard allows you to define all information necessary to generate a setup.tcl file to drive the foundation flows. This GUI walks the customer through library definition in such a way the required information can be obtained while hiding the underlying complexities of the MMMC timing infrastructure. It also provides the user all the necessary foundation flow variable settings. The user can also re-enter the wizard and restore the current settings.

Note: You can supplement the generated setup.tcl with additional variables from the documentation or TEMPLATE directory. The wizard does not split the variables between `setup.tcl` in `edi_config.tcl` or support the Low Power and Hierarchical flows

The Wizard can be enabled from the last tab on the tool menu.

Note: All *fields in the wizard are mandatory. All fields without * are optional. The Wizard facilitates your enteries and shows you a summary at the end of each stage. The files you need for the wizard to run are placed in the folder `../FF` and its sub folders. The system adds the variables you do not enter.

In this section we discuss the following:

In this section, we discuss:

- Starting the Flow Wizard

- View and Save Scripts

- Setting Up the Library for the Flow Wizard

- Designing the Flow

- Timing the Flow

- Adding Power to the Flow

- Adding Power to the Flow - Summary

- Setting up the Tool

- Setting up Plug-in Scripts

- Completing Setup

Starting the Flow Wizard

Change to the directory `ff_workshop/wizard` and enter this command to start the software:

```
Innovus
```

This command will start the Innovus Implementation System in the graphical mode.

Choose *Flow - Foundation Flow Wizard*.

Note: The wizard allows you to start from scratch, from the current design in memory, or from an existing `setup.tcl`. In this section we will discuss starting from scratch.

Welcome to Wizard - Basic Fields and Options

How Do You want to Start

Start from Scratch	Choose this option to automatically configure the Wizard. You will still need to select files and enter names of files but the background process will take care of most of the configurations. In this section we will discuss starting from scratch.
Load the Design Setup from Memory	Choose this option if you have already created a setup.tcl file or the system has it in memory. Click *Load*.
Load Previously Installed Script	Click the icon on the field to select a file from the system and to modify it as you go along creating the configurations to save in setup.tcl.

Foundation Flow Install Directory	Select the install directory by browsing the system and click *Install*.
Save Foundation Flow Database At	Select where you want to save the database.
Save Foundation Flow Reports At	Select where you want to save the database.
Continue	Click to move to next screen

For other Wizard forms, click link below:

Setting Up the Library for the Flow Wizard

Designing the Flow

Timing the Flow

Adding Power to the Flow
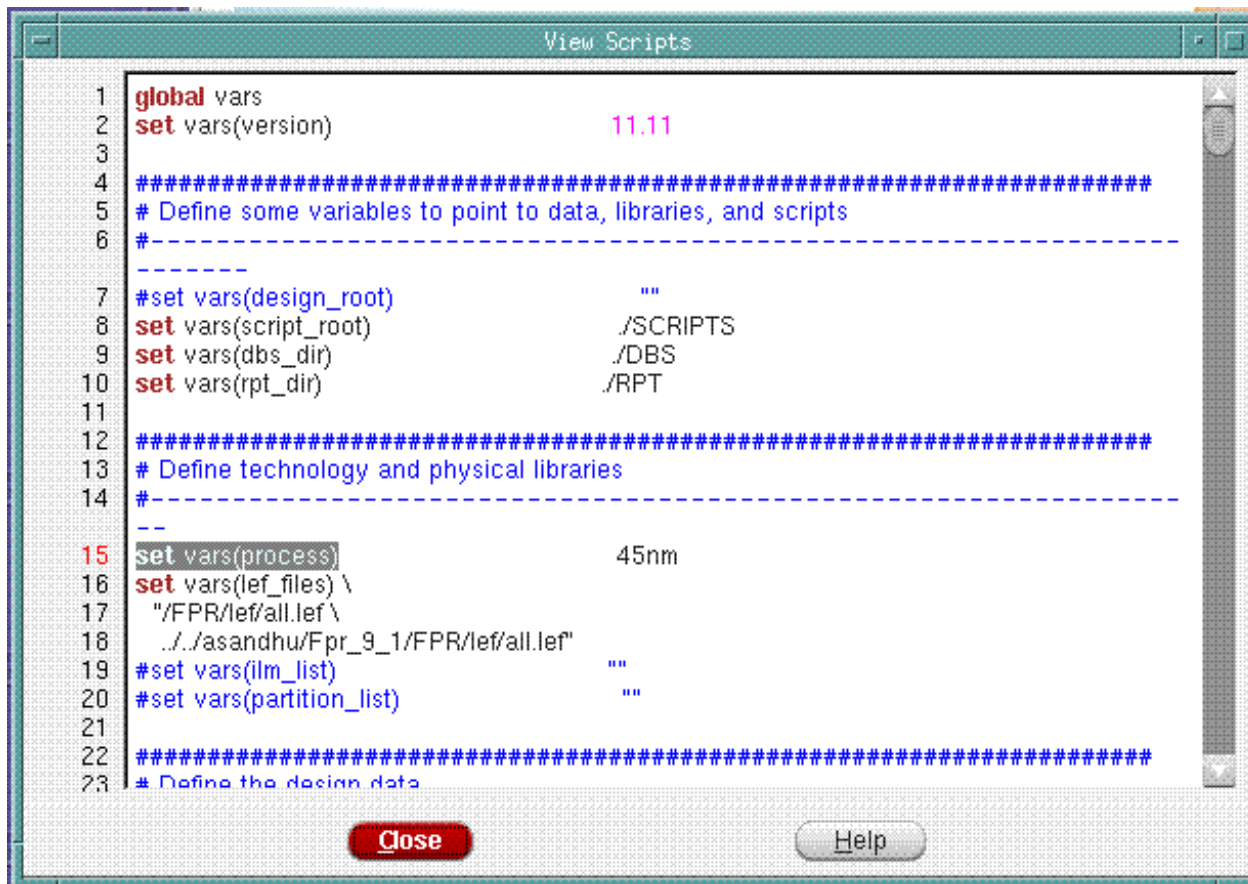
Setting up the Tool

Setting up Plug-in Scripts

Completing Setup

View and Save Scripts

As you complete your setup you can now View your Foundation Flow scripts. Click the View button on top right to view the script:

You can also save the script by clicking the Save button top right. Choose where you want to save the different scripts: Setup, Innovus Configure and LP Configure.
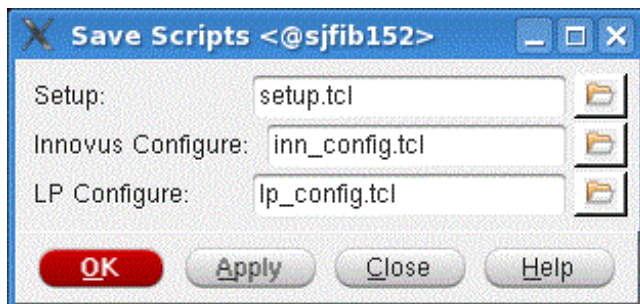
Choose the folder and click Apply.

Setting Up the Library for the Flow Wizard

In this section we cover:

Setting Up the Library Information

Adding Library to the Flow - Summary

For other Wizard forms, click link below:

Starting the Flow Wizard

Designing the Flow

Timing the Flow

Adding Power to the Flow

Setting up the Tool

Setting up Plug-in Scripts

Completing Setup

Adding Library to the Flow - Summary

The Library summary display.
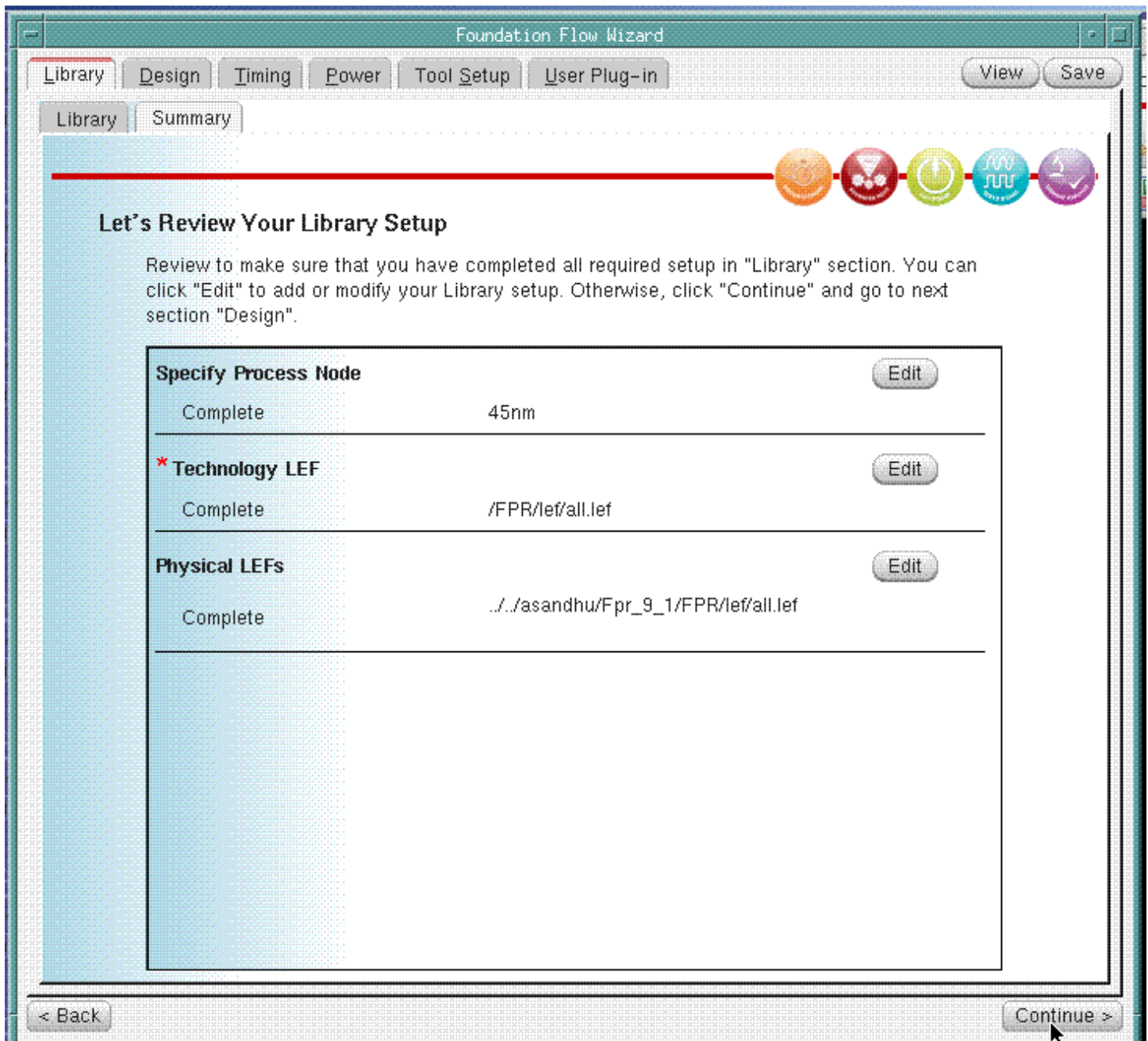
Setup Library Summary - Basic Fields and Options

Specify Process Node	Displays the type of node you specified. Click *Edit* to go back to the *Library* form and change value.
Technology LEF	Displays the LEF file you specified. Click *Edit* to go back to the *Library* form and change value.

Physical LEF	Displays the LEF file you specified. Click *Edit* to go back to the *Library* form and change value.

See also,

Setting Up the Library Information

Setting Up the Library Information

The following screen appears:

Setting Up the Wizard - Basic Fields and Options

Setting Up the Process Node of Your Design	Sets up the process node. This is pre-populated, change it to the value you want.
Specify the Technology LEF File for your Design	Select the LEF file directory by browsing the system.
Use the File Browser to select Additional LEF Libraries for your File	Select additional LEF libraries, if any. See Designing the Flow - Netlist
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Adding Library to the Flow - Summary

Designing the Flow

In this section we cover the following

- Adding Design to the Flow - Summary

- Designing the Flow - Clock Tree

- Designing the Flow - Netlist

For other Wizard forms, click link below:

- Starting the Flow Wizard

- Setting Up the Library for the Flow Wizard

- Timing the Flow

- Adding Power to the Flow

- Setting up the Tool

- Setting up Plug-in Scripts

- Completing Setup

# Adding Design to the Flow - Summary

The Design summary display.
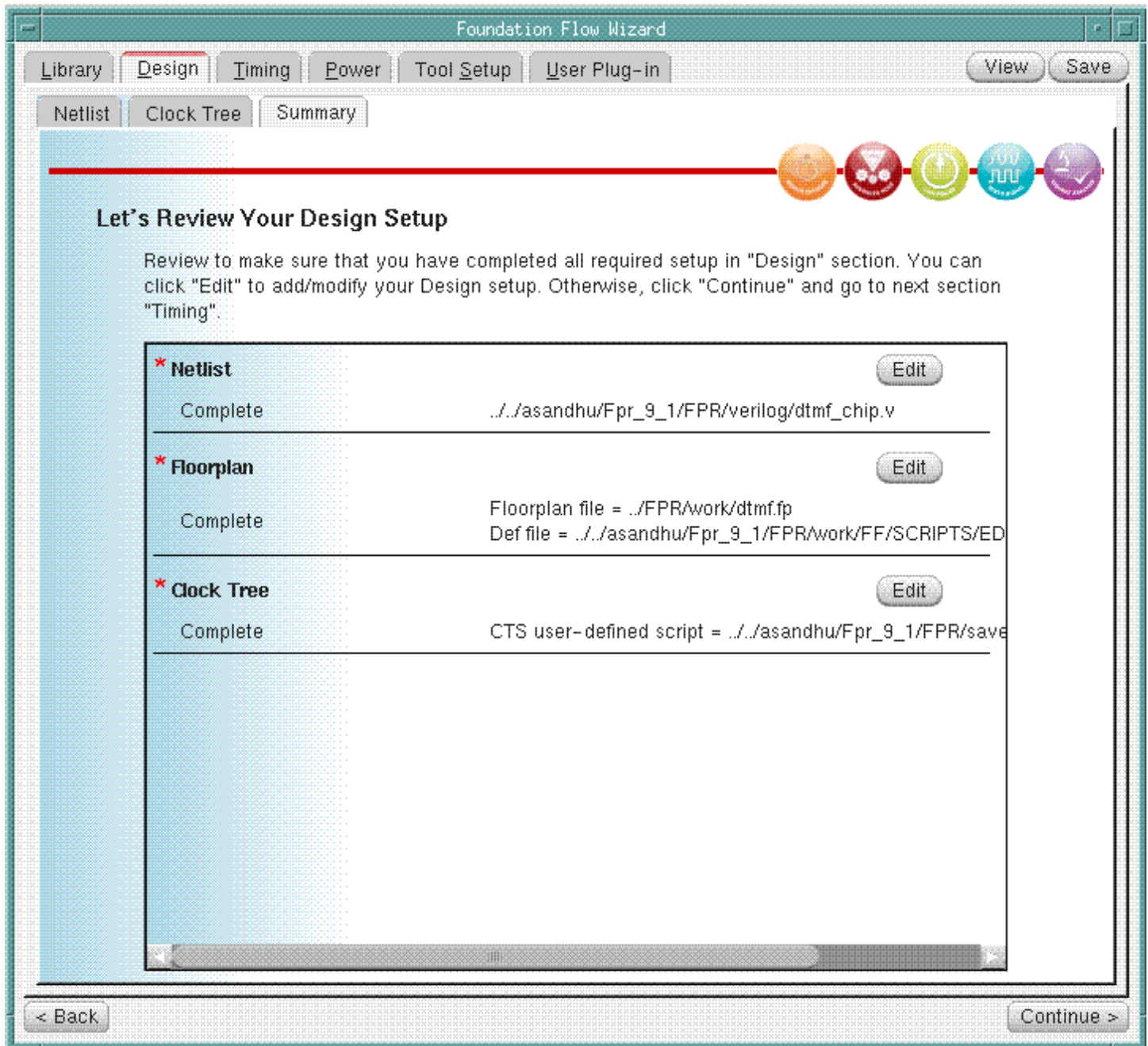
Setup Design Summary - Basic Fields and Options

Netlist	Displays the netlist you specified. Click *Edit* to go back to the *Library* form and change value.	
Floorplan	Displays the floorplan you specified. Click *Edit* to go back to the *Library* form and change value.	
Clock Tree	Displays the clock tree you specified. Click *Edit* to go back to the *Library* form and change value.	

See also,

Designing the Flow - Netlist

Designing the Flow - Clock Tree

Designing the Flow - Clock Tree

The following screen appears:

Setup Your Clock Tree Synthesis Constraints

In this section, you can specify your own Clock Tree Specification file or allow to let Innovus System generate one for you automatically.

*How do you want to generate your clock tree(s)?

- ● Automatically by Innovus using the following clock cells: [Select]
- ○ Use my clock tree specification file(s): [...]
- ○ By my user-defined script: [ ]

*Indicates mandatory fields

< Back   Continue >

Setup Your Clock Tree Synthesis Constraints - Basic Fields and Options

Automatically by Innovus Using the Following Clock Cells	Allows the wizard to generate the clock tree automatically.
Use my Clock Tree Spec File	Select to use your own clock tree specification file from the system. See Designing the Flow - Netlist.
By My User Defined Script	Browse and select your own script
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

Reviewing the Clock Tree Setup

In the next screen you can review the configurations you have entered. You can edit them or, if satisfied, click *Continue*.

See also,

Designing the Flow - Netlist

Adding Design to the Flow - Summary

Designing the Flow - Netlist

The following screen appears:

Foundation Flow Wizard

Library | Design | Timing | Power | Tool Setup | User Plug-in | View | Save

Netlist | Clock Tree | Summary

Specify Verilog netlist(s) and floorplan

In this section, you are required to specify the Verilog netlist(s) and floorplan for your design. Other fields are optional.

* **Choose the Verilog netlist(s) for your design**

../../asandhu/Fpr_9_1/FPR/verilog/dtmf_chip.v

...

* Design Name: test

* **Choose the floorplan file(s) for your design**

● Yes. Use the floorplan file: ../FPR/work/dtmf.fp

Use the DEF file:

../../asandhu/Fpr_9_1/FPR/work/FF/SCRIPTS/EDI/run_init.tcl

...

○ No. I will provide a script to generate the complete floorplan in User Plug-In section

Specify power and ground net name(s)

Power net name(s) : power

Ground net names(s) : net

* Indicates mandatory fields

< Back | Continue >

Setup Your Netlist and Floorplan - Basic Fields and Options

Choose the Verilog Netlist for your Design	Select the Verilog netlist by browsing the system	
	Design Name	Name of the design.
Choose the Floorplan File for your Design		
	Yes: Choose the Floorplan File	Select the Floorplan file by browsing the system and additionally, Use the DEF file
	No	Select to use a Script to Generate the Complete Floorplan in User Plug-in

Specify Power and Ground Net Name

Power Net Name	Enter value of power net name.
Ground Net Name	Enter value of power net name
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

Notes

During the course of this wizard you may need to find files from your system. When you click ... the
following type of screen will show up.
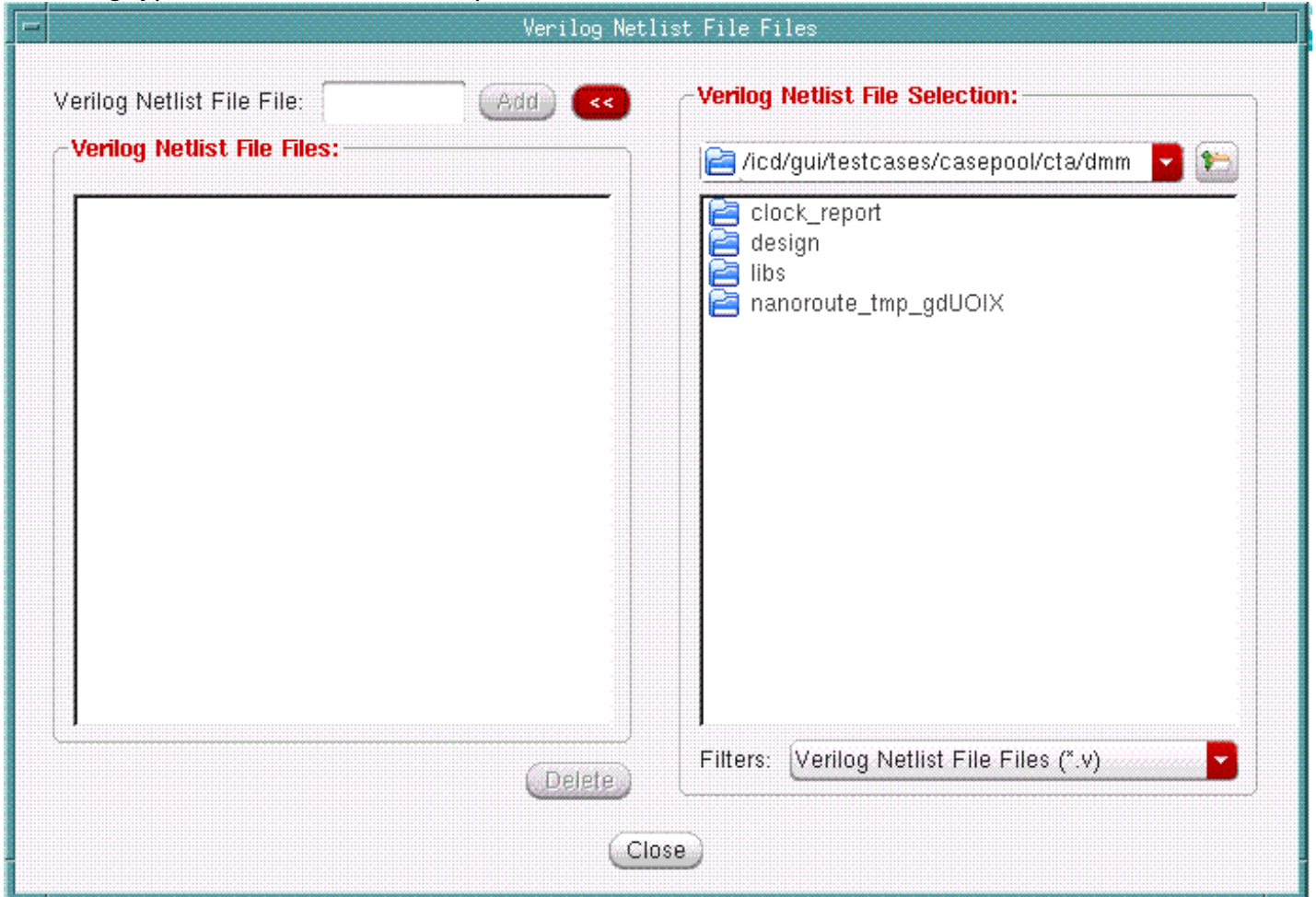
Double Pane Selection Form - Fields and Options

Right Pane	Displays the relevant files and folders in the system. You can browse and select the other directories.
Add	Select the file that you need and click *Add*. The right pane opens. Select the file and click <<. The file is added to the left pane.
Delete	Select the file that you need and click *Delete*.
Filter	Specifies a filter pattern for selecting the files.

See also,

Designing the Flow - Clock Tree

Adding Design to the Flow - Summary

Timing the Flow

In this section we cover:

- Timing the Flow - Timing

- Timing the Flow - Library Set

- Timing the Flow - RC Corner

- Timing the Flow - Delay Corner

- Timing the Flow - Constraint Mode

- Timing the Flow - Analysis View

- Reviewing the Timing Setup

For other Wizard forms, click link below:

Starting the Flow Wizard

Setting Up the Library for the Flow Wizard

Designing the Flow

Adding Power to the Flow

Setting up the Tool

Setting up Plug-in Scripts

Completing Setup

Timing the Flow - Timing

The following screen appears:

Foundation Flow Wizard

Library | Design | Timing | Power | Tool Setup | User Plug-in | View | Save

Timing | Library Set | RC Corner | Delay Corner | Constraint Mode | Analysis View | Summary

Setup Your Design for Timing-Driven Place and Route

In this section, enter all the information necessary to initialize your timing environment.

Are your timing libraries ECSM or CCS?
- Yes
- No

Do you want to enable SignalStorm delay calculation?
- Yes, enable SignalStorm DelayCalculation before: pre_place
- No

Do you want to enable OnChip Variation?
- Yes, enable OCV before: pre_postcts
- No

< Back | Continue >

Timing the Flow - Timing - Basic Fields and Options

Are you Timing Libraries ECSM or CCS Model?	Select *Yes* or *No*
Do you want to Enable SignalStorm Delay Calculation?	Select *Yes* or *No.* If you select *Yes* then choose at what stage from the drop down menu.
Do you want to enable onChipVariation?	Select *Yes* or *No.* If you select *Yes* then choose at what stage from the drop down menu.
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Timing the Flow - Library Set

Timing the Flow - RC Corner

Timing the Flow - Delay Corner

Timing the Flow - Constraint Mode

Timing the Flow - Analysis View

Reviewing the Timing Setup

Timing the Flow - Library Set

The following screen appears:
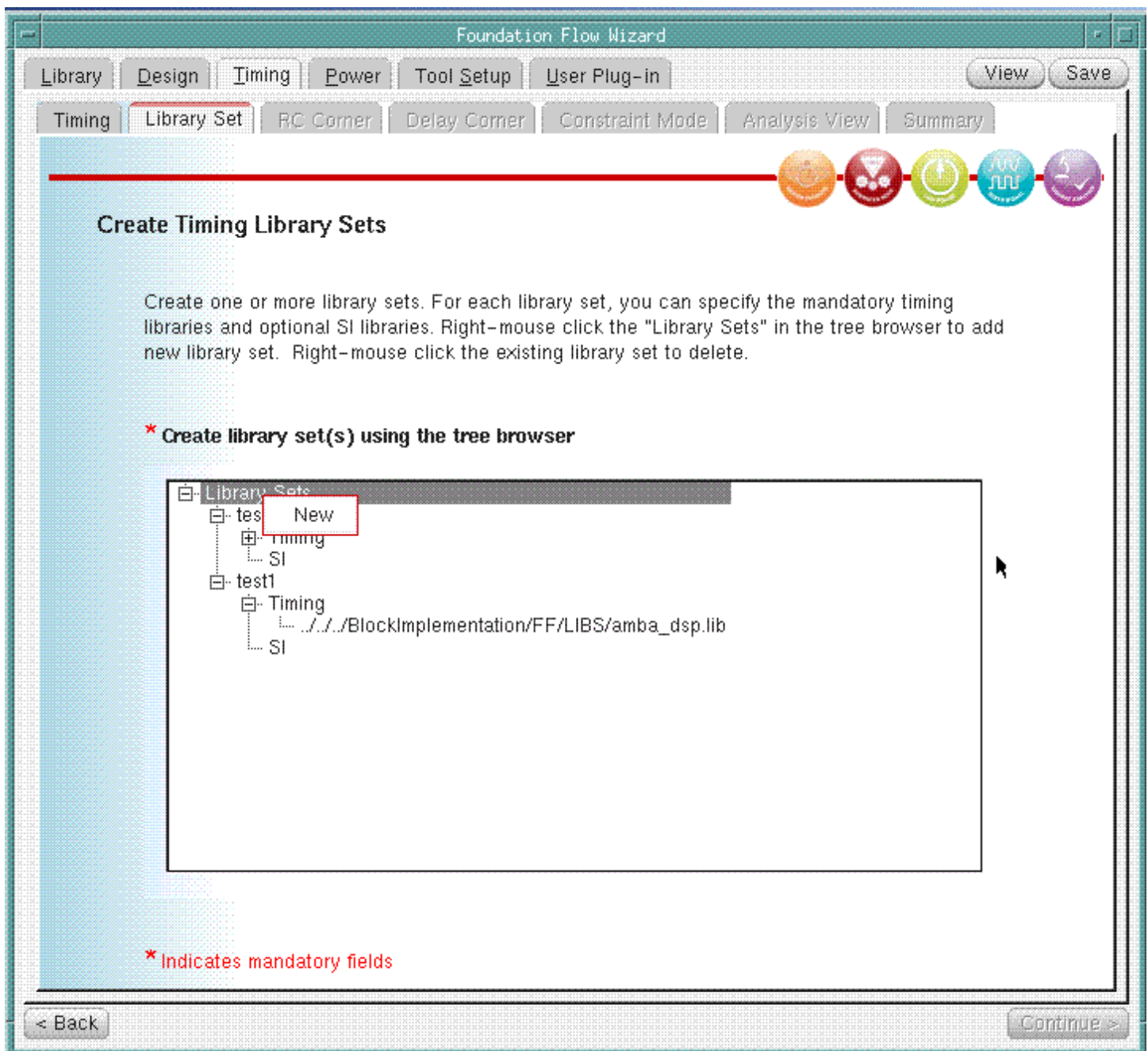
Create Timing Library Sets - Basic Fields and Options

Create Library Sets Using the Tree Browser	Use right mouse click to *Add* and once added, to *Edit* or *Delete*. The following form opens File Menu > Add Library Set. This form will take you to further forms where you make selections and then come back to the Wizard.
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Timing the Flow - Timing

Timing the Flow - RC Corner

Timing the Flow - Delay Corner

Timing the Flow - Constraint Mode

Timing the Flow - Analysis View

Reviewing the Timing Setup

Timing the Flow - RC Corner

The following screen appears:
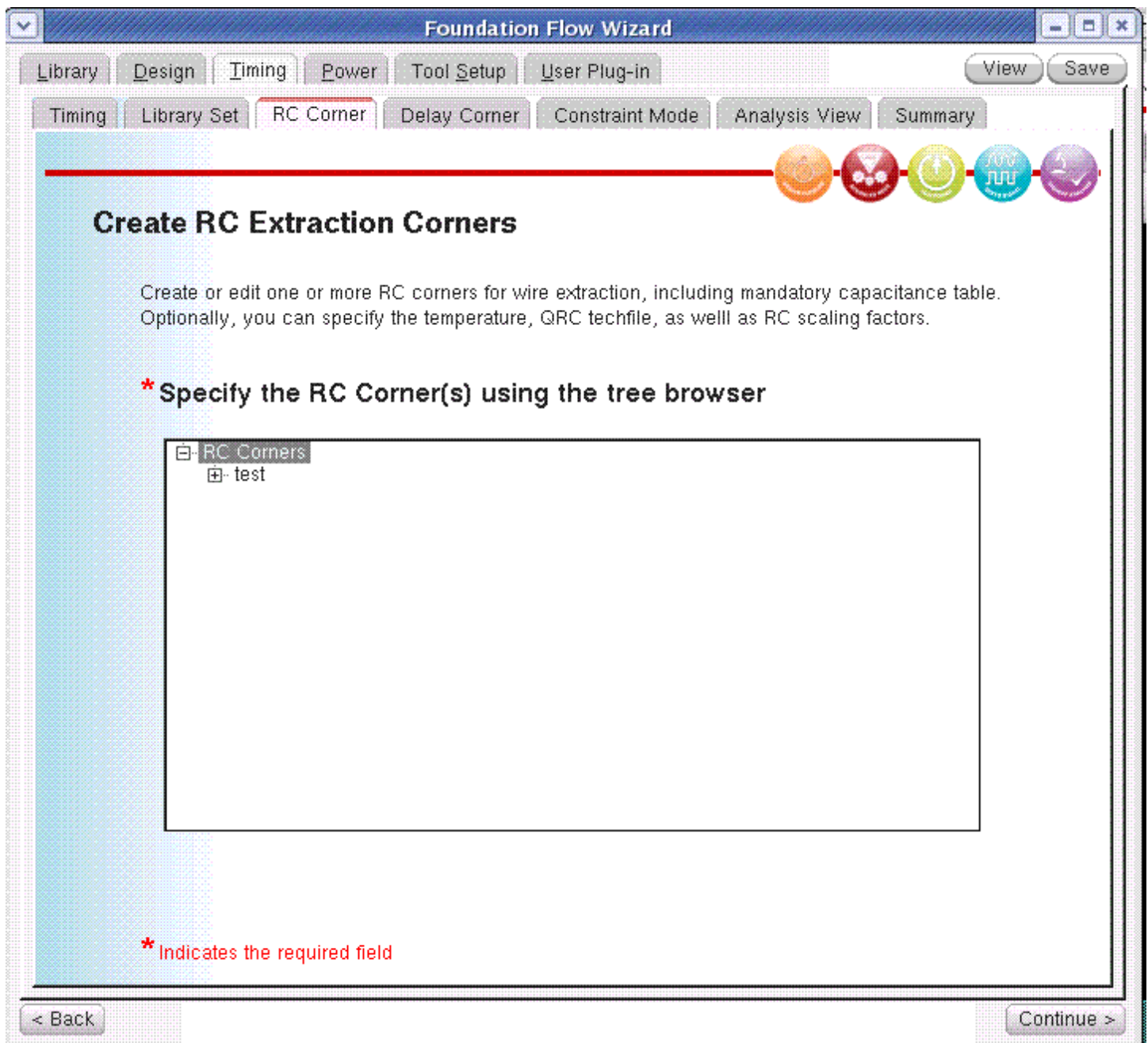
Create RC Extraction Corners - Basic Fields and Options

Specify the RC Corner Using the Tree Browser	Use right mouse click to *Add* and once added, to *Edit* or *Delete*. The following form opens File Menu > Add RC Corner. This form will take you to further forms where you make selections and then come back to the Wizard.
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Timing the Flow - Timing

Timing the Flow - Library Set

Timing the Flow - Delay Corner

Timing the Flow - Constraint Mode

Timing the Flow - Analysis View

Reviewing the Timing Setup

Timing the Flow - Delay Corner

The following screen appears:

Create Delay Corner Sets - Basic Fields and Options

Do you want to Enable Clock Path Pessimism Removal?

Enable CPPR After	Select the stage from the drop-down menu

Specify Delay Corner(s) Using the Matrix Table	Select the button to turn it *On* or *OFF.* When you click OFF the following form opens File Menu > Add Delay Corner. This form will take you to further forms where you make selections and then come back to the Wizard.
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Timing the Flow - Timing

Timing the Flow - Library Set

Timing the Flow - RC Corner

Timing the Flow - Constraint Mode

Timing the Flow - Analysis View

Reviewing the Timing Setup

Timing the Flow - Constraint Mode

The following screen appears:

Create Constraint Modes

This is to associate a list of SDC constraint files with a specified constraint mode name. This constraint mode name can be referred to later when creating analysis view. Right-mouse click on the "Constraint Modes" in the tree browser to add new constraint mode. Right-mouse click on the existing constraint mode to delete.

*Display and edit the constraint modes using the tree browser

- Constraint Modes
  - test
  - test2

*Indicates the required field

< Back   Continue >

Create Constraint Modes - Basic Fields and Options

Display and Edit the Constraint Modes Using the Tree Browser	Use mouse right-click Add Constraint Mode and the form will take you to File Menu > Add Constraint Mode where you make selections and then come back to the Wizard.
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Timing the Flow - Timing

Timing the Flow - Library Set

Timing the Flow - RC Corner
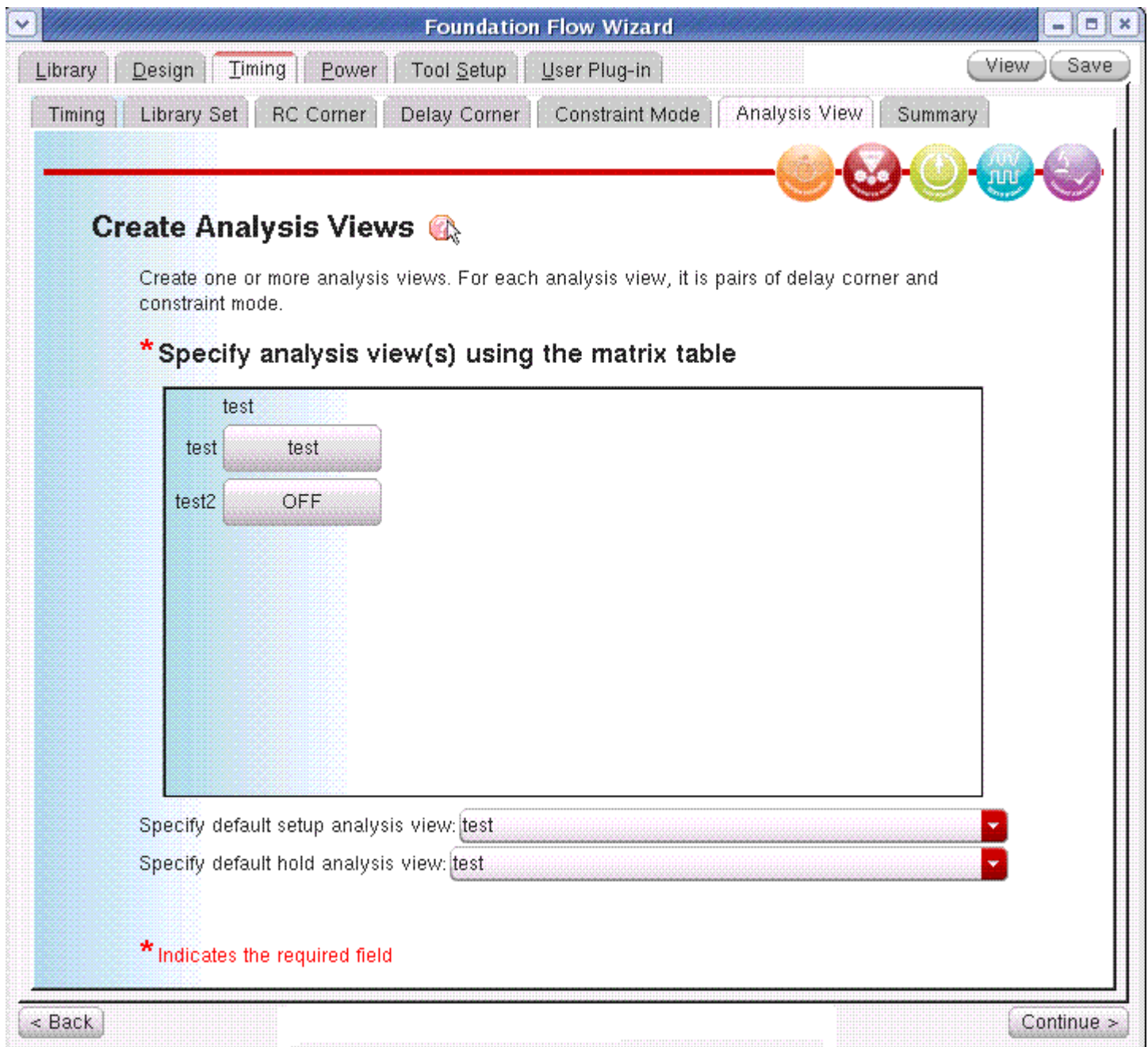
Timing the Flow - Delay Corner

Timing the Flow - Analysis View

Reviewing the Timing Setup

Timing the Flow - Analysis View

The following screen appears:

Innovus Foundation Flows Guide
Using the Wizard--Timing the Flow

Create Analysis Views - Basic Fields and Options

Specify Analysis View(s) using the Matrix Table	Select the button to turn it *On* or *OFF*
Specify Default Setup Analysis Views	Specify the view you want. Choose from the drop-down list
Specify Default Hold Analysis Views	Specify the view you want. Choose from the drop-down list
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Timing the Flow - Timing

Timing the Flow - Library Set

Timing the Flow - RC Corner

Timing the Flow - Delay Corner

Timing the Flow - Constraint Mode

Reviewing the Timing Setup

Reviewing the Timing Setup

In the next screen you can review the configurations you have entered. You can edit them or, if satisfied, click *Continue*. Depending on your entries, the summary will look like this.
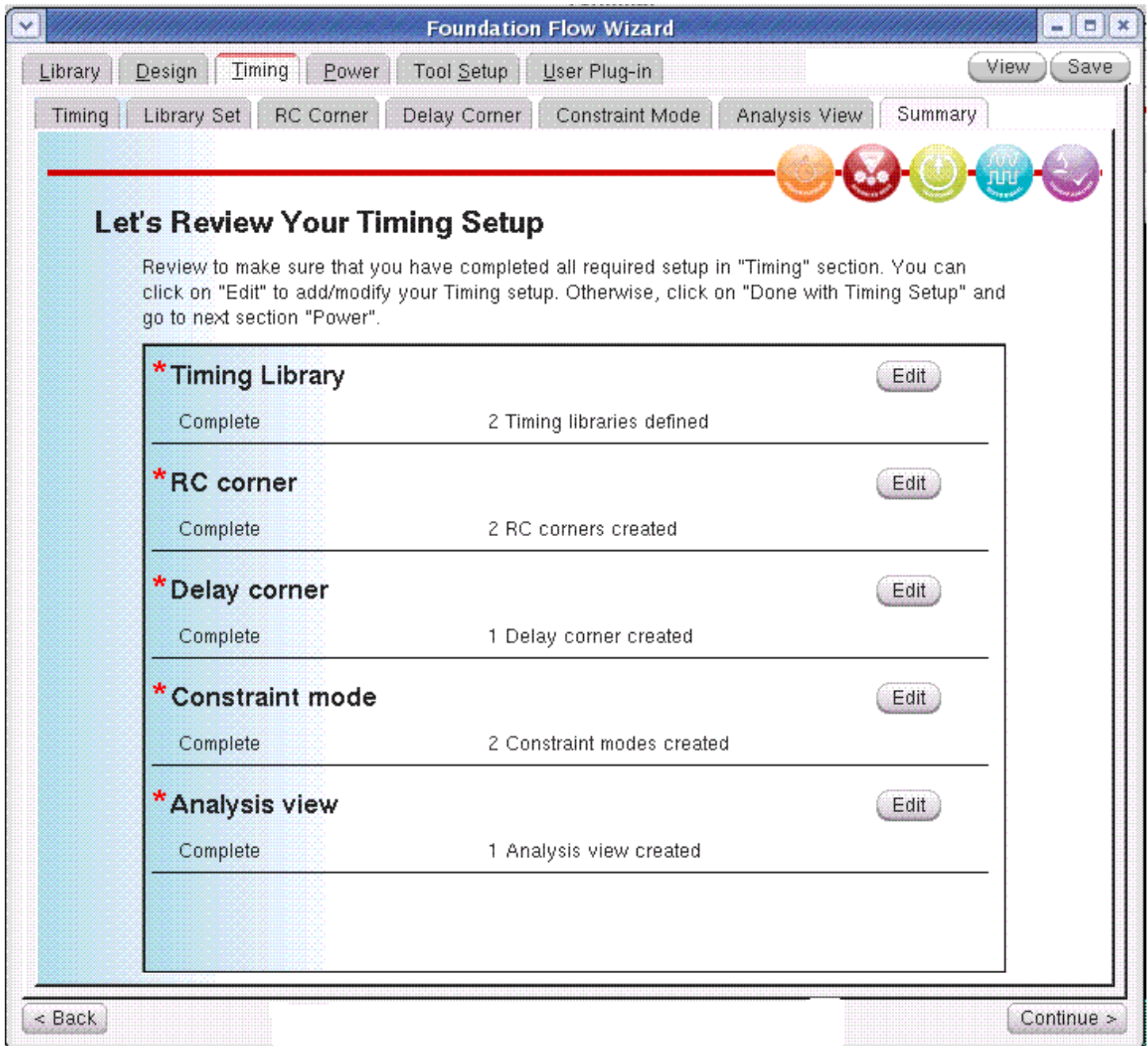
See also,

Timing the Flow - Timing
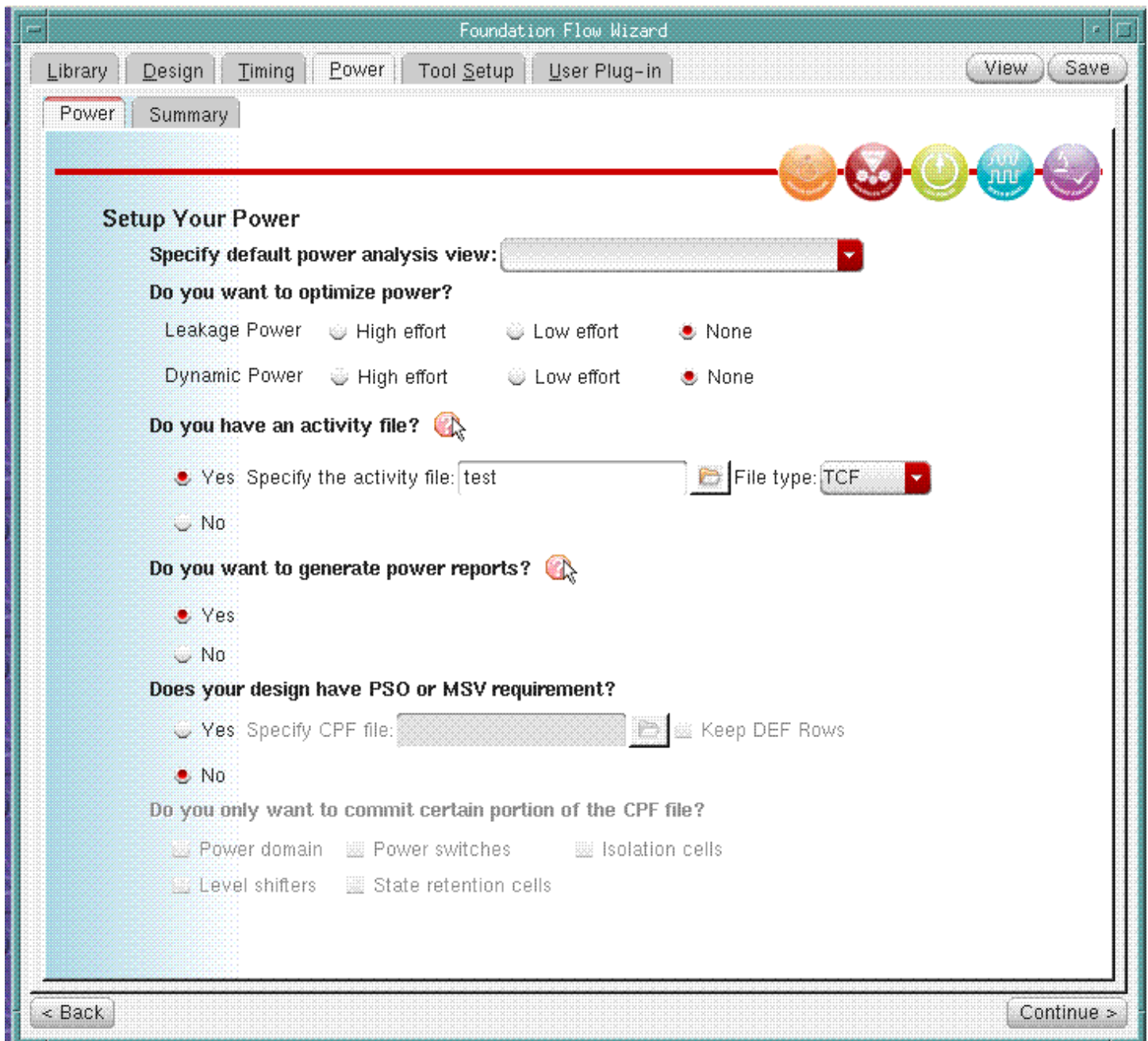
Timing the Flow - Library Set

Timing the Flow - RC Corner

Adding Power to the Flow

The following screen appears:

Innovus Foundation Flows Guide
Using the Wizard--Adding Power to the Flow

Setup Your Power - Basic Fields and Options

Specify Default Power Analysis View	The name of the file appears in a drop-down list, choose it or any other.
Do You Want to Optimize Power	Select *Leakage Power* or *Dynamic Power* to use if you want to optimize power. Each of the options comes in three flavours: *High*, *Medium*, and *Low*. This is optional.
Do you have an Activity File?	Specify the activity file if you have one and select it from the system. Else, select *No*. (Default)
Do you want to Generate Power Reports?	Select *Yes* if you want to generate reports. Else, select *No*. (Default)
Does your Power have MSO or PSV Requirements?	Specify the CPF file if you have MSO or PSV requirements and you can additionally keep DEF rows. Else, select *No*. (Default)
Do you want to Commit Certain Portion of the CPF File?	Select from options if you want to commit certain sections of the CPF files. This option is active if you specify a CPF file in the previous option.
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

Reviewing the Power Setup

In the next screen you can review the configurations you have entered. You can edit them or, if satisfied, click *Continue*.

For other Wizard forms, click link below:

Starting the Flow Wizard

Setting Up the Library for the Flow Wizard

Designing the Flow

Timing the Flow

Setting up the Tool

May 2021   102   Product Version 21.10

Setting up Plug-in Scripts

Completing Setup

Adding Power to the Flow - Summary

The Power summary display.

Setup Power Summary - Basic Fields and Options

Power Analysis View	Displays the name you specified for the power analysis view.
Optimize Power	Lists your selections.
Activity File	Lists your selections.
Report Power	Lists your selections.
CPF File	Lists your selections.
Edit	Click to go back to the previous screen and make changes.

Setting up the Tool

In this section we cover:

- Setting up the Tool - Setup

- Setting up the Tool - Placement

- Setting up the Tool - Optimization

- Setting up the Tool - CTS

- Setting up the Tool - Route

- Setting up the Tool - Signal Intergrity

- Setting up the Tool - Design For Manufacture (DFM)

- Setting up the Tool - Multi-CPU

- Setting up the Tool - Multi-CPU - Distributed Processing

- Reviewing the Tool Setup

For other Wizard forms, click link below:

- Starting the Flow Wizard

- Setting Up the Library for the Flow Wizard

- Designing the Flow

- Timing the Flow

Innovus Foundation Flows Guide
Using the Wizard--Setting up the Tool

- Adding Power to the Flow

- Setting up Plug-in Scripts

- Completing Setup

Setting up the Tool - Setup

The following screen appears:
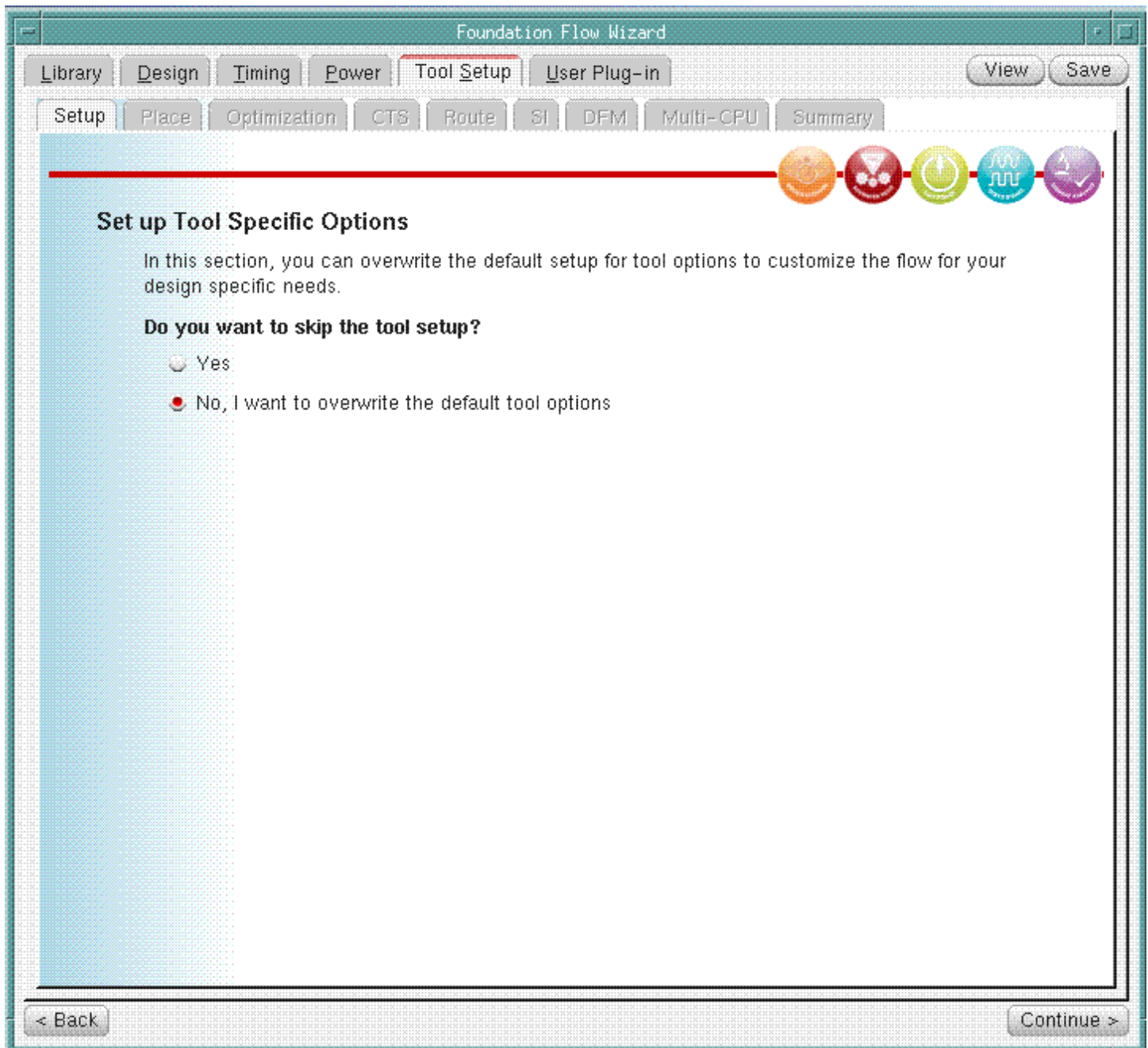
Setup Tool Specific Options Basic Fields and Option

Do you want to Skip the Tool Setup	*Yes* Select if you want to skip. *No* Select if you want to overwrite the default tool options	
Back	Click to move to the previous screen and change any entry, if needed.	
Continue	Click to move to next screen.	

See also,

Setting up the Tool - Placement

Setting up the Tool - Optimization

Setting up the Tool - CTS

Setting up the Tool - Route

Setting up the Tool - Signal Intergrity

Setting up the Tool - Design For Manufacture (DFM)

Setting up the Tool - Multi-CPU

Setting up the Tool - Multi-CPU - Distributed Processing

Reviewing the Tool Setup

Setting up the Tool - Placement

If you choose to set up the tool, the following screen appears:

Foundation Flow Wizard

Library Design Timing Power Tool Setup User Plug-in   View Save

Setup Place Optimization CTS Route SI DFM Multi-CPU Summary

Set up Placement Options

Optionally, you can overwrite the default setting for placement in this section.

☐ Do not run pre-place optimization ☐ Run in-place optimization
☐ Place IO pins ☐ Enable clock gate aware if you have clock gating cells in your design

Specify congestion effort: [medium ▼]

Specify number of rows for jtag placement: [_____]

Specify tie hi/lo cells: [_____] [Select]

Specify filler cells: [_____] [Select]

Specify jtag cells: [_____] [Select]

Add well tap cells?
● No ○ Yes, use max gap rule [_____] ○ Yes, use cell interval rule [_____]
☐ Use welltap checkerboard pattern
Specify verify rules between welltap cells: [_____]

Specify well tap cells: [_____] [Select]

Specify Pre Endcap cells: [_____] [Select]

Specify Post Endcap cells: [_____] [Select]

< Back   Continue >

Set up Placement Options - Basic Fields and Option

Do Not Run Pre-Place Optimization	Check the box to not run Pre-Placement optimization
Run In-place Optimization	Check the box to run in-place optimization
Place IO Pins	Check the box to place IO pins
Enable Clock Gate Aware If You Have Clock Gating Cells in Your Design	Check the box to enable clock gate aware if you have clock gating cells in your design
Specify Congestion Effort	Select from drop-down list.
Specify Number Of Rows For JTAG Placement	Enter value to specify number of rows for JTAG placement
Specify Tie Hi/Lo Cells	Select the file from a new screen that shows you the selected Hi/Lo Cells cells in right hand folder. You can *Add* or *Delete* cells to the left hand folder. Then click *Apply*
Specify Filler Cells	Select the file from a new screen that shows you the selected Filler Cells cells in right hand folder. You can *Add* or *Delete* cells to the left hand folder. Then click *Apply*
Specify Well Taps	Select the file from a new screen that shows you the selected Well Tap cells in right hand folder. You can *Add* or *Delete* cells to the left hand folder. Then click *Apply*
Specify JTAG Cells	Select the file from a new screen that shows you the selected JTAG Cells in right hand folder. You can *Add* or *Delete* cells to the left hand folder. Then click *Apply*
Select	Click to select the required file from a new screen that shows you the selected cells in right hand folder. You can *Add* or *Delete* cells to the left hand folder. Then click *Apply*.

See also,

Setting up the Tool - Optimization

Setting up the Tool - CTS

Setting up the Tool - Route

Setting up the Tool - Signal Intergrity

Setting up the Tool - Design For Manufacture (DFM)

Setting up the Tool - Multi-CPU

Setting up the Tool - Multi-CPU - Distributed Processing

Reviewing the Tool Setup

Setting up the Tool - Optimization

The following screen appears:

Set up Optimization Options - Basic Fields and Option

Assign Buffer	Check box to assign buffer
Preserve Constraints on IO Port	Check box to preserve constraints on IO ports
Specify critical range of TNS Optimization	Enter value for critical range of TNS optimization
High Timing Effort	Check box for High Timing effort
Enable Clock Gate Aware If You Have Clock Gating Cells in Your Design	Check box to enable clock gating aware if you have clock gating cells in your design
Enable Clock Gating	Check box to enable clock gating
Enable useful Skew and Select Skew Buffers	Select value from drop-down to enable useful skew and select skew buffers
Enable Hold Fixing	Check box to enable hold fixing
Disable Hold Fixing for IO	Check box to disable hold fixing for IO
Enable Hold Fixing allow TNS degradation	Check box to enable hold fixing allowing TNS degradation
Resize Shifter and Isolation Instances	Check box to resize shifter and isolation instances
Specify Dont Use List	Enter value for dont use list
Specify Use List	Enter value for use list
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Setting up the Tool - Placement

Setting up the Tool - CTS

Setting up the Tool - Route

Setting up the Tool - Signal Intergrity

Setting up the Tool - Design For Manufacture (DFM)

Setting up the Tool - Multi-CPU

Setting up the Tool - Multi-CPU - Distributed Processing

Reviewing the Tool Setup

Setting up the Tool - CTS

The following screen appears:
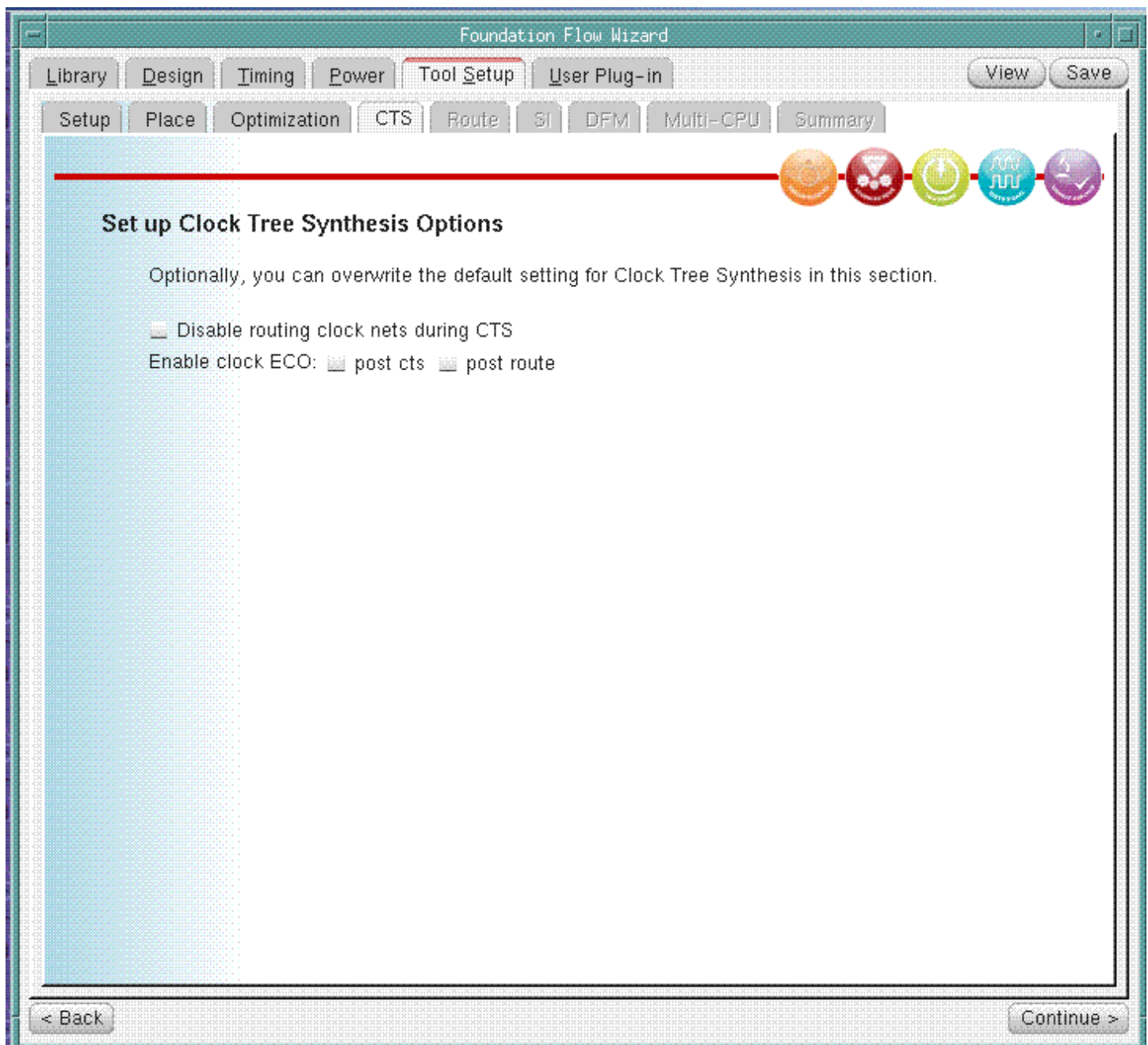
Set up CTS Options - Basic Fields and Option

Disable Routing Clock Nets During CTS	Check box to disable routing
Enable Clock ECO	Select *Post-CTS* or *Post-Route*, or both
Back	To move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Setting up the Tool - Placement

Setting up the Tool - Optimization

Setting up the Tool - Route

Setting up the Tool - Signal Intergrity

Setting up the Tool - Design For Manufacture (DFM)

Setting up the Tool - Multi-CPU

Setting up the Tool - Multi-CPU - Distributed Processing

Reviewing the Tool Setup

Setting up the Tool - Route

The following screen appears:
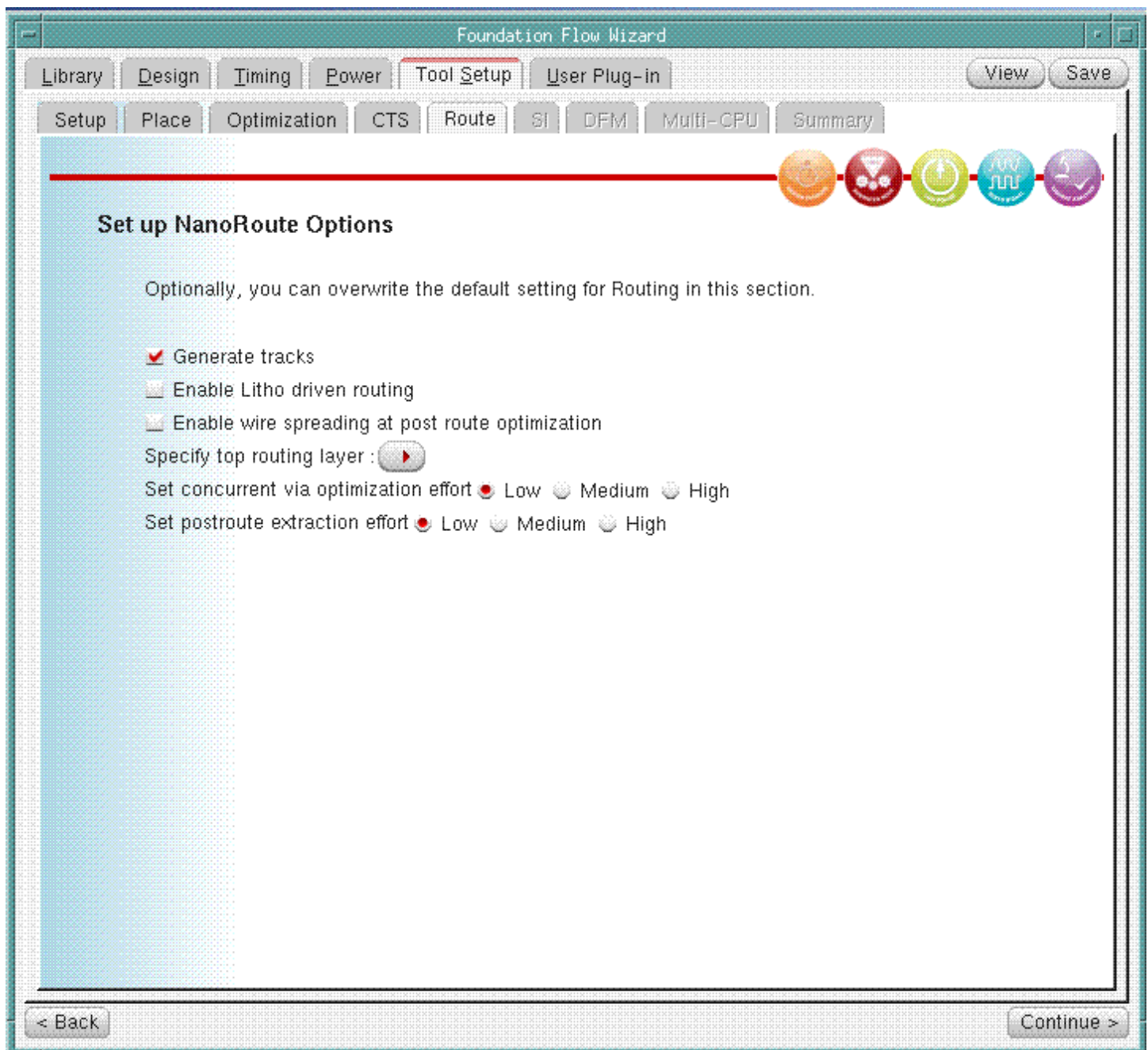
Set up NanoRoute Options - Basic Fields and Option

Generate Tracks	Check box to generate tracks
Enable Litho Driven Routing	Check box to enable litho driven routing
Enable Wire Spreading at Post-Route Optimization	Check box to enable wire spreading at post-route optimization
Specify Top Routing Layer	Select options from drop-down
Set Concurrent via Optimization Effort	Select *Low*, *Medium*, or *High*
Set Post-Route Effort	Select *Low*, *Medium*, or *High*
Back	To move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Setting up the Tool - Placement

Setting up the Tool - Optimization

Setting up the Tool - CTS

Setting up the Tool - Signal Intergrity

Setting up the Tool - Design For Manufacture (DFM)

Setting up the Tool - Multi-CPU

Setting up the Tool - Multi-CPU - Distributed Processing

Reviewing the Tool Setup

Setting up the Tool - Signal Intergrity

Setting up the Tool - Signal Intergrity

The following screen appears:

Set Up SI Options - Basic Fields and Option

How Do You Want to Set Up Extraction Thresholds?	Select radio button if you want to use system defaults or if you want to customize the thresholds, enter, • *Coupling Capacitance Threshold* • *Relative Capacitance Threshold* • *Total Capacitance Threshold* |
How Do You want to Set Noise Thresholds?	Select radio button if you want to use system defaults or if you want to customize the thresholds, enter, • *Delta Delay Threshold* • *Celtic Settings* |
How Will You Be Running Signoff Timing?	Select use *Innovus/ETS* or *3rd party Tool*
What Effort Level of Extraction Do You want for the Signoff Step?	Select *High, Medium, Low* effort
Back	To move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Setting up the Tool - Placement

Setting up the Tool - Optimization

Setting up the Tool - CTS

Setting up the Tool - Route

Setting up the Tool - Design For Manufacture (DFM)

Setting up the Tool - Multi-CPU

Setting up the Tool - Multi-CPU - Distributed Processing

Reviewing the Tool Setup

Setting up the Tool - Design For Manufacture (DFM)

Setting up the Tool - Design For Manufacture (DFM)

The following screen appears:

Set Up DFM Options - Basic Fields and Option

Insert Metal Fill Before	Select where you want to insert metalfill from drop-down menu
Specify Custom Tcl to Add Metalfill	Select the file from a Windows Explorer format which shows the files in directories
Open Access Abstract View Name	Select the file from a Windows Explorer format which shows the files in directories
Open Access Layout View Name	Select the file from a Windows Explorer format which shows the files in directories
Open Access Referance Library	Select the file from a Windows Explorer format which shows the files in directories
GDS Map File	Select the file from a Windows Explorer format which shows the files in directories
List of GDS Files	List of GDS Files opens. Click >> to open right pane for you to browse the exact file, select and *Add*. The file and path shows up on the left pane. Provide a name for the GDS File and click Close.
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Setting up the Tool - Placement

Setting up the Tool - Optimization

Setting up the Tool - CTS

Setting up the Tool - Route

Setting up the Tool - Signal Intergrity

Setting up the Tool - Multi-CPU

Setting up the Tool - Multi-CPU - Distributed Processing

Reviewing the Tool Setup

Setting up the Tool - Multi-CPU

The following screen appears:
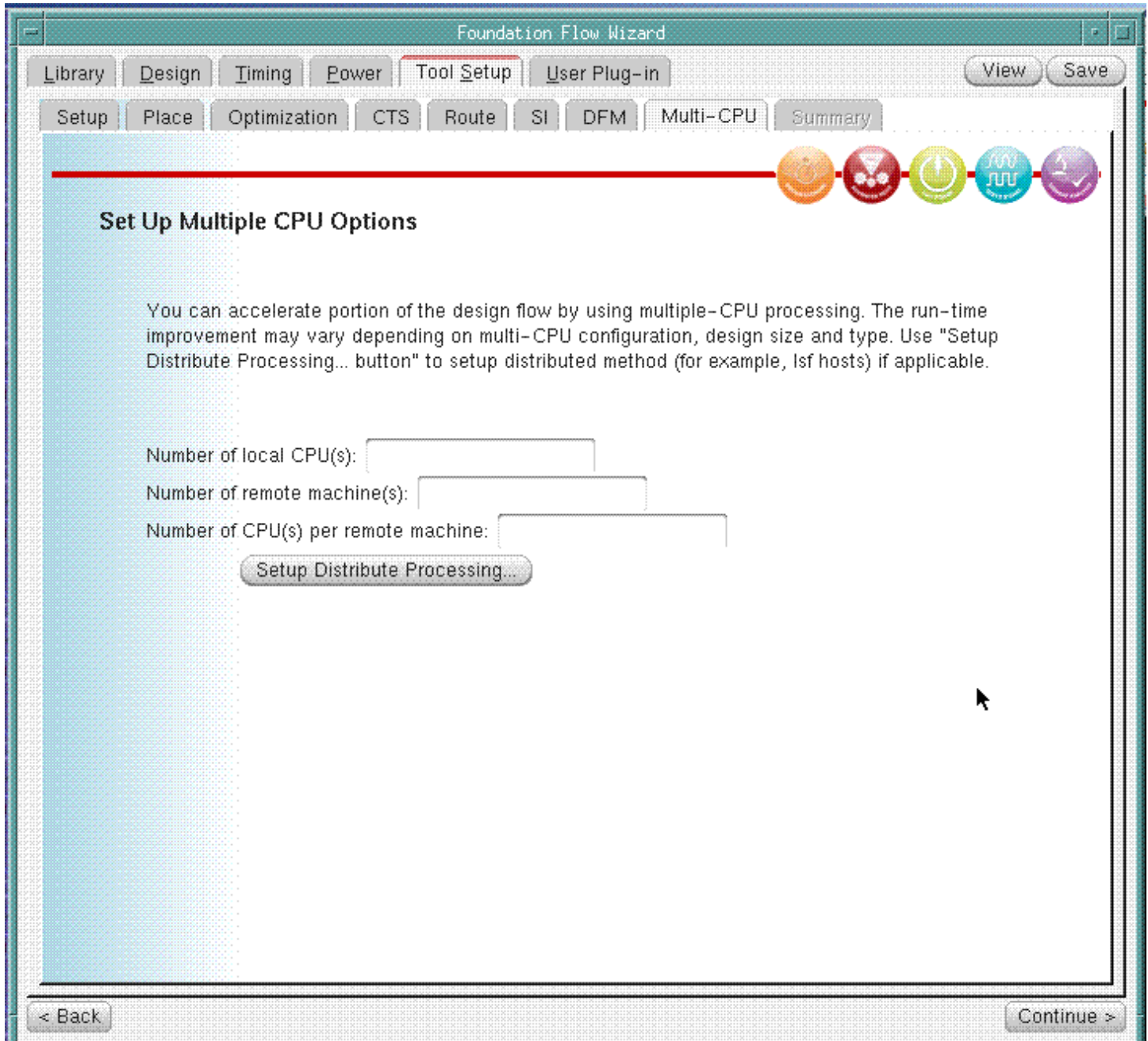
Tool Setup - Multi-CPU Options - Basic Fields and Option

Number of Local CPUs	Enter value for number of local CPUs
Number of Remote Machines	Enter value for number of remote machines
Number of CPUs per Remote Machine	Enter value for number of CPUs per remote machine
Setup Distribute Processing	Click to setup distribued processing. the following screen opens: Setting up the Tool - Multi-CPU - Distributed Processing.
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

See also,

Setting up the Tool - Placement

Setting up the Tool - Optimization

Setting up the Tool - CTS

Setting up the Tool - Route

Setting up the Tool - Signal Intergrity

Setting up the Tool - Design For Manufacture (DFM)

Reviewing the Tool Setup

Setting up the Tool - Multi-CPU - Distributed Processing

The following screen appears:

Tool Setup - Multi-CPU Options - Distributed Processing - Basic Fields and Option

LSF	Select and enter values for *Queue*, *Resource*, *Arguments*
Custom	Select and enter script path
Local	Select if CPUs are local

| RSH | Select and provide *Host Name* |

See also,

Setting up the Tool - Placement

Setting up the Tool - Optimization

Setting up the Tool - CTS

Setting up the Tool - Route

Setting up the Tool - Signal Intergrity

Setting up the Tool - Design For Manufacture (DFM)

Setting up the Tool - Multi-CPU

Reviewing the Tool Setup

Reviewing the Tool Setup

In the next screen you can review the configurations you have entered. You can edit them or, if satisfied, click *Continue*. Depending on your entries, the summary will look like this.
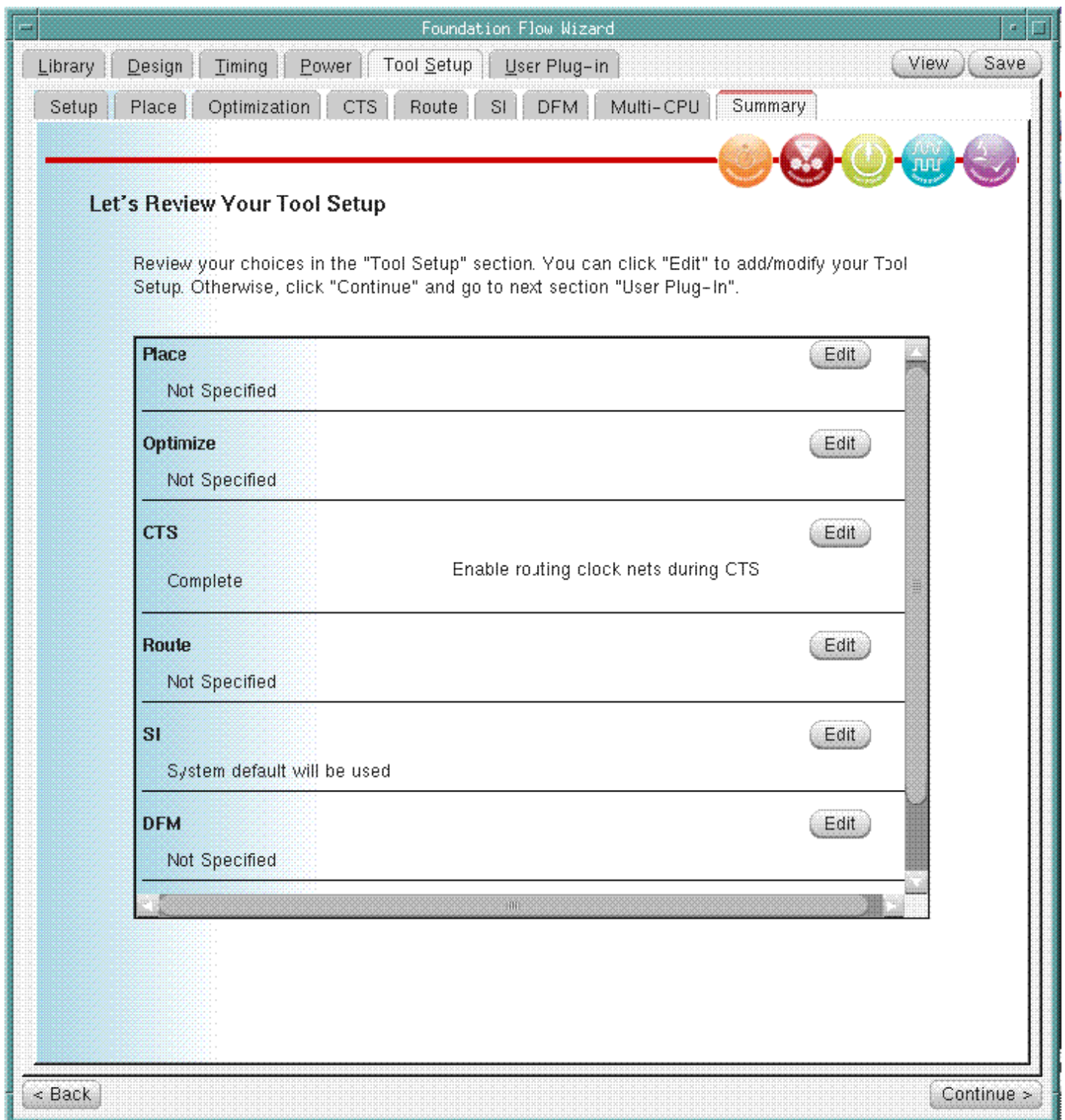
Set up Summary Options - Basic Fields and Option

Place	Lists your Power selections
Optimize	Lists your Optimization selections
CTS	Lists your CTS selections
Route	Lists your Route selections
SI	Lists your SI selections
DFM	Lists your DFM selections
Multi-CPU	Lists your multi-CPU selections
Edit	Click to go back to screen and make changes

See also,

Setting up the Tool - Placement

Setting up the Tool - Optimization

Setting up the Tool - CTS

Setting up the Tool - Route

Setting up the Tool - Signal Intergrity

Setting up the Tool - Design For Manufacture (DFM)

Setting up the Tool - Multi-CPU

Setting up the Tool - Multi-CPU - Distributed Processing

Setting up Plug-in Scripts

The following screen appears:
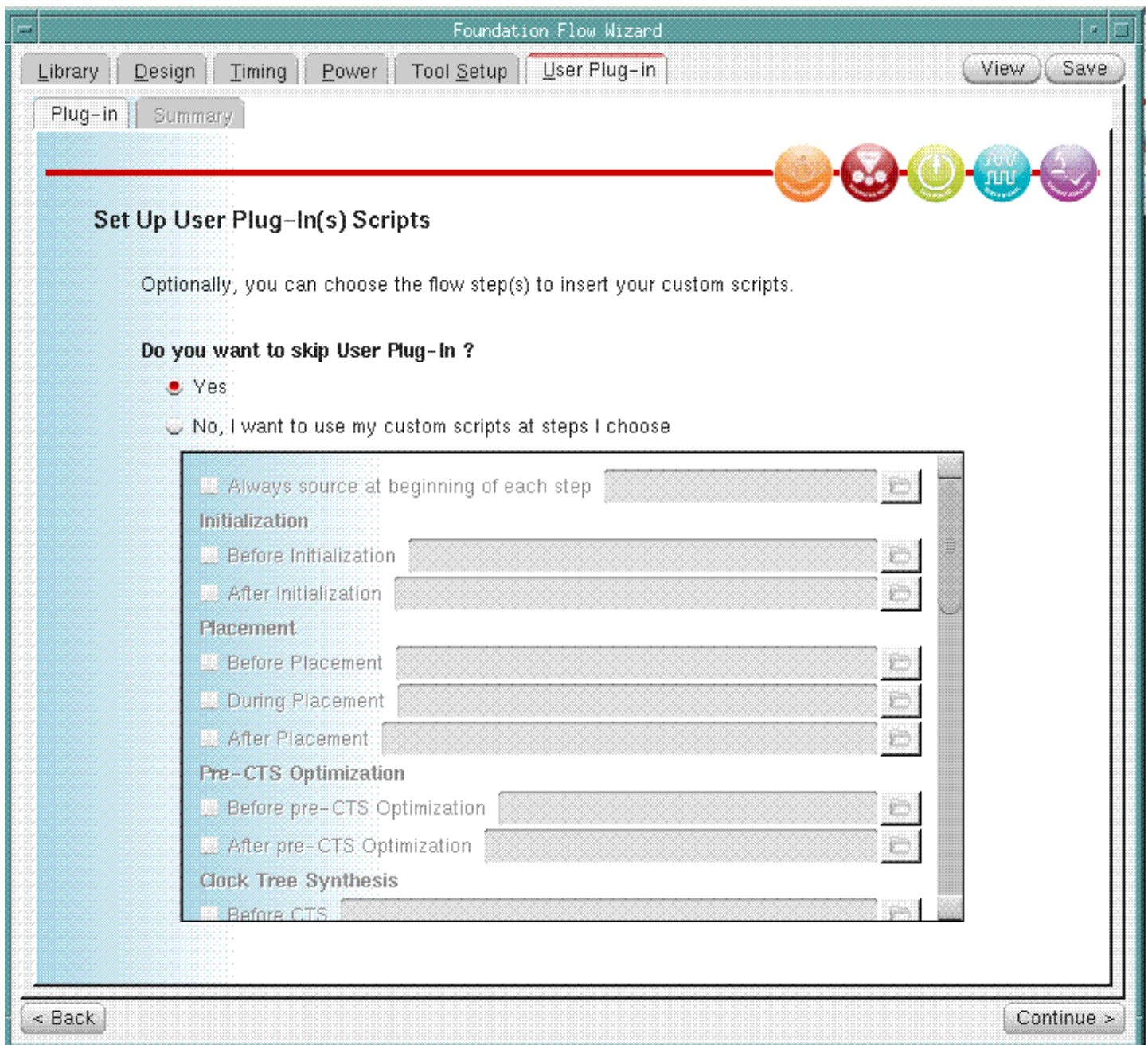
Setup User Plug-in Scripts - Basic Fields and Options

Do you want to Skip User Plug-in?	*Yes. No* Specify entries if you want to run your own scripts at specific stages of the flow.

Always Source at Beginning of Each Step	Check box and select file
Initialization - Before Installation or After Installation or	Check box and select file
Placement - Before Placement or During Placement or After Placement	Check box and select file
Pre-CTS Optimization - Before Pre-CTS Optimization or After pre-CTS Optimization	Check box and select file
Clock Tree Synthesis - Before CTS or During CTS or After CTS	Check box and select file
Post CTS Optimization - Before post-CTS Optimization or After post-CTS Optimization	Check box and select file
Routing - Before Routing or After Routing	Check box and select file
Post-Route Optimization - Before Post-Route Optimization or After Post-Route Optimization	Check box and select file
Post-Route Optimization + Hold Fixing - Before Post-Route Optimization or After Post-Route Optimization	Check box and select file
Post-Route Optimization + Hold Fixing + SI - Before Post-Route Optimization or After Post-Route Optimization	Check box and select file
Post-Route Signoff - Before Post-Route Signoff or After Post-Route Signoff	Check box and select file
Sign-off - Before Signoff or After Signoff	Check box and select file
Back	Click to move to the previous screen and change any entry, if needed.
Continue	Click to move to next screen.

For other Wizard forms, click link below:

Starting the Flow Wizard

Setting Up the Library for the Flow Wizard

Designing the Flow

Timing the Flow

Adding Power to the Flow

Setting up the Tool

Completing Setup

Completing Setup

The following screen appears:
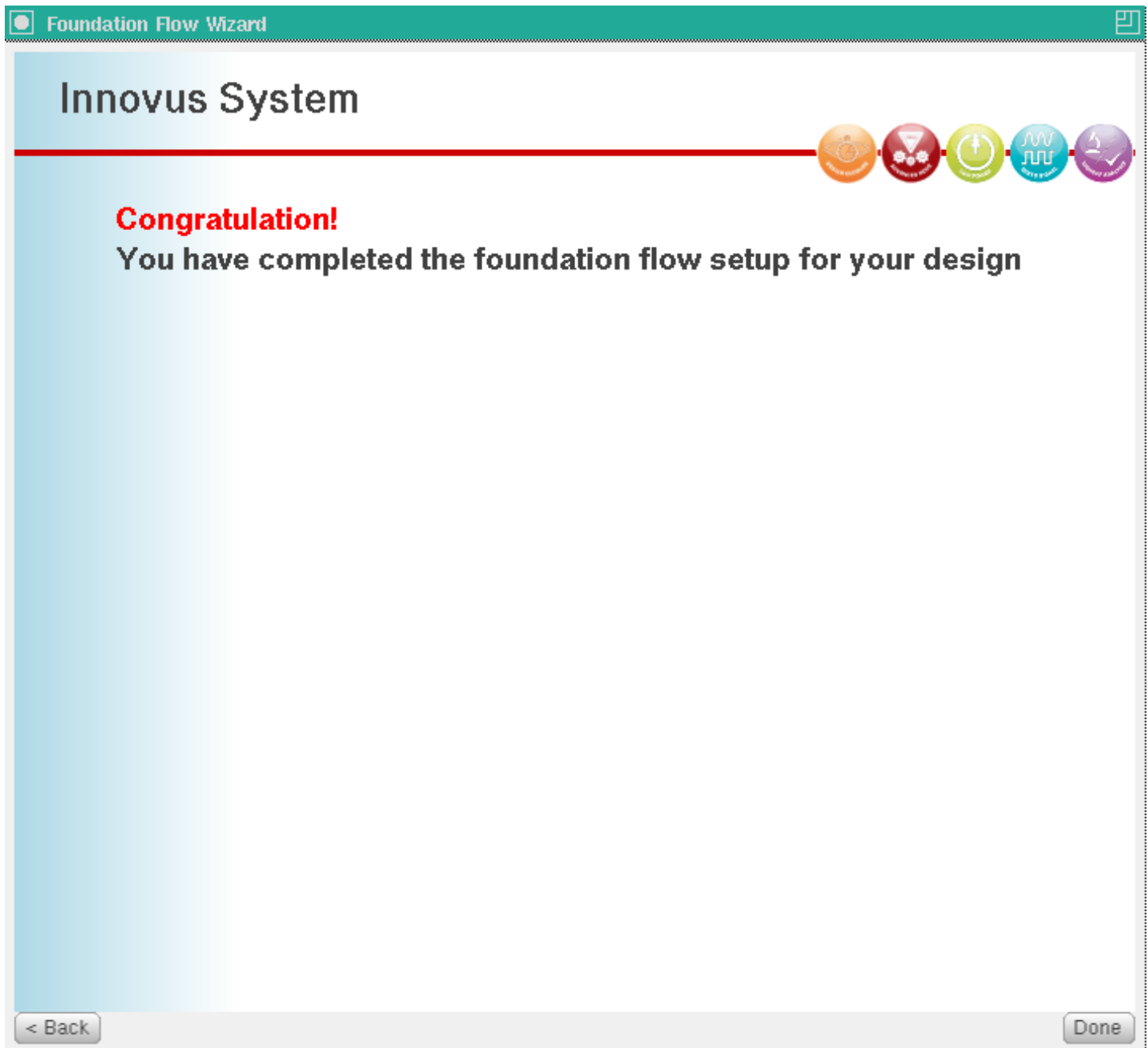
Completing - Basic Fields and Options

The final screen shows the wizard is now set up and can save entries in a setup.tcl file.

For other Wizard forms, click link below:

Starting the Flow Wizard

Setting Up the Library for the Flow Wizard

Designing the Flow

Timing the Flow

Adding Power to the Flow

Setting up the Tool

Setting up Plug-in Scripts

5

Code Generator

From this release, Innovus provides a code generator to enable users to quickly generate scripts to load their design data, setup their timing environment, and execute the recommended implementation flow from placement through signoff.

Note: Foundation Flow code generator generates EDI 11.1 based scripts automatically. No changes to `setup.tcl` are required. With this release, the design import command `init_design` replaces the command `loadConfig`.

In this section we cover:

- Introduction to Code Generator

- Usage

- Example Scripts

- Results

- Error.Warning Messages

Introduction to Code Generator

From this release, Innovus provides a code generator to enable users to quickly generate scripts to load their design data, setup their timing environment, and execute the recommended implementation flow from placement through signoff. The feature helps the users overcome the earlier difficult to read and debug scripts. This feature helps those customers who would like to use the scripts as a reference flow as well as for auditing purposes where readability is important.

The following diagram depicts how the Code Generator fits into the Foundation Flow.
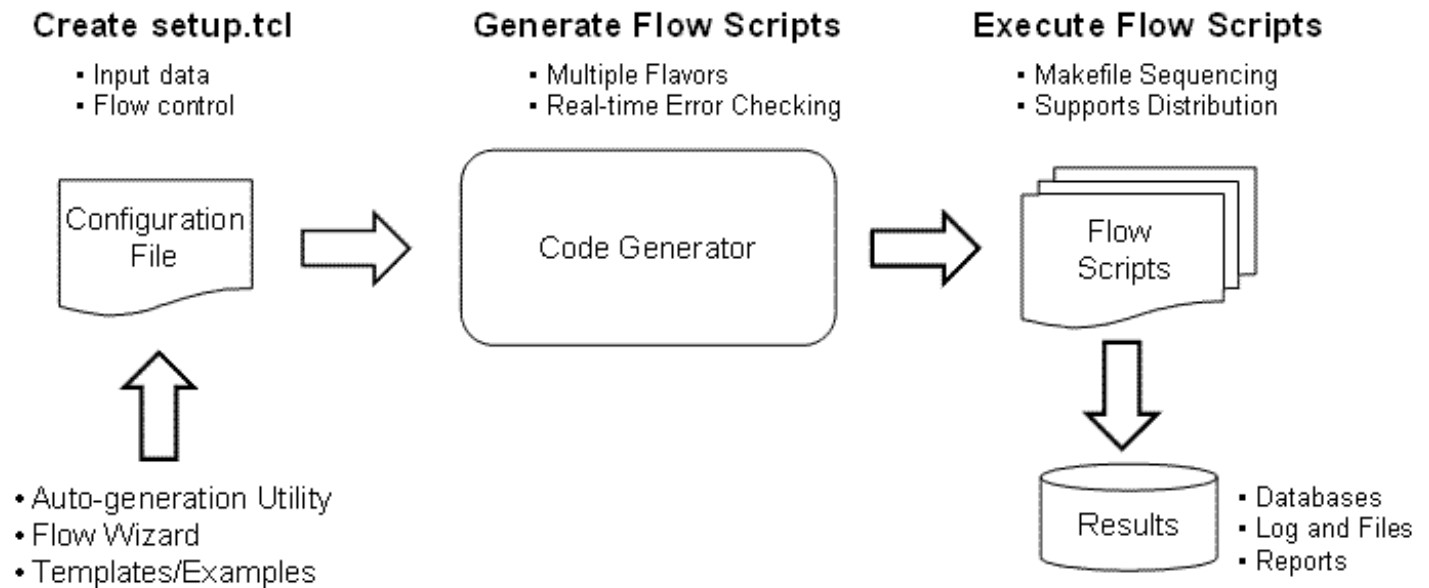
The foundation flow is in the SCRIPTS directory and the code generator is `SCRIPTS/gen_flow.tcl` (symbolic link to `gen_edi_flow.tcl`).

To learn more on Innovus Code Generator follow these links:

- Advantages
- TCL Command
- Command Line Options
- Examples:

Advantages

The following are the advantages of the Code Generator:

- Through the Code Generator the scripts will be generated instead of being executed.

- All flow will have the command sequence based on the user settings. A clean, flat script will be generated for conditional code that is based on static user settings (for example, `setup.tcl`).

- All conditional code branches that require run time specific data will be generated as conditionals (that is, they cannot be flattened).
  For example, since analysis view can be altered via user plug-ins, checks must be made to determine which corners are active:

```
set vars(active_rc_corners) [list]
foreach view [concat [all_setup_analysis_views] [all_hold_analysis_views]] {
set corner [get_delay_corner [get_analysis_view $view -delay_corner] -rc_corner]
if {[lsearch $vars(active_rc_corners) $corner] == -1 } {
lappend vars(active_rc_corners) $corner
}
```

- The following procedures will be covered in `procs.tcl`.

Procedure	Type	Description
system_info		Reports system info of the host machine `vars(report_system_info)`
source_file	*file*	Wrapper around source to catch syntax errors
source_plug	*plug_in*	Loads one or more plug-in scripts, catch errors, etc.
report_time		Reported elapsed CPU time `vars(report_run_time)`

- The code generator is backwards compatible (it will work with existing code).

- For the LP flow, there are these additional procedures:

Procedure	Description
get_power_domains	Returns a list of power domains
modify_power_domains	Modifies power domains*
add_power_switches	Adds power switches*
route_secondary_pg_nets	Routes secondary power/ground nets*
insert_welltaps_endcaps	Inserts welltaps and endcaps cells

Note: * operates based on user variable settings in the `lp_config.tcl` file

These procedures are used by the foundation flow at various points in the scripts. They are also available for the user to call in plug-in scripts. To access use the following syntax:

```
FF_EDI::proc_name [option]
```

Note: Once the scripts have been generated, the use model will be identical to previous releases

- When the scripts are generated, the code generator will check for data consistency; i.e. all

required variables are properly defined, necessary files exists, etc.

TCL Command

Usage: `tclsh SCRIPTS/gen_flow.tcl` *options steps*

Step	Description
a	Execute step a only
a-b	Execute all steps between step a and b, inclusive
a-	Execute all steps in the flow from step a, inclusive
all	Execute all steps in the flow

Note: "a" and "b" should be valid steps for the specific mode.

Command Line Options

The following are the command line options:

Command	Description
-h | -help	Provides verbose help on the support variables.
-m | -mode	Sets the script mode. Default is "flat". Valid modes are: flat and hier
-f | -flat	Flattens/unrolls levels.
- full	Unrolls plug-ins and tags (include contents in the main flow script)
- none	References plug-ins and tags as procedures.
-d | -dir	The code generator creates a design specific set of run scripts in a FF directory in the local user directory. You can optionally change the name of the output directory using this option.

-s | -setup	Provide the directory containing the setup.tcl setup file required to run the Foundation Flow. Default is ".".

Examples:

```
gen_flow –H route
```

The above command returns a list of variables related to the route step, as shown below:

```
===============================================================================
=====================
                                      variable (command) : values
------------------------------------------------- route ----------------------------
----------------
                              in_route_opt (route_design) : FALSE true
   route_secondary_pg_nets (route_pg_pin_use_signal_route) : true FALSE
         secondary_pg_nets (route_pg_pin_use_signal_route) : List of global nets for
secondary power/ground
                         route_clock_nets (set_cts_mode) : TRUE false
                litho_driven_routing (set_nanoroute_mode) : true FALSE
                    multi_cut_effort (set_nanoroute_mode) : MEDIUM high
             postroute_spread_wires (set_nanoroute_mode) : true FALSE
  secondary_pg,cell_pin_pairs (set_pg_pin_use_signal_route) : List of cell:pin pairs
-------------------------------------------------- dbs ----------------------------
----------------
                                        dbs_dir (none) : Database directory
                              dbs_format (save_design) : FE oa
                    oa_abstract_name (save_design) : OA Abstract view name
                      oa_layout_name (save_design) : OA Layout view name
                    save_constraints (save_design) : true FALSE (Save
constraints with DBS?)
                              save_rc (save_design) : true FALSE (Save RCDB with
DBS?)
------------------------------------------------- report ----------------------------
----------------
                              capture_metrics (none) : true FALSE
                                  check_setup (none) : TRUE false
                                 html_summary (none) : HTML summary file
```

```
                                report_power (none) : TRUE false
                             report_run_time (none) : TRUE false
                          report_system_info (none) : true FALSE
                                     rpt_dir (none) : Reports directory
                                 time_info_db (none) : Time info DB file
                                time_info_rpt (none) : Time info report file
================================================================================
======================
```

Note: Default values are in uppercase (all capitals). Valid categories include:

- flow
- hier
- dbs
- synth
- init
- ilm
- mmmc
- power
- place
- opt
- cts
- hold
- route
- extract
- noise
- report
- distribute
- misc

See also,

Usage

Example Scripts

Results

Error.Warning Messages

Usage

Below are the steps to use the Code Generator:

1. Dump the foundation flow code generator from the Innovus interface (GUI or TUI) or download from the download manager.

    ◦ GUI: Flows > Create Foundation Flow Template

    ◦ TUI: writeFlowTemplate --directory *directory*

2. Create a `setup.tcl` through either Using the Wizard or using `gen_setup.tcl`.

    ◦ To create using wizard, simply invoke the wizard and follow the on screen instructions. When finished, save as `setup.tcl`.

    ◦ To create using `gen_setup.tcl`,

        ▪ load an existing Innovus database

        ▪ and `source` *directory*`/SCRIPTS/gen_setup.tcl` (the existing data base should be the starting point for the foundation flow)

        ▪ Edit the resulting setup.auto.tcl and edi_config.auto.tcl files (if necessary)

        ▪ Rename the files to `setup.tcl` and edi_`config.tcl`, respectively.

3. Run the code generator: `tclsh` *directory*`/SCRIPTS/gen_flow.tcl` *options*

4. Execute the flow: `make`

Note: See the "Makefile Options - Script Updates" and "Makefile Options - Single Process Execution" sections of this page for more information.

To learn more on Innovus Code Generator Usage see:

- Execution Tips
- Makefile Options - Script Updates
- Makefile Options - Single Process Execution

Execution Tips

Once the flow scripts are generated the use model for the flow execution differs from earlier in these ways:

- The generated scripts are expaned (unrolled) based on the variables in the `setup.tcl`, so any change in the `setup.tcl` will only affect the scripts if they are regenerated.

```
– make init
```

- The code generator runs again automatically. This is intended to keep the generated scripts synchronized with `setup.tcl`. This can be disabled by running one of the following three commands:

```
make UPDATE=no
gen_flow.tcl -n
set vars(make_update) no
```

- Makefile generation can be disabled entirely using the `-n` option to gen_edi_flow.tcl

```
make prects
```

- The flow will run the steps from <last completed> to `prects`
- The flow can be executed either in a single process or in multi-process; the default is multi-process (i.e. each step runs a separate Innovus process). This is more deterministic but also increases runtime due to re-loading and re-timing the step databases. To run a single process flow, use

```
make single FF_STOP=prects
```

- Any completed step in the flow can be checked for logical equivalence (against `vars(netlist)`) by running:

```
make lec_<step>
```

- Any completed step in the flow loaded into an Innovus GUI session by running:

```
make debug_<step>
```

Makefile Options - Script Updates

Since the script are now dynamic (generated) vs static, a change in the configuration files say setup.tcl will only take effect when scripts are re-generated. So, the default behavior of the Makefile is to update the scripts automatically anytime one of the input configuration files change. This behavior can be disabled, by:

```
make UPDATE=no
```

Note: A new option `?u|-rundir` allows the scripts/Makefile to be generated elsewhere than the current work directory ".".

.

Makefile Options - Single Process Execution

The foundation flow defaults to executing each step using different Innovus process (a new Innovus session is invoked for each step). To invoke the foundation flow using a single process, use:

make single
Additionally, to invoke a selected set of steps using a single process, use:

make single FF_STOP=end_step
This will start at the current step and stop at the step FF_STOP.

Note: To force the FF to re-run previously executed steps, use FF_START=begin_step or simply remove the make semaphore files for the steps you want to re-run. The make protected variable or abstract data type (semaphore) is the step file in the make directory (i.e. make/place, make/prects) which denotes which steps have completed successfully.

See also,

Introduction to Code Generator

Example Scripts

Results

Error.Warning Messages

Example Scripts

Following are the examples of the code generation:

- SINGLE STEP

- STEP RANGE

- ALL STEPS

- HIERARCHICAL FLOW - Example Scripts

See also,

Introduction to Code Generator

Usage

Results

Error.Warning Messages

SINGLE STEP

```
% tclsh SCRIPTS/gen_flow.tcl init

<FF> Finished loading setup.tcl

<FF> GENERATING STEP init


-----------------------------------------------

<FF> CODE GENERATION COMPLETE


-----------------------------------------------
```

STEP RANGE

```
% tclsh SCRIPTS/gen_flow.tcl route-signoff

<FF> Finished loading setup.tcl

<FF> GENERATING STEP route
```

```
<FF> GENERATING STEP postroute

<FF> GENERATING STEP postroute_hold

<FF> GENERATING STEP postroute_si_hold

<FF> GENERATING STEP postroute_si

<FF> GENERATING STEP signoff


---------------------------------------------


<FF> CODE GENERATION COMPLETE


---------------------------------------------
```

ALL STEPS

```
w/OUTPUT DIRECTORY RENAMED (-d MYFF)

% tclsh SCRIPTS/gen_flow.tcl -d MYFF all

<FF> OUTPUT DIRECTORY == MYFF

<FF> Finished loading setup.tcl


---------------------------------------------


<FF> Generating scripts for dtmf_recvr_core



---------------------------------------------


w/setup.tcl located in another directory (-s MYSETUPDIR) and plug-ins/tags imported (-f full)
```

```
% tclsh SCRIPTS/gen_flow.tcl -s MYSETUPDIR -f full all
```

HIERARCHICAL FLOW - Example Scripts

HIERARCHICAL FLOW

```
% tclsh SCRIPTS/gen_flow.tcl -m hier al


----------------------------------------------


<FF> Generating scripts for dtmf_recvr_core


----------------------------------------------


<FF> GENERATING STEP assemble


<FF> Finished loading Innovus configuration file


<FF> GENERATING STEP partition


----------------------------------------------


<FF> Generating scripts for dtmf_recvr_core


----------------------------------------------


<FF> GENERATING STEP init


<FF> GENERATING STEP place


<FF> GENERATING STEP prects


<FF> GENERATING STEP cts
```

```
<FF> GENERATING STEP postcts

<FF> GENERATING STEP postcts_hold

<FF> GENERATING STEP route

<FF> GENERATING STEP postroute

<FF> GENERATING STEP postroute_hold

<FF> GENERATING STEP postroute_si_hold

<FF> GENERATING STEP postroute_si

<FF> GENERATING STEP signoff

-----------------------------------------------

<FF> GENERATING flat Makefile ...

-----------------------------------------------

<FF> GENERATING STEP debug

-----------------------------------------------

<FF> Generating scripts for tdsp_core

-----------------------------------------------

<FF> GENERATING STEP init

<FF> GENERATING STEP place

<FF> GENERATING STEP prects
```

```
<FF> GENERATING STEP cts

<FF> GENERATING STEP postcts

<FF> GENERATING STEP postcts_hold

<FF> GENERATING STEP route

<FF> GENERATING STEP postroute

<FF> GENERATING STEP postroute_hold

<FF> GENERATING STEP postroute_si_hold

<FF> GENERATING STEP postroute_si

<FF> GENERATING STEP signoff

----------------------------------------------

<FF> GENERATING flat Makefile ...

----------------------------------------------

<FF> GENERATING STEP debug

----------------------------------------------

<FF> GENERATING hier Makefile ...

----------------------------------------------

<FF> GENERATING STEP debug

----------------------------------------------
```

```
<FF> CODE GENERATION COMPLETE


-----------------------------------------------


See Sample Script - Code Generator for an example implementation.
```

Results

The code generator default to creating an instance of the foundation flow in a `FF` directory in the local user directory. The user can optionally override this using the option
`-d directory.`
Here is an example of the files created by the code generator:

- Flat Mode (all steps)
- Hierarchical Mode (all steps)

See also,

Introduction to Code Generator

Usage

Example Scripts

Error.Warning Messages

Flat Mode (all steps)

Flat Mode (all steps):

```
FF


FF/vars.tcl


FF/run.conf


FF/procs.tcl
```

```
FF/Innovus

FF/Innovus/run_all.tcl

FF/Innovus/run_init.tcl

FF/Innovus/run_place.tcl

FF/Innovus/run_prects.tcl

FF/Innovus/run_cts.tcl

FF/Innovus/run_postcts.tcl

FF/Innovus/run_route.tcl

FF/Innovus/run_postroute.tcl

FF/Innovus/run_postroute_si.tcl

FF/Innovus/run_signoff.tcl

FF/Innovus/run_debug.tcl

FF/Innovus/gen_html.tcl

Makefile
```

Hierarchical Mode (all steps)

Hierarchical Mode (all steps):

```
Makefile
```

```
FF/vars.tcl
```

```
FF/run.conf
```

```
FF/procs.tcl
```

```
FF/Innovus/run_assemble.tcl
```

```
FF/Innovus/run_partition.tcl
```

```
FF/Innovus/run_debug.tcl
```

```
PARTITION
```

```
PARTITION/Makefile.partition
```

```
PARTITION/Makefile
```

```
PARTITION/dtmf_recvr_core/FF
```

```
PARTITION/dtmf_recvr_core/FF/vars.tcl
```

```
PARTITION/dtmf_recvr_core/FF/Innovus
```

```
PARTITION/dtmf_recvr_core/FF/Innovus/run_all.tcl
```

```
PARTITION/dtmf_recvr_core/FF/Innovus/run_init.tcl
```

```
PARTITION/dtmf_recvr_core/FF/Innovus/run_place.tcl
```

```
PARTITION/dtmf_recvr_core/FF/Innovus/run_prects.tcl
```

```
PARTITION/dtmf_recvr_core/FF/Innovus/run_cts.tcl
```

```
PARTITION/dtmf_recvr_core/FF/Innovus/run_postcts.tcl
```

PARTITION/dtmf_recvr_core/FF/Innovus/run_postcts_hold.tcl

PARTITION/dtmf_recvr_core/FF/Innovus/run_route.tcl

PARTITION/dtmf_recvr_core/FF/Innovus/run_postroute.tcl

PARTITION/dtmf_recvr_core/FF/Innovus/run_postroute_hold.tcl

PARTITION/dtmf_recvr_core/FF/Innovus/run_postroute_si_hold.tcl

PARTITION/dtmf_recvr_core/FF/Innovus/run_postroute_si.tcl

PARTITION/dtmf_recvr_core/FF/Innovus/run_signoff.tcl

PARTITION/dtmf_recvr_core/FF/Innovus/run_debug.tcl

PARTITION/dtmf_recvr_core/FF/procs.tcl

PARTITION/dtmf_recvr_core/Makefile

PARTITION/tdsp_core/FF

PARTITION/tdsp_core/FF/vars.tcl

PARTITION/tdsp_core/FF/Innovus

PARTITION/tdsp_core/FF/Innovus/run_all.tcl

PARTITION/tdsp_core/FF/Innovus/run_init.tcl

PARTITION/tdsp_core/FF/Innovus/run_place.tcl

PARTITION/tdsp_core/FF/Innovus/run_prects.tcl

PARTITION/tdsp_core/FF/Innovus/run_cts.tcl

```
PARTITION/tdsp_core/FF/Innovus/run_postcts.tcl

PARTITION/tdsp_core/FF/Innovus/run_postcts_hold.tcl

PARTITION/tdsp_core/FF/Innovus/run_route.tcl

PARTITION/tdsp_core/FF/Innovus/run_postroute.tcl

PARTITION/tdsp_core/FF/Innovus/run_postroute_hold.tcl

PARTITION/tdsp_core/FF/Innovus/run_postroute_si_hold.tcl

PARTITION/tdsp_core/FF/Innovus/run_postroute_si.tcl

PARTITION/tdsp_core/FF/Innovus/run_signoff.tcl

PARTITION/tdsp_core/FF/Innovus/run_debug.tcl

PARTITION/tdsp_core/FF/procs.tcl

PARTITION/tdsp_core/Makefile
```

Error.Warning Messages

The following are the most frequent error and warning messages.

- Warning on Missing Required Data
- Warning to Replace Old Variables

See also,

Introduction to Code Generator

Usage

Example Scripts

Results

Warning on Missing Required Data

When (required) data is missing or there are syntax errors in the setup.tcl, code generator will abort with an error message. You can override the error messages by using
set vars(abort) 0
when you set this, an error message will still be issued but the code generator will complete.
A sample error message:

```
tclsh SCRIPTS/gen_flow.tcl all


---------------------------------------------


<FF> Generating scripts for dtmf_recvr_core


---------------------------------------------


#  -------------------------------------------------


#                    Error Summary


#  -------------------------------------------------


#  (1) A verilog netlist file must be defined


#  ------------------------------------
```

Warning to Replace Old Variables

Warning to Replace Old Variables

When old variables that have changed are being used the system displays this error message so that the user can update their `setup.tcl` accordingly.
A sample error message:
Warning Summary

```
Variable xcap_factor has been changed to post_route_xcap_factor ... please update
```

6

CPF-Based Low Power Flow

CPF-Based Low Power Flow is one of the important implementation of the foundation flow.

In this section we cover:

- Introduction to CPF-based Low Power Implementation
- Flow Overview of CPF-based Low Power Flow
- Before you Begin Implementing CPF-based Low Power Flow
- Input for CPF-based Low Power Flow
    - Additional Input

Introduction to CPF-based Low Power Implementation

This guide describes the sequence and details of the tasks in the default and recommended Innovus Implementation System (Innovus) software flow to implement a block or flat chip using low power techniques such as multiple-supply voltage (MSV) and power shutoff. This flow assumes that the floorplan is complete (though it does allow the flexibility to implement floorplanning tasks through plug-in which can be called in the flow). The flow lets you finish the implementation while,

- meeting the timing and physical design requirements,
- resolving signal integrity (SI) problems,
- and considering on-chip variation (OCV).

The flow also takes advantage of the Innovus software multiple-CPU processing capabilities to accelerate the design process and, where appropriate, runs features in the multi-threading or distributed processing mode. As input, you provide a Common Power Format (CPF) file that captures design and technology-related power constraints. CPF lets you define those rules that Innovus implements for those objects that the low power design uses. These objects could be:

level shifters, isolation cells, power domains, and power switches. CPF also includes the information Innovus needs for multi-mode multi-corner (MMMC) timing analysis. With the CPF-Based Low Power Implementation Flow design, you do not need Innovus commands to define views, corners, and modes for MMMC.

For the main design tasks the flow uses the Innovus supercommands with as few nondefault options as possible.

For a block or a flat chip, these commands are the starting point for the low power design. Ultimately, each design that you implement might have a different final set of commands, but this flow is the recommended starting point before customizing a specific design or technology. When you have completed the tasks described in this guide, you will have a flat MSV design that is ready for DRC and LVS checks and other sign-off tasks.

Flow Overview of CPF-based Low Power Flow

The CPF-Based low power implementation flow starts with a completed floorplan (you also have the flexibility to implement floorplanning tasks through plug-in called in the flow). At the end of each main task, there is an optimization step, and the task is complete when violations are fixed and the timing is acceptable. The CPF-Based Low Power flow is complete and the design is ready for sign-off tasks when the design meets the timing, the CPF-based low power checks are in place, and the physical design requirements are met.
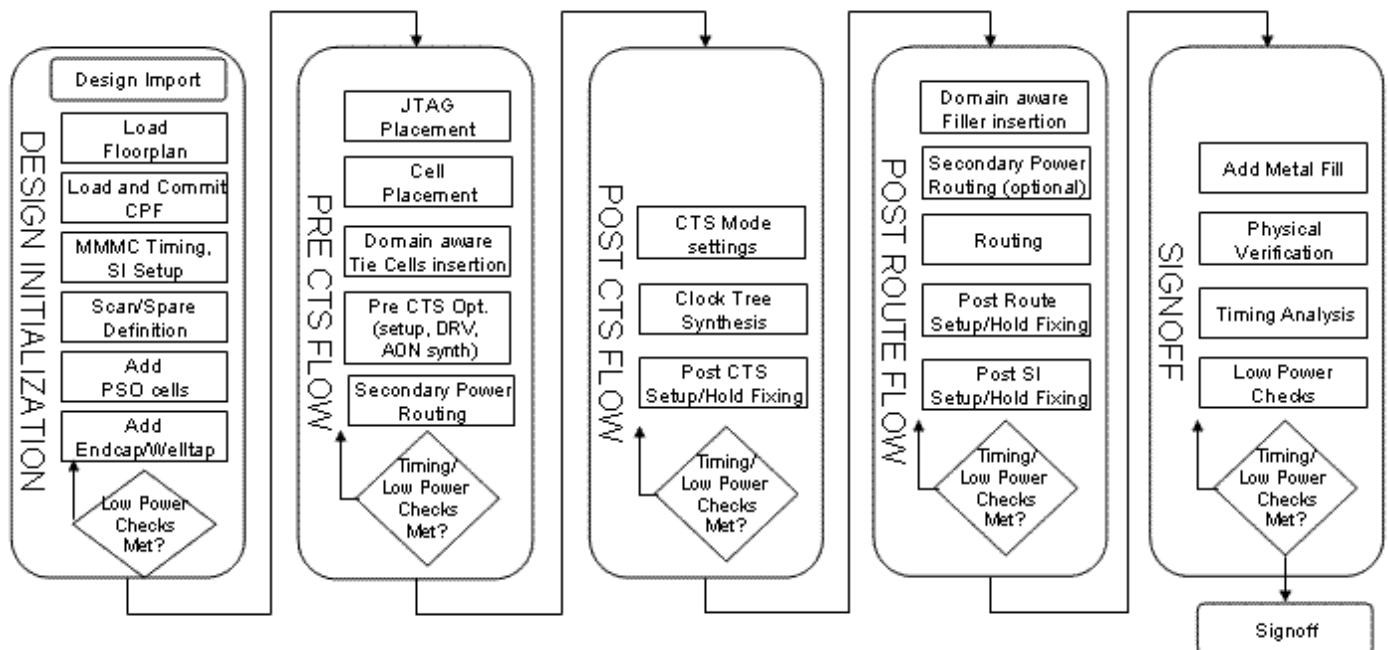
Before you Begin Implementing CPF-based Low Power Flow

The CPF-Based low power implementation flow assumes that timing and routing constraints are feasible. Make sure the following tasks are complete before starting this flow:

- Floorplanning. The information is optional. If you have the floorplan file, you can use it as an input, or you can perform the floorplanning tasks through the plug-ins in the flow.

  - Hard block placement

  - Region definition

  - I/O pad placement

  - Creation of placement and routing blockages

- Creation of the clock tree specification file

- Creation of RC scale factors

- Creation of CPF file

- Layer map file for the QRC and GDS export

- Config. file for the QRC extraction

For more information, see the following sections of the *Innovus User Guide*

- In the *Data Preparation* chapter *Data Preparation*

- In the *Synthesizing Clock Trees* chapter *Synthesizing Clock Trees*

- In the *RC Extraction* chapter *RC Extraction*

Input for CPF-based Low Power Flow

The following list specifies the input for the CPF-Based low power implementation flow:

- Innovus configuration file (setup.tcl) with the following information specified:

  - Timing libraries (*.lib)

  - LEF libraries (*.lef)

  - Timing constraints (*.sdc)

- ○ Capacitance table or QRC technology file

- ○ SI libraries (*.cdb or *.udn) (recommended but not required)

- Verilog netlist (*.v)

- Floorplan file (*.fp or *.def)

- Clock tree specification file

- Scan chain information (*.tcl or *.def)

- GDS Layer map file (*.gds)

- CPF files (*.cpf)

For more information, see Data Preparation chapter in the Innovus User Guide

Additional Input

In addition, depending on the design and technology, you may need to supply values for the following items:

- RC scaling factors

- OCV derating factors

- Metal fill parameters

- Filler cell names

- Tie cell names

- Welltap cells

- Endcap cells

- Secondary power cell:pin pair

- Clock gating cell names

- Spare instance names

- JTAG instance names

- JTAG rows

- LSF queue

- Dont Use Cells

- Delay Cells

- CTS Cells

See also,

Defining Variables and Specifying Values for Command Modes

Creating Variables for CPF-based Low Power Flow

Results for CPF-based Low Power Flow

Example - CPF-Based Low Power Foundation Flow

Defining Variables and Specifying Values for Command Modes

The CPF-Based Low Power implementation flow uses a set of Tcl scripts that defines the flow variables and specifies the timing environment, command modes, and metal fill rules. All of the Innovus Foundation Flows rely on these scripts to keep the flows clean and easy to manage. The scripts are sourced from the main flow script.

- Describing lp_config.tcl
  This file is unique for each low power design. For Low Power design, the user must edit this file along with the setup.tcl and edi_config.tcl. This file contains Low Power variables such as:

    - TIE cells for PD

    - FILLER cells for PD

    - ENDCAP and WELLTAP implementation

    - Secondary PG Routing

    - AON Net synthesis

    - Modify Power Domains (size, clearance, etc)

    - Power switch planning
      In this section we also cover:

- Timing Environment Initialization

- MMMC Timing and SI Setup for Low Power P_R Implementation

See also,

- [Creating Variables for CPF-based Low Power Flow](#)

- [Results for CPF-based Low Power Flow](#)

- [Example - CPF-Based Low Power Foundation Flow](#)

MMMC Timing and SI Setup for Low Power

Describing lp_config.tcl

Describing lp_config.tcl

Low power configuration file overlay. This file contains foundation flow variables that are specific to the LP/CPF flow and should be used in addition to the `setup.tcl` and `edi_config.tcl`.

```
set vars(cpf_keep_rows) true | false
```

The `vars(power_domains)` is optional. If not defined, the power domain list will be picked up automatically.

```
set vars(power_domains) power domain list
```

TIE cell information
The variable `vars(tie_cells)` is defined in the `setup.tcl` and is used to define a *global* tie cell list. This list will be used by default for each power domain. It can be overridden for a given power domain by setting:

```
set {{vars(power domain,tie_cells)}} <tie cells for power domain>
```

FILLER cell information
The variable `vars(filler_cells)` is defined in the setup.tcl and is used to define a *global* filler cell list. This list will be used by default for each power domain. It can be overridden for a given power domain by setting:

```
set vars(power domain,filler_cells) <fillers cells for <power domain>>
```

Welltap cell information
The variable `vars(welltaps)` is defined in the setup.tcl and is used to define a *global* welltap cell list. This list will be used by default for each power domain. It can be overridden for a given power domain by setting:

```
set vars(power_domain,welltaps) <welltap cell list for power_domain>
set vars(power_domain, welltaps,checkerboard) true or false
set vars(power_domain,welltaps,max_gap) max gap in microns
set vars(power_domain,welltaps,cell_interval) cell interval in microns
set vars(power_domain,welltaps,row_offset) row offset in microns
set vars(power_domain, welltaps,verify_rule) verify rule distance
```

Endcap cell information

The variables `vars(pre_endcap)` and `vars(post_endcap)` are set in the setup.tcl file and are used to define *global* endcap cells. These used by default for each power domain. They can be overridden for a given power domain setting:

```
set vars(power_domain,pre_endcap)  <pre endcap cell for power_domain>
set vars(power_domain,post_endcap) <post endcap cell for power_domain>
```

Always on net buffering

By default SOCE does always-on-net synthesis for SRPG control signal and PSO enable signals as part of `optDesign -preCTS`, but this can be manually done for specific nets if necessary. To do this, define the following variables and uncomment the buffer_always_on_nets proc.

```
set vars(always_on_buffers) list of always on buffers
set vars(always_on_nets) list of always on nets
set vars(always_on_nets,max_fanout) max fanout limit for always on nets
set vars(always_on_nets,max_tran) max transition on always on nets
set vars(always_on_nets,max_skew) max skew for always on nets
set vars(always_on_nets,max_delay) max delay for always on nets
set vars(power_domain,always_on_buffers) <buffers for power domain>
```

Secondary power/ground routing

Automatic secondary power routing can be enabled during the foundation flows by setting `vars(route_secondary_pg_nets)` to true and providing cell pin pair information to identify the connections requiring routing (Example: AONBUFFD1:TVDD LSL2HCD4:VDDL).

```
set vars(route_secondary_pg_nets) [true | false]
set vars(secondary_pg,cell_pin_pairs) secondary power cell pin pair list
```

In addition, the following can optionally defined either globally or per p/g net

```
set vars(secondary_pg,max_fanout) max fanout for secondary power routing
set vars(secondary_pg,pattern) secondary power routing pattern trunk | steiner
set vars(secondary_pg,non_default_rule) non-default rule for secondary p/g/
routing
```

To optionally override for a given p/g net(s), use `vars(route_secondary_pg_nets)` to define the list of nets to be overridden and then override `vars(<p/g net>,<option>)`

```
set vars(secondary_pg,nets) list of power/ground nets
set vars(p/g_net,max_fanout) max fanout
set vars(p/g_net,pattern) trunk | steiner
set vars(p/g_net,non_default_rule) non default rule
```

runCLP options

```
set vars(clp_options) options for runCLP
Ex: set vars(clp_options)  " -cmd ../DATA/chip_top_backend_clp.do extraVlog\../DATA/dummy.v"
```

Modify power domain
The foundation flow contains set of utilities in `utils.tcl`, it has modify_power_domains procedure for implementing the power domain bonding box clearance surrounding to the domain and route search extensions. For doing power domain modifications user has to define below variables and can call `modify_power_domains` procedure through plug-in:

```
set vars(power_domain,bbox) llx lly urx ury; bondary coordinates in microns
set vars(power_domain,rs_exts) top bot left right; distance in microns
set vars(power_domain,min_gaps) top bot left right; distance in microns
```

Power Shut-off Planning
The foundation flow contains set of utilities in utils.tcl, it has add_power_switches procedure for implementing the power switch insertion. User has to define below variables and can call add_power_switches procedure through plug-in:

```
set vars(power_domain,switchable) true | false
set vars(power_domain,switch_type) column | ring
set vars(power_domain,switch_cell) PSO cell name
set vars(power_domain,input_enable_pin) PSO cell input enable pin
set vars(power_domain,output_enable_pin) PSO cell output enable pin
set vars(power_domain,input_enable_net) PSO cell input enable net
set vars(power_domain,output_enable_net) PSO cell output enable net
set vars(power_domain,switch_instance) switchModuleInstance
set vars(power_domain,top_offset) top offset in microns
set vars(power_domain,bottom_offset) bottom offset in microns
set vars(power_domain,right_offset) right offset in microns
set vars(power_domain,left_offset) left offset in microns
```

Below variables are for column based PSO implementation

```
set vars(power_domain,checker_board) true | false
set vars(power_domain,horizonal_pitch) in microns
set vars(power_domain,column_height) Switch cell column height in microns
set vars(power_domain,skip_rows) Number of rows to skip
set vars(power_domain,back_to_back_chain) true|false
```

Connects the enableNetOut at the top of a column to the enableNetIn at the top of the next column, and connects the enableNetOut at the bottom of the column to the enableNetIn at the bottom of the next column. Below variables are for ring based PSO implementation:

```
set vars(power_domain,top_ring) 1|0
set vars(power_domain,bottom_ring) 1|0
set vars(power_domain,right_ring) 1|0
set vars(power_domain,left_ring) 1|0
```

... defines which side of the power domain to insert switches:

```
set vars(power_domain,top_switch_cell) pso cell name
set vars(power_domain,bottom_switch_cell) pso cell name
set vars(power_domain,left_switch_cell) pso cell name
set vars(power_domain,right_switch_cell) pso cell name
```

... define pso cell name for each side of the power domain

```
set vars(power_domain,top_filler_cell) filler cell name
set vars(power_domain,bottom_filler_cell) filler cell name
set vars(power_domain,left_filler_cell) filler cell name
set vars(power_domain,right_filler_cell) filler cell name
set vars(power_domain,corner_cell_list) corner cell name
```

... define filler cell name for each side of the power domain

```
set vars(power_domain,top_switches)      <"number -distribute">
set vars(power_domain,bottom_switches)   <"number -distribute">
set vars(power_domain,left_switches)     <"number -distribute">
set vars(power_domain,right_switches)    <"number -distribute">
```

... define the number of switches for each side of the power domain

See also,

Timing Environment Initialization

MMMC Timing and SI Setup for Low Power P_R Implementation

Timing Environment Initialization

This is done through the procedure `ff_initialize_timing` {minmax | mmmc} defined inside `Innovus/procs.tcl`. Depending on the variation of the flow you are using default, postroute MMMC, or MMMC the timing environment is initialized to either minmax or mmmc timing. The MMMC timing environment is required for accurate sign-off timing and is enabled as follows:

- In the default flow, it is enabled for the final timing analysis step.

- In the postroute MMMC flow, it is enabled after routing.

- In the MMMC flow, it is enabled throughout the flow.

See also,

Describing lp_config.tcl

MMMC Timing and SI Setup for Low Power P_R Implementation

MMMC Timing and SI Setup for Low Power P_R Implementation

MMMC Timing and SI Setup for Low Power P&R Implementation

Create RC Corners

```
create_rc_corner –name $rc_corner  -cap_table\
$vars($rc_corner,cap_table) \




–preRoute_res $vars($rc_corner,pre_route_res_factor)\


–preRoute_cap $vars($rc_corner,pre_route_cap_factor)\


–preRoute_clkres\ $vars($rc_corner,pre_route_clk_res_factor)\


–preRoute_clkcap\ $vars($rc_corner,pre_route_clk_cap_factor)\


–postRoute_res\ $vars($rc_corner,post_route_res_factor)\


–postRoute_cap\ $vars($rc_corner,post_route_cap_factor)\


–postRoute_clkres\ $vars($rc_corner,post_route_clk_res_factor)\


–postRoute_clkcap\ $vars($rc_corner,post_route_clk_cap_factor)\


–postRoute_xcap\ $vars($rc_corner,post_route_xcap_factor)\


–T $vars($rc_corner,T)\


-qx_tech_file $vars($rc_corner, qx_tech_file)\


-qx_conf_file $vars($rc_corner, qx_conf_file) }
```

Note: If the low power P&R implementation is not CPF-based or the CPF file does not have its analysis-views defined, then the user has to create library sets, delay corners, constraint mode and analysis views.

Please refer to the following section in this guide for these:

- Describing setup.tcl

- Describing edi_config.tcl

- Example Settings for Each Script

See also,

- Describing lp_config.tcl

- Timing Environment Initialization

Creating Variables for CPF-based Low Power Flow

Creating Variables for CPF-based Low Power Flow

This section covers the following:

- Create Library Sets
- Create Delay Corners
- Create Constraint Modes
- Create Analysis Views
- Attach RC Corners to Delay Corners
- Update library_sets with SI Libraries, if defined
- Define setup and hold Analysis Views and Enable the Default Views

See also,

Defining Variables and Specifying Values for Command Modes

Results for CPF-based Low Power Flow

Example - CPF-Based Low Power Foundation Flow

Create Library Sets

```
foreach library_set $vars(library_sets) {

    if {[info exists vars($library_set,si)] && ($vars($library_set,si) != "")} {

        create_library_set -name $library_set \

                        -timing $vars($library_set,timing) \
```

```
                               -si $vars($library_set,si)


        } else {


            create_library_set -name $library_set -timing $vars($library_set,timing)


        }


    }
```

See also,

Create Delay Corners

Create Constraint Modes

Create Analysis Views

Attach RC Corners to Delay Corners

Update library_sets with SI Libraries, if defined

Define setup and hold Analysis Views and Enable the Default Views

Create Delay Corners

Create Delay Corners

```
foreach delay_corner $vars(delay_corners) {


    set command "create_delay_corner -name $delay_corner "


    append command "-library_set $vars($delay_corner,library_set) "


    append command "-rc_corner $vars($delay_corner,rc_corner) "


    if {[info exists vars($delay_corner,opcond)] && [info exists \
vars($delay_corner,opcond_library)]} {
```

```
          append command "-opcond_library $vars($delay_corner,opcond_library) "

          append command "-opcond $vars($delay_corner,opcond) "

     }

     eval $command

}
```

See also,

[Create Library Sets](#)

[Create Constraint Modes](#)

[Create Analysis Views](#)

[Attach RC Corners to Delay Corners](#)

[Update library_sets with SI Libraries, if defined](#)

[Define setup and hold Analysis Views and Enable the Default Views](#)

Create Constraint Modes

```
Puts "<FF> Creating constraint modes: $vars(constraint_modes)..."

   foreach constraint_mode $vars(constraint_modes) {

      create_constraint_mode -name $constraint_mode \

                      -sdc_files $vars($constraint_mode,pre_cts_sdc)

   }
```

See also,

[Create Library Sets](#)

Create Delay Corners

Create Analysis Views

Attach RC Corners to Delay Corners

Update library_sets with SI Libraries, if defined

Define setup and hold Analysis Views and Enable the Default Views

Create Analysis Views

```
set vars(all_analysis_views) [concat $vars(setup_analysis_views) \
$vars(hold_analysis_views)]


Puts "<FF> Creating analysis views: $vars(all_analysis_views) ..."


foreach analysis_view $vars(all_analysis_views) {

if {[lsearch [all_analysis_views] $analysis_view] == -1} {

        create_analysis_view -name $analysis_view \


                        -constraint_mode \ $vars($analysis_view,constraint_mode) \


                        -delay_corner $vars($analysis_view,delay_corner)


    }


  }
```

Note: The steps mentioned below are common for CPF and non-CPF based low power P&R implementation.

See also,

Create Library Sets

Create Delay Corners

Create Constraint Modes

Attach RC Corners to Delay Corners

Update library_sets with SI Libraries, if defined

Define setup and hold Analysis Views and Enable the Default Views

Attach RC Corners to Delay Corners

```
foreach corner $vars(delay_corners) {


    Puts "Updating delay corner $corner with RC Corner $vars($corner,rc_corner)\
..."


    update_delay_corner -name $corner -rc_corner $vars($corner,rc_corner)


}
```

See also,

Create Library Sets

Create Delay Corners

Create Constraint Modes

Create Analysis Views

Update library_sets with SI Libraries, if defined

Define setup and hold Analysis Views and Enable the Default Views

Update library_sets with SI Libraries, if defined

```
foreach library_set $vars(library_sets) {

if {[info exists vars($library_set,si)] && ($vars($library_set,si) != "")} {

update_library_set -name $library_set \

-si $vars($library_set,si)

}
```

}

Set derating factors for delay corners and enable CPPR. CPPR should only be enabled for designs using derating or designs with specific requirements

```
set_timing_derate -delay_corner $corner -data -cell_delay -late \
$vars($corner,data_cell_late)


set_timing_derate -delay_corner $corner -data -cell_delay -early \
$vars($corner,data_cell_early)


set_timing_derate -delay_corner $corner -data -net_delay -late \
$vars($corner,data_net_late)


set_timing_derate -delay_corner $corner -data -net_delay -early \
$vars($corner,data_net_early)


set_timing_derate -delay_corner $corner -clock -cell_delay -late \
$vars($corner,clock_cell_late)


set_timing_derate -delay_corner $corner -clock -cell_delay -early \
$vars($corner,clock_cell_early)


set_timing_derate -delay_corner $corner -clock -net_delay -late \
$vars($corner,clock_net_late)


set_timing_derate -delay_corner $corner -clock -net_delay -early \
$vars($corner,clock_net_early)
```

See also,

Create Library Sets

Create Delay Corners

Create Constraint Modes

Create Analysis Views

Attach RC Corners to Delay Corners

Define setup and hold Analysis Views and Enable the Default Views

Define setup and hold Analysis Views and Enable the Default Views

Define setup/hold Analysis Views and Enable the Default Views

```
set_analysis_view -setup $vars(active_setup_views) -hold $vars(active_hold_views)
```

```
set_default_view -setup $vars(default_setup_view) -hold $vars(default_hold_view)
```

See also,

Create Library Sets

Create Delay Corners

Create Constraint Modes

Create Analysis Views

Attach RC Corners to Delay Corners

Update library_sets with SI Libraries, if defined

Results for CPF-based Low Power Flow

The design is now ready for sign-off physical and timing verification. The output would be:

- Design data
    - Verilog netlist
    - SPEF file
    - GDS file
    - Innovus log file
    - Command file
- Reports
    - Timing reports

- Clock tree reports

- Verify connectivity

- Verify DRC

- Verify metal density

- Verify process antenna

- Power report

- Conformal Low Power report

See also,

Defining Variables and Specifying Values for Command Modes

Creating Variables for CPF-based Low Power Flow

Example - CPF-Based Low Power Foundation Flow

Example - CPF-Based Low Power Foundation Flow

Plug-in scripts usage in the reference design:

The following (optional) plug-ins are supported by the flow. They allow for design specific requirements to be included but allow for flow scripts to remain unchanged. To enable a plug-in, simply define the appropriate plug-in variable in the setup.tcl, and it will be sourced automatically in the flow at the appropriate time.

Plugin	Description
`always_source_tcl`	Sourced at the beginning of every script
`pre_partition_tcl`	Before partitioning
`post_partition`	After partitioning
`pre_init_tcl`	Before design initialization

`post_init_tcl`	After design initialization
`pre_place_tcl`	Before `place_opt_design`
`place_tcl`	Replaces call to `place_opt_design`
`post_place_tcl`	After `place_opt_design`
`pre_cts_tcl`	Before `ccopt_design`
`cts_tcl`	Replaces call to `ccopt_design`
`post_cts_tcl`	After `ccopt_design`
`pre_route_tcl`	Before routeDesign
`post_route_tcl`	After routeDesign
`pre_si_tcl`	Before SI fixing
`post_si_tcl`	After SI fixing
`pre_signoff_tcl`	Before final STA
`post_signoff_tcl`	After final STA

The sections below describe some of the plug-in enabled in the reference design.

See also,

Defining Variables and Specifying Values for Command Modes

Creating Variables for CPF-based Low Power Flow

Results for CPF-based Low Power Flow

CPF File_3a

CPF File:

Define Library Sets and Low Power Cells

set_cpf_version 1.0e

```
define_library_set –name 0v864 –libraries " \

../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtwc0d720d72.lib \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtwc0d720d72.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpwc0d720d72.lib \

../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtwc0d72.lib \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtwc0d72.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpwc0d72.lib \

../LIBS/STDCELL/LIBERTY/HVT_CG/tcbn65lphvtcgwc0d72.lib \

../LIBS/STDCELL/LIBERTY/LVT_CG/tcbn65lplvtcgwc0d72.lib \

../LIBS/STDCELL/LIBERTY/NVT_CG/tcbn65lpcgwc0d72.lib \

../LIBS/RAMS_ROMS/LIBERTY/WORST_0864/ulp_dp_512x36cm4sw1bk1.lib \

../LIBS/RAMS_ROMS/LIBERTY/WORST_0864/ulp_rom_512x32cm16.lib \

../LIBS/RAMS_ROMS/LIBERTY/WORST_0864/ulp_sp_2kx32cm8sw1bk2.lib \
```

```
../LIBS/RAMS_ROMS/LIBERTY/WORST_0864/ulp_sp_4kx32cm8sw1bk4.lib \

../LIBS/IO/LIBERTY/tpzn65lpgv2wc_0864.lib \

../LIBS/ANALOG/LVDS/LIBERTY/lvds_wc_0864.lib \

../LIBS/IP/LIBERTY/pso_wc_0864.lib \

../LIBS/IP/LIBERTY/Module_A_wc_0864.lib \

../LIBS/IP/LIBERTY/Module_B_wc_0864.lib \

../LIBS/IP/LIBERTY/Module_U_wc_0864.lib \

../LIBS/IP/LIBERTY/Module_W_wc_0864.lib \

"


define_library_set -name 1v080 -libraries  \

../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtwc0d720d9.lib \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtwc0d720d9.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpwc0d720d9.lib \

../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtwc0d90d9.lib \

../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtwc.lib \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtwc.lib \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtwc0d90d9.lib \
```

```
../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpwc0d90d9.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpwc.lib \

../LIBS/STDCELL/LIBERTY/HVT_CG/tcbn65lphvtcgwc.lib \

../LIBS/STDCELL/LIBERTY/LVT_CG/tcbn65lplvtcgwc.lib \

../LIBS/STDCELL/LIBERTY/NVT_CG/tcbn65lpcgwc.lib \

../LIBS/RAMS_ROMS/LIBERTY/WORST_1080/ulp_dp_512x36cm4sw1bk1.lib \

../LIBS/RAMS_ROMS/LIBERTY/WORST_1080/ulp_rom_512x32cm16.lib \

../LIBS/RAMS_ROMS/LIBERTY/WORST_1080/ulp_sp_2kx32cm8sw1bk2.lib \

../LIBS/RAMS_ROMS/LIBERTY/WORST_1080/ulp_sp_4kx32cm8sw1bk4.lib \

../LIBS/IO/LIBERTY/tpzn65lpgv2wc.lib \

../LIBS/ANALOG/PLL/LIBERTY/PL0042.scx3_tsmc_cln80gt_hvt_ss_1p08v_125c.lib \

../LIBS/IP/LIBERTY/pso_wc_1080.lib \

../LIBS/IP/LIBERTY/Module_A_wc_1080.lib \

../LIBS/IP/LIBERTY/Module_B_wc_1080.lib \

../LIBS/IP/LIBERTY/Module_U_wc_1080.lib \



define_library_set -name 1v320 -libraries " \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtbc0d881d1.lib \
```

```
../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtbc0d881d1.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpbc0d881d1.lib \

../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtbc1d11d1.lib \

../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtbc.lib \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtbc.lib \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtbc1d11d1.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpbc.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpbc1d11d1.lib \

../LIBS/STDCELL/LIBERTY/HVT_CG/tcbn65lphvtcgbc.lib \

../LIBS/STDCELL/LIBERTY/LVT_CG/tcbn65lplvtcgbc.lib \

../LIBS/STDCELL/LIBERTY/NVT_CG/tcbn65lpcgbc.lib \

../LIBS/RAMS_ROMS/LIBERTY/BEST_1320/ulp_dp_512x36cm4sw1bk1.lib \

../LIBS/RAMS_ROMS/LIBERTY/BEST_1320/ulp_rom_512x32cm16.lib \

../LIBS/RAMS_ROMS/LIBERTY/BEST_1320/ulp_sp_2kx32cm8sw1bk2.lib \

../LIBS/RAMS_ROMS/LIBERTY/BEST_1320/ulp_sp_4kx32cm8sw1bk4.lib \

../LIBS/IO/LIBERTY/tpzn65lpgv2bc.lib \

../LIBS/ANALOG/PLL/LIBERTY/PL0042.scx3_tsmc_cln80gt_hvt_ff_1p32v_0c.lib \

../LIBS/IP/LIBERTY/pso_bc_1320.lib   \
```

```
../LIBS/IP/LIBERTY/Module_A_bc_1320.lib \

../LIBS/IP/LIBERTY/Module_B_bc_1320.lib \

../LIBS/IP/LIBERTY/Module_U_bc_1320.lib \

"

define_library_set -name 1v056 -libraries " \

../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtbc0d880d88.lib \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtbc0d880d88.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpbc0d880d88.lib \

../LIBS/STDCELL/LIBERTY/HVT/tcbn65lphvtbc0d88.lib \

../LIBS/STDCELL/LIBERTY/LVT/tcbn65lplvtbc0d88.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpbc0d88.lib \

../LIBS/STDCELL/LIBERTY/HVT_CG/tcbn65lphvtcgbc0d88.lib \

../LIBS/STDCELL/LIBERTY/LVT_CG/tcbn65lplvtcgbc0d88.lib \

../LIBS/STDCELL/LIBERTY/NVT_CG/tcbn65lpcgbc0d88.lib \

../LIBS/RAMS_ROMS/LIBERTY/BEST_1056/ulp_dp_512x36cm4sw1bk1.lib \

../LIBS/RAMS_ROMS/LIBERTY/BEST_1056/ulp_rom_512x32cm16.lib \

../LIBS/RAMS_ROMS/LIBERTY/BEST_1056/ulp_sp_2kx32cm8sw1bk2.lib \

../LIBS/RAMS_ROMS/LIBERTY/BEST_1056/ulp_sp_4kx32cm8sw1bk4.lib \
```

```
../LIBS/IO/LIBERTY/tpzn65lpgv2bc_1056.lib \

../LIBS/ANALOG/LVDS/LIBERTY/lvds_bc_1056.lib \

../LIBS/IP/LIBERTY/pso_bc_1056.lib   \

../LIBS/IP/LIBERTY/Module_A_bc_1056.lib \

../LIBS/IP/LIBERTY/Module_B_bc_1056.lib \

../LIBS/IP/LIBERTY/Module_U_bc_1056.lib \

../LIBS/IP/LIBERTY/Module_W_bc_1056.lib \

"

define_library_set -name 2v5 -libraries " \

../LIBS/IP/LIBERTY/Module_U.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpwc.lib \

"



define_library_set -name 3v3 -libraries " \

../LIBS/IP/LIBERTY/Module_U.lib \

../LIBS/STDCELL/LIBERTY/NVT/tcbn65lpwc.lib \

"

define_always_on_cell -cells { PTBUFFD8 PTBUFFD4 PTBUFFD2 PTBUFFD1 \
```

```
PTBUFFD8HVT PTBUFFD4HVT PTBUFFD2HVT PTBUFFD1HVT \

PTBUFFD8LVT PTBUFFD4LVT PTBUFFD2LVT PTBUFFD1LVT } \

-power_switchable VDD -power TVDD -ground VSS


define_isolation_cell -cells { ISOLOD8 ISOLOD4 ISOLOD2 ISOLOD1 ISOHID8 ISOHID4 \

ISOHID2 ISOHID1 ISOLOD8LVT ISOLOD4LVT ISOLOD2LVT ISOLOD1LVT ISOHID8LVT \

ISOHID4LVT ISOHID2LVT ISOHID1LVT ISOLOD8HVT ISOLOD4HVT ISOLOD2HVT \

ISOLOD1HVT ISOHID8HVT ISOHID4HVT ISOHID2HVT ISOHID1HVT } -power VDD \

-enable   ISO -valid_location to


define_isolation_cell -cells { LVLLHCD8LVT LVLLHCD4LVT LVLLHCD2LVT LVLLHCD1LVT \

LVLLHCD8 LVLLHCD4 LVLLHCD2 LVLLHCD1 LVLLHCD8HVT \

LVLLHCD4HVT LVLLHCD2HVT LVLLHCD1HVT } -power VDD -ground VSS -enable \

NSLEEP -valid_location to


define_level_shifter_cell -cells { LVLLHCD8LVT LVLLHCD4LVT LVLLHCD2LVT \

LVLLHCD1LVT LVLLHCD8 LVLLHCD4 LVLLHCD2 LVLLHCD1 \

LVLLHCD8HVT LVLLHCD4HVT LVLLHCD2HVT LVLLHCD1HVT } -input_voltage_range \

0.864:1.08:0.216 -output_voltage_range 0.864:1.08:0.216 -direction up \
```

```
-input_power_pin VDDL -output_power_pin VDD -ground VSS -enable NSLEEP \

-valid_location to


define_level_shifter_cell -cells { LVLLHD8LVT LVLLHD4LVT LVLLHD2LVT LVLLHD1LVT \

LVLLHD8 LVLLHD4 LVLLHD2 LVLLHD1 LVLLHD8HVT LVLLHD4HVT \

LVLLHD2HVT LVLLHD1HVT } -input_voltage_range 0.864:1.08:0.216 \

-output_voltage_range 0.864:1.08:0.216 -direction up -input_power_pin VDDL \

-output_power_pin VDD -ground VSS -valid_location to


define_power_switch_cell -cells { CDN_RING_SW HDRSID2 HDRSID0 } \

-stage_1_enable NSLEEPIN -stage_1_output NSLEEPOUT -type header \

-power_switchable VDD -power TVDD


define_state_retention_cell -cells { RSDFCSD4 RSDFCSD2 RSDFCSD1 RSDFCRD4 \

RSDFCRD2 RSDFCRD1 RSDFCD4 RSDFCD2 RSDFCD1 RSDFCSD4LVT RSDFCSD2LVT \

RSDFCSD1LVT RSDFCRD4LVT RSDFCRD2LVT RSDFCRD1LVT \

RSDFCD4LVT   RSDFCD2LVT RSDFCD1LVT RSDFCSD4HVT \

RSDFCSD2HVT RSDFCSD1HVT RSDFCRD4HVT RSDFCRD2HVT \

RSDFCRD1HVT RSDFCD4HVT RSDFCD2HVT RSDFCD1HVT } -clock_pin CP \
```

```
-restore_function !NRESTORE -save_function SAVE \

-power_switchable VDD -power TVDD -ground VSS

set_hierarchy_separator /

set_design chip_top
```

Macro Models

```
set_macro_model Module_U

create_nominal_condition -name norm_1v080 -voltage 1.08

create_nominal_condition -name norm_0v864 -voltage 0.864

create_nominal_condition -name ana_2p5 -voltage 2.5

create_nominal_condition -name ana_3p3 -voltage 3.3

create_nominal_condition -name off -voltage 0

create_power_domain -name PD_3p3v

create_power_domain -name PD_2p5v

create_power_domain -name PDdvfs -boundary_ports { Module_U_1* Module_U_2* \

Module_U_3* Module_U_4* Module_U_5* Module_U_6* Module_U_7* Module_U_8* \
```

```
Module_U_9* Module_U_10* Module_U_11* Module_U_12* Module_U_13* \

Module_U_14* Module_U_15* Module_U_16* Module_U_17* Module_U_18* \

Module_U_19* Module_U_20* Module_U_21* Module_U_22* Module_U_23* \

Module_U_24* Module_U_25* dp dn rrefext Module_U_29* Module_U_30* } \

-default -external_controlled_shutoff


create_power_mode -name PMdvfs_mode1 -default -domain_conditions { \

PD_3p3v@ana_3p3 PD_2p5v@ana_2p5 PDdvfs@norm_1v080 }

create_power_mode -name PMdvfs_mode2 -domain_conditions { PD_3p3v@ana_3p3 \

PD_2p5v@ana_2p5 PDdvfs@norm_0v864 }

create_power_mode -name PM_socoff -domain_conditions { PD_3p3v@ana_3p3 \

PD_2p5v@ana_2p5 PDdvfs@off }


update_power_domain -name PD_3p3v -primary_power_net vdda33 \

-primary_ground_net vssa

update_power_domain -name PD_2p5v -primary_power_net vdda25 \

-primary_ground_net vssa

update_power_domain -name PDdvfs -primary_power_net VDDdvfs \

-primary_ground_net VSSsoc
```

```
end_macro_model
```

Create Power Domains

```
create_power_domain -name PDdvfs -boundary_ports { GPIO_pad[*] SMC_addr_pad[*] \

SMC_data_pad[*] SMC_n_CS_pad[*] SMC_n_be_pad[*] SMC_n_we_pad[*] \

TTC_ext_clk_pad[*] TTC_wfrm_pad[*] spi_N_ss_out_pad[*] } -default \

-external_controlled_shutoff -shutoff_condition !power_on_pin

create_power_domain -name PDpll -instances { i_pll_controller i_PLL \

i_pad_frame/i_pin_12M i_pad_frame/i_pll_clk  i_pad_frame/*PLL_AGND1_H \

i_pad_frame/*PLL_AVDD1_H \

i_pad_frame/*PLL_DGND_i* \

i_pad_frame/*PLL_DVDD_i* \

i_pad_frame/*PLL_AGND0_H \

i_pad_frame/*PLL_AVDD0_H \

*FILLER_PDpll_*} -boundary_ports { \

pin_12M_pad pin_pllclk_pad } -external_controlled_shutoff \

-shutoff_condition !power_on_pin

create_power_domain -name PDw -instances { i_Module_W i_pad_frame/i_lvds \
```

```
i_pad_frame/i_rf_rstn i_pad_frame/i_rf_EN i_pad_frame/rf_SW[0].i_rf_SW \

i_pad_frame/rf_SW[1].i_rf_SW i_pad_frame/rf_SW[2].i_rf_SW \

i_pad_frame/rf_SW[3].i_rf_SW i_pad_frame/*VDDw_i*} -boundary_ports { hissrxip hissrxin \

hissclkp hissclkn hissrxqp hissrxqn rf_resetn_pad rf_en_pad hisstxip \

hisstxin hisstxqp hisstxqn rf_sw_pad[*] } -external_controlled_shutoff \

-shutoff_condition !power_on_pin

create_power_domain -name PDrom -instances { \

i_rom_subsystem/i_rom_wrap/i_rom_core } -shutoff_condition \

!i_apb_subsystem/i_power_ctrl/pwr1_on_rom -secondary_domains { PDdvfs }

create_power_domain -name PDsmc -instances { i_apb_subsystem/i_smc *FILLER_PDsmc_*} \

-shutoff_condition !i_apb_subsystem/i_power_ctrl/pwr1_on_smc \

-secondary_domains { PDdvfs }

create_power_domain -name PDwakeup -instances { i_wakeup \

i_pad_frame/i_shift_en i_pad_frame/i_scan_mode i_pad_frame/i_pin_rst \

i_pad_frame/i_pin_32K i_pad_frame/i_ext_wakeup i_pad_frame/i_wkup_done \

i_pad_frame/i_power_on i_pad_frame/*VDDwakeup_i* *FILLER_PDwakeup_*} -boundary_ports {
scan_mode_pad pin_32k_pad \

pin_reset_pad ext_wakeup_pin wkup_done_pin power_on_pin shift_en_pad }

create_power_domain -name PD_3p3v
```

```
create_power_domain -name PD_2p5v

create_power_domain -name i_apb_subsystem__i_uart1__PDuart_SW -instances { \

i_apb_subsystem/i_uart1 *FILLER_i_apb_subsystem__i_uart1__PDuart_SW_*} -shutoff_condition \

!i_apb_subsystem/i_power_ctrl/pwr1_on_urt -secondary_domains { PDdvfs }


update_power_domain -name PDdvfs -primary_power_net VDDdvfs \

-primary_ground_net VSSsoc

update_power_domain -name PDpll -primary_power_net VDDpll -primary_ground_net \

VSSsoc

update_power_domain -name PDw -primary_power_net VDDw -primary_ground_net \

VSSsoc

update_power_domain -name PDrom -primary_power_net VDDrom -primary_ground_net \

VSSsoc

update_power_domain -name PDsmc -primary_power_net VDDsmc -primary_ground_net \

VSSsoc

update_power_domain -name PDwakeup -primary_power_net VDDwake \

-primary_ground_net VSSsoc

update_power_domain -name PD_3p3v -primary_power_net vdda33 \
```

```
-primary_ground_net vssa

update_power_domain -name PD_2p5v -primary_power_net vdda25 \

-primary_ground_net vssa

update_power_domain -name i_apb_subsystem__i_uart1__PDuart_SW -primary_power_net \

VDDuart -primary_ground_net VSSsoc
```

Macro Model Mapping

```
set_instance i_Module_U -model Module_U -domain_mapping { {PDdvfs PDdvfs} \

{PD_3p3v PD_3p3v} {PD_2p5v PD_2p5v} }
```

Create Power and Ground Nets

```
create_power_nets -nets VDDdvfs -voltage { 0.864:1.080:0.216 } \

-external_shutoff_condition { !power_on_pin } -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_power_nets -nets VDDpll -voltage 1.08 -external_shutoff_condition { \

!power_on_pin } -peak_ir_drop_limit 0 -average_ir_drop_limit 0

create_power_nets -nets VDDsmc -voltage { 0.864:1.080:0.216 } -internal \

-peak_ir_drop_limit 0 -average_ir_drop_limit 0

create_power_nets -nets VDDrom -voltage { 0.864:1.080:0.216 } -internal \
```

```
-peak_ir_drop_limit 0 -average_ir_drop_limit 0

create_power_nets -nets VDDw -voltage 0.864 -external_shutoff_condition { \

!power_on_pin } -peak_ir_drop_limit 0 -average_ir_drop_limit 0

create_power_nets -nets VDDwake -voltage 1.08 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_power_nets -nets AVDD0_H -voltage 2.5 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_power_nets -nets AVDD1_H -voltage 2.5 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_power_nets -nets vdda33 -voltage 3.3 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_power_nets -nets vdda25 -voltage 2.5 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_power_nets -nets VDDuart -voltage { \

0.864:1.080:0.216 } -internal -peak_ir_drop_limit 0 -average_ir_drop_limit \

0

create_power_nets -nets DUMMY_ESD_VDD -voltage 2.5 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0
```

```
create_ground_nets -nets AGND0_H -voltage 0.0 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_ground_nets -nets AGND1_H -voltage 0.0 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_ground_nets -nets GUARD -voltage 0.0 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_ground_nets -nets VSSsoc -voltage 0.00 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_ground_nets -nets vssa -voltage 0.00 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0

create_ground_nets -nets DUMMY_ESD_VSS -voltage 0.0 -peak_ir_drop_limit 0 \

-average_ir_drop_limit 0
```

Create Operating Corners

```
create_operating_corner -name PDw_wc -voltage 0.864 -process 1 -temperature \

125 -library_set 0v864

create_operating_corner -name PDw_bc -voltage 1.056 -process 1 -temperature 0 \

-library_set 1v056
```

```
create_operating_corner -name PDdvfs1_wc -voltage 1.08 -process 1 -temperature \

125 -library_set 1v080

create_operating_corner -name PDdvfs1_bc -voltage 1.320 -process 1 \

-temperature 0 -library_set 1v320

create_operating_corner -name PDdvfs2_wc -voltage 0.864 -process 1 \

-temperature 125 -library_set 0v864

create_operating_corner -name PDdvfs2_bc -voltage 1.056 -process 1 \

-temperature 0 -library_set 1v056

create_operating_corner -name PDoff_dvfs1_wc -voltage 0 -process 1 \

-temperature 125 -library_set 1v080

create_operating_corner -name PDoff_dvfs1_bc -voltage 0 -process 1 \

-temperature 125 -library_set 1v320

create_operating_corner -name PDoff_dvfs2_wc -voltage 0 -process 1 \

-temperature 125 -library_set 0v864

create_operating_corner -name PDoff_dvfs2_bc -voltage 0 -process 1 \

-temperature 125 -library_set 1v056
```

Create Nominal Condition

```
create_nominal_condition -name norm_1v080 -voltage 1.08

create_nominal_condition -name norm_0v864 -voltage 0.864

create_nominal_condition -name off -voltage 0

create_nominal_condition -name ana_2p5 -voltage 2.5

create_nominal_condition -name ana_3p3 -voltage 3.3


update_nominal_condition -name norm_1v080 -library_set 1v080

update_nominal_condition -name norm_0v864 -library_set 0v864

update_nominal_condition -name ana_2p5 -library_set 2v5

update_nominal_condition -name ana_3p3 -library_set 3v3
```

Create Power Modes

```
create_power_mode -name PMdefault -default -domain_conditions { \

PDpll@norm_1v080 PDwakeup@norm_1v080 PDdvfs@norm_1v080 PDw@norm_0v864 \

PDrom@norm_1v080 PDsmc@norm_1v080 \

i_apb_subsystem__i_uart1__PDuart_SW@norm_1v080 PD_3p3v@ana_3p3 \

PD_2p5v@ana_2p5 }

create_power_mode -name PMdvfs1_smc_off -domain_conditions { PDpll@norm_1v080 \
```

```
PDwakeup@norm_1v080 PDdvfs@norm_1v080 PDw@norm_0v864 PDrom@norm_1v080 \

i_apb_subsystem__i_uart1__PDuart_SW@norm_1v080 PD_3p3v@ana_3p3 \

PD_2p5v@ana_2p5 }

create_power_mode -name PMdvfs1_uart_off -domain_conditions { PDpll@norm_1v080 \

PDwakeup@norm_1v080 PDdvfs@norm_1v080 PDw@norm_0v864 PDsmc@norm_1v080 \

PDrom@norm_1v080 i_apb_subsystem__i_uart1__PDuart_SW@off PD_3p3v@ana_3p3 \

PD_2p5v@ana_2p5 }

create_power_mode -name PMdvfs1_rom_off -domain_conditions { PDpll@norm_1v080 \

PDwakeup@norm_1v080 PDdvfs@norm_1v080 PDw@norm_0v864 PDsmc@norm_1v080 \

i_apb_subsystem__i_uart1__PDuart_SW@norm_1v080 PD_3p3v@ana_3p3 \

PD_2p5v@ana_2p5 }

create_power_mode -name PMdvfs2 -domain_conditions { PDpll@norm_1v080 \

PDwakeup@norm_1v080 PDdvfs@norm_0v864 PDw@norm_0v864 PDrom@norm_0v864 \

PDsmc@norm_0v864 i_apb_subsystem__i_uart1__PDuart_SW@norm_0v864 \

PD_3p3v@ana_3p3 PD_2p5v@ana_2p5 }

create_power_mode -name PMsocoff -domain_conditions { PDwakeup@norm_1v080 \

i_apb_subsystem__i_uart1__PDuart_SW@off PD_3p3v@ana_3p3 PD_2p5v@ana_2p5 }
```

```
update_power_mode -name PMdefault -sdc_files \

../DATA/chip_top.sdc

update_power_mode -name PMdvfs2 -sdc_files \

../DATA/chip_top_dvfs2.sdc
```

Create Analysis Views

```
create_analysis_view -name AVdefault_BC -mode PMdefault -domain_corners { \

PDdvfs@PDdvfs1_bc PDpll@PDdvfs1_bc PDw@PDw_bc PDrom@PDdvfs1_bc \

PDsmc@PDdvfs1_bc i_apb_subsystem__i_uart1__PDuart_SW@PDdvfs1_bc \

PDwakeup@PDdvfs1_bc }

create_analysis_view -name AVdefault_WC -mode PMdefault -domain_corners { \

PDdvfs@PDdvfs1_wc PDpll@PDdvfs1_wc PDw@PDw_wc PDrom@PDdvfs1_wc \

PDsmc@PDdvfs1_wc i_apb_subsystem__i_uart1__PDuart_SW@PDdvfs1_wc \

PDwakeup@PDdvfs1_wc }

create_analysis_view -name AVdvfs2_BC -mode PMdvfs2 -domain_corners { \

PDdvfs@PDdvfs2_bc PDpll@PDdvfs1_bc PDw@PDw_bc PDrom@PDdvfs2_bc \

PDsmc@PDdvfs2_bc i_apb_subsystem__i_uart1__PDuart_SW@PDdvfs2_bc \

PDwakeup@PDdvfs1_bc }
```

```
create_analysis_view -name AVdvfs2_WC -mode PMdvfs2 -domain_corners { \

PDdvfs@PDdvfs2_wc PDpll@PDdvfs1_wc PDw@PDw_wc PDrom@PDdvfs2_wc \

PDsmc@PDdvfs2_wc i_apb_subsystem__i_uart1__PDuart_SW@PDdvfs2_wc \

PDwakeup@PDdvfs1_wc }
```

Create Global Connections

```
create_global_connection -net VDDw -pins { VDD } -domain PDw

create_global_connection -net VSSsoc -pins { VSS } -domain PDw

create_global_connection -net VDDw -pins { VDDw } -domain PDw

create_global_connection -net VSSsoc -pins { VSSsoc } -domain PDw

create_global_connection -net VDDdvfs -pins { VDD } -domain PDdvfs

create_global_connection -net VSSsoc -pins { VSS } -domain PDdvfs

create_global_connection -net VDDw -pins { VDDL } -domain PDdvfs

create_global_connection -net VDDdvfs -pins { VDD_STATE } -domain PDdvfs

create_global_connection -net VDDdvfs -pins { VDDA } -domain PDdvfs

create_global_connection -net VDDdvfs -pins { TVDD } -domain PDdvfs

create_global_connection -net VDDwake -pins { VDD } -domain PDwakeup

create_global_connection -net VSSsoc -pins { VSS } -domain PDwakeup
```

```
create_global_connection -net VDDdvfs -pins { VDDL } -domain PDwakeup

create_global_connection -net VDDpll -pins { VDD } -domain PDpll

create_global_connection -net VSSsoc -pins { VSS } -domain PDpll

create_global_connection -net VDDdvfs -pins { VDDL } -domain PDpll

create_global_connection -net VDDrom -pins { VDD } -domain PDrom

create_global_connection -net VSSsoc -pins { VSS } -domain PDrom

create_global_connection -net VDDdvfs -pins { TVDD } -domain PDrom

create_global_connection -net VDDsmc -pins { VDD } -domain PDsmc

create_global_connection -net VDDdvfs -pins { TVDD } -domain PDsmc

create_global_connection -net VSSsoc -pins { VSS } -domain PDsmc
```

Create Low Power Rules

```
create_power_switch_rule -name pwr_smc_rule -domain PDsmc -external_power_net \

VDDdvfs

create_power_switch_rule -name pwr_rom_rule -domain PDrom -external_power_net \

VDDdvfs

create_power_switch_rule -name i_apb_subsystem__i_uart1__pwr_uart_rule -domain \

i_apb_subsystem__i_uart1__PDuart_SW -external_power_net VDDdvfs
```

```
create_isolation_rule -name i_apb_subsystem__i_uart1__iso_uart_hi \

-isolation_condition { i_apb_subsystem/i_power_ctrl/isolate_urt } -pins { \

i_apb_subsystem/i_uart1/ua_nrts } -from { \

i_apb_subsystem__i_uart1__PDuart_SW } -isolation_target from \

-isolation_output high

create_isolation_rule -name i_apb_subsystem__i_uart1__iso_uart_lo \

-isolation_condition { i_apb_subsystem/i_power_ctrl/isolate_urt } -from { \

i_apb_subsystem__i_uart1__PDuart_SW } -exclude { \

i_apb_subsystem/i_uart1/ua_nrts } -isolation_target from -isolation_output \

low

create_isolation_rule -name iso_rom -isolation_condition { \

i_apb_subsystem/i_power_ctrl/isolate_rom } -from { PDrom } \

-isolation_target from -isolation_output low

create_isolation_rule -name iso_smc_hi -isolation_condition { \

i_apb_subsystem/i_power_ctrl/isolate_smc } -pins { \

i_apb_subsystem/i_smc/smc_n_be[0] i_apb_subsystem/i_smc/smc_n_be[1] \

i_apb_subsystem/i_smc/smc_n_be[2] i_apb_subsystem/i_smc/smc_n_be[3] \

i_apb_subsystem/i_smc/smc_n_cs[0] i_apb_subsystem/i_smc/smc_n_cs[1] \
```

```
i_apb_subsystem/i_smc/smc_n_cs[2] i_apb_subsystem/i_smc/smc_n_cs[3] \

i_apb_subsystem/i_smc/smc_n_cs[4] i_apb_subsystem/i_smc/smc_n_cs[5] \

i_apb_subsystem/i_smc/smc_n_cs[6] i_apb_subsystem/i_smc/smc_n_cs[7] \

i_apb_subsystem/i_smc/smc_hready i_apb_subsystem/i_smc/smc_hresp[0] \

i_apb_subsystem/i_smc/smc_n_ext_oe i_apb_subsystem/i_smc/smc_n_rd \

i_apb_subsystem/i_smc/smc_n_we[3] i_apb_subsystem/i_smc/smc_n_we[2] \

i_apb_subsystem/i_smc/smc_n_we[1] i_apb_subsystem/i_smc/smc_n_we[0] \

i_apb_subsystem/i_smc/smc_n_wr } -from { PDsmc } -isolation_target from \

-isolation_output high

create_isolation_rule -name iso_smc_lo -isolation_condition { \

i_apb_subsystem/i_power_ctrl/isolate_smc } -from { PDsmc } -exclude { \

i_apb_subsystem/i_smc/smc_hready i_apb_subsystem/i_smc/smc_hresp[0] \

i_apb_subsystem/i_smc/smc_n_be* i_apb_subsystem/i_smc/smc_n_cs* \

i_apb_subsystem/i_smc/smc_n_wr i_apb_subsystem/i_smc/smc_n_we* \

i_apb_subsystem/i_smc/smc_n_rd i_apb_subsystem/i_smc/smc_n_ext_oe } \

-isolation_target from -isolation_output low

create_isolation_rule -name iso_towakeup_hi -isolation_condition { \

i_wakeup/isolate_wkup } -to { PDwakeup } -isolation_target from \
```

```
-isolation_output high

create_level_shifter_rule -name DvfsToPll -from { PDdvfs } -to { PDpll }

create_level_shifter_rule -name DvfsToWakeup -from { PDdvfs } -to { PDwakeup }

create_level_shifter_rule -name Module_WToDvfs -from { PDw } -to { PDdvfs }

create_state_retention_rule -name retn_smc -instances { \

i_apb_subsystem/i_smc/i_apb/i_cfreg0 i_apb_subsystem/i_smc/i_apb/i_cfreg1 \

i_apb_subsystem/i_smc/i_apb/i_cfreg2 i_apb_subsystem/i_smc/i_apb/i_cfreg3 \

i_apb_subsystem/i_smc/i_apb/i_cfreg4 i_apb_subsystem/i_smc/i_apb/i_cfreg5 \

i_apb_subsystem/i_smc/i_apb/i_cfreg6 i_apb_subsystem/i_smc/i_apb/i_cfreg7 \

i_apb_subsystem/i_smc/eco_logic_type4[0].smc_eco_type4 \

i_apb_subsystem/i_smc/eco_logic_type4[1].smc_eco_type4 \

i_apb_subsystem/i_smc/eco_logic_type4[2].smc_eco_type4 } -restore_edge \

!i_apb_subsystem/i_power_ctrl/nrestore_smc -save_edge \

i_apb_subsystem/i_power_ctrl/save_smc -target_type flop

update_power_switch_rule -name pwr_smc_rule -prefix FE_CPF_PS \

-peak_ir_drop_limit 0 -average_ir_drop_limit 0

update_power_switch_rule -name pwr_rom_rule -prefix FE_CPF_PS \
```

```
-peak_ir_drop_limit 0 -average_ir_drop_limit 0

update_power_switch_rule -name i_apb_subsystem__i_uart1__pwr_uart_rule -prefix \

FE_CPF_PS -peak_ir_drop_limit 0 -average_ir_drop_limit 0


update_isolation_rules -names i_apb_subsystem__i_uart1__iso_uart_hi -location to \

-prefix CPF_ISO_

update_isolation_rules -names i_apb_subsystem__i_uart1__iso_uart_lo -location to \

-prefix CPF_ISO_

update_isolation_rules -names iso_rom -location to -prefix CPF_ISO_

update_isolation_rules -names iso_smc_hi -location to -prefix CPF_ISO_

update_isolation_rules -names iso_smc_lo -location to -prefix CPF_ISO_

update_isolation_rules -names iso_towakeup_hi -location to -prefix CPF_ISO_


update_level_shifter_rules -names DvfsToPll -location to -prefix CPF_LS_

update_level_shifter_rules -names DvfsToWakeup -location to -prefix CPF_LS_

update_level_shifter_rules -names Module_WToDvfs -location to -prefix CPF_LS_
```

post_cts_tcl_3a

post_cts_tcl:

This plug-in is called after CTS inside the run_cts.tcl flow script. You can use this plug-in to adjust IO timings based on clock insertion delays. The sample commands mentioned below are incorporated in the post_cts plug-in. You can modify these commands according to your design needs.

```
set_interactive_constraint_modes [all_constraint_modes -active]


set_propagated_clock [all_clocks]


set_clock_propagation propagated


set_interactive_constraint_modes {}



set_interactive_constraint_modes {PMdefault}


source ../DATA/chip_top_propagated.sdc


set_interactive_constraint_modes {}



set_interactive_constraint_modes {PMdvfs2}


source ../DATA/chip_top_dvfs2_propagated.sdc


set_interactive_constraint_modes {}
```

post_init_tcl_3a

post_init_tcl:

This plug-in is called after importing the database and CPF inside the run_init.tcl flow script. You can use this plug-in for floorplan related tasks, which cover:

- Die/core boundary creation

- Placement of hard macros/blocks

- Power domain sizing and clearance surrounding to it

- Placement and routing blockages in the floorplan

- IO ring creation

- And PSO planning
  Below are some of the sample commands incorporated in the post_init plug-in. You can modify these commands according to your design needs:

Define Core Boundary

```
floorPlan -site core -d 3000 3000 55.0 55.0 55.0 55.0
```

Relative Placement of Hard Macros/Blocks

```
relativePlace \

i_sram_subsystem/i_1_sram_voltage_island/i_0_SRAM_2kx32_wrap/i_sram_core \

CORE -orientation R90 -relation L -alignedBy B \

-xSpace 18.0000 -ySpace 20.0000

relativePlace \
```

```
i_sram_subsystem/i_2_sram_voltage_island/i_0_SRAM_2kx32_wrap/i_sram_core \

i_sram_subsystem/i_1_sram_voltage_island/i_0_SRAM_2kx32_wrap/i_sram_core \

-orientation R270 -relation R -alignedBy B -xSpace 18.0000 -ySpace 0.0000
```

Power Domain Size and Clearance Information Surrounding it

```
modifyPowerDomainAttr PDsmc -box  280 1325 440 1550 \

-rsExts 15 15 15 15 -minGaps 15 15 15 15
```

Column and Ring-based Power Switch Cell Adding

```
addPowerSwitch -checkerBoard \

-column  \

-powerDomain i_apb_subsystem__i_uart1__PDuart_SW \

-backToBackChain \

-bottomOffset 60 \

-enableNetIn $n \

-enableNetOut pduart_powered_off \

-enablePinIn NSLEEPIN \

-enablePinOut NSLEEPOUT \

-globalSwitchCellName HDRSID2 \
```

```
-leftOffset 20 \

-reportFile $vars(rpt_dir)/PDurt_pso.rpt \

-switchModuleInstance i_apb_subsystem/i_uart1 \

-topOffset 2 \

-rightOffset 2 \

-horizontalPitch 70 \

-height 125 \

-skipRows 5


addPowerSwitch -powerDomain PDsmc -ring \

-switchModuleInstance i_apb_subsystem/i_smc \

-enableNetIn $n \

-enableNetOut pdsmc_powered_off \

-enablePinIn NSLEEPIN \

-enablePinOut NSLEEPOUT \

-reportFile  $vars(rpt_dir)/PDsmc_pso.rpt \

-cornerCellList CDN_RING_CORNER_UL \

-fillerCellNameBottom CDN_RING_FILLER \
```

```
-fillerCellNameLeft CDN_RING_FILLER \

-fillerCellNameRight CDN_RING_FILLER \

-fillerCellNameTop CDN_RING_FILLER \

-bottomSide 1 \

-leftSide 1 \

-rightSide 1 \

-topSide 1 \

-leftOffset 2 \

-rightOffset 2 \

-topOffset 2 \

-bottomOffset 2 \

-bottomNumSwitch 4 -distribute \

-leftNumSwitch 3 -distribute \

-topNumSwitch 4 -distribute \

-rightNumSwitch 3 -distribute \

-switchCellNameLeft CDN_RING_SW \

-switchCellNameRight CDN_RING_SW \

-switchCellNameTop CDN_RING_SW \
```

```
-switchCellNameBottom CDN_RING_SW
```

Create Halo for Macros

```
deleteHaloFromBlock -allBlock

addHaloToBlock 2 2 2 2 i_D0TCM/i_sram_core_1

addHaloToBlock 2 2 2 2 i_D1TCM/i_sram_core_1

addHaloToBlock 2 2 2 2 i_D0TCM/i_sram_core_0

addHaloToBlock 2 2 2 2 i_ITCM/i_sram_core
```

Add IO fillers, these are Inserted from Largest to Smallest

```
addIoFiller -cell PFILLER20

addIoFiller -cell PFILLER10

addIoFiller -cell PFILLER5

addIoFiller -cell PFILLER1

addIoFiller -cell PFILLER05

addIoFiller -cell PFILLER0005
```

post_place_tcl_3a

post_place_tcl:

This plug-in is called after cell placement inside the run_place.tcl flow script. You can use this plug-in for standard cell rail creation and check the connectivity and geometry on power nets.

Standard Cell Rail Creation for Power Domains

Rail Creation for uart Power Domain

```
sroute -noBlockPins -noPadRings -noPadPins -noStripes \

-straightConnections { straightWithDrcClean straightWithChanges } \

-nets { VDDuart VSSsoc } -powerDomains {i_apb_subsystem__i_uart1__PDuart_SW} \

-targetViaTopLayer 4 \

-crossoverViaTopLayer 4 -verbose
```

Rail Creation for PDdvfs Power Domain

```
sroute -noBlockPins -noPadRings -noPadPins -noStripes \

-straightConnections { straightWithDrcClean straightWithChanges } \

-nets { VDDdvfs VSSsoc } -powerDomains { PDdvfs }  \

-targetViaTopLayer 4 -crossoverViaTopLayer 4 -verbose
```

Rail Creation for PDsmc Power Domain

```
sroute -noBlockPins -noPadRings -noPadPins -noStripes \

-straightConnections { straightWithDrcClean straightWithChanges } \

-nets { VDDsmcVSSsoc } -powerDomains { PDsmc }  -targetViaTopLayer 4 \

-crossoverViaTopLayer 4 -layerChangeTopLayer 4
```

Rail Creation for PDwakeup Power Domain

```
sroute -noBlockPins -noPadRings -noPadPins -noStripes \

-straightConnections { straightWithDrcClean straightWithChanges } \

-nets { VDDwake VSSsoc } -powerDomains { PDwakeup }  \

-targetViaTopLayer 4 -crossoverViaTopLayer 4 -layerChangeTopLayer 4
```

Check Connectivity and Geometry on Special Nets

```
verifyConnectivity -type special -nets { VSSsoc VDDdvfs VDDw \

VDDsmc VDDuart VDDwake VDDpll AGND1_H AGND0_H AVDD1_H AVDD0_H \

vdda33 vdda25 vssa}

verifyGeometry -allowPadFillerCellsOverlap \

-allowRoutingBlkgPinOverlap -allowRoutingCellBlkgOverlap
```

post_prects_tcl_3a

post_prects_tcl:

This plug-in is called after the prects optimization is completed inside the run_prects.tcl flow script. You can use this plug-in for secondary power pin routing for low power cells (State retention flops/always-on-buffers/level-shifters) and check the connectivity and geometry on power nets. The sample commands below are incorporated in the post_prects plug-in. You can modify these commands according to your design needs.

Note: `route_secondary_pg_nets` is a procedure implemented in `utils.tcl`, it does secondary pg routing.

```
route_secondary_pg_nets
```
`route_secondary_pg_nets` is part of `utils.tcl` and it internally calls for `setPGPinUseSignalRoute` and `routePGPinUseSignalRoute` commands for routing secondary PG nets.

pre_cts_tcl_3a

pre_cts_tcl:

This plug-in is called before CTS inside the run_cts.tcl flow script. You can use this plug-in to define non-default CTS mode settings. The sample below displays commands incorporated in the pre_cts plug-in. You can modify these commands according to your design needs.

```
setCTSMode -topPreferredLayer 9 -bottomPreferredLayer 3  -preferredExtraSpace 2
```

pre_init_tcl_3a

pre_init_tcl:

This plug-in is called before importing the database, inside the run_init.tcl flow script. You can use this plug-in for tasks which can be done before importing the design. For example, use this plug-in to insert buffers on tie-high/tie-low assign statements. The sample commands mentioned below are incorporated in the pre_init plug-in. You can modify these commands according to your design needs.

```
setImportMode –bufferTieAssign $vars(buffer_tie_assign)
```

pre_place_tcl_3a

pre_place_tcl:

This plug-in is called before doing cell placement inside the `run_place.tcl` flow script.
User can use this plug-in for:

- Power planning related tasks which includes:

    ○ Power planning for power domains (ring/stripe creations)

    ○ Power Shut-off cell power hookup

    ○ Welltap/Bias cell power hookup

- And commands for enabling always-on-net synthesis as part of preCTS optimization.
  Below are some of the sample commands incorporated in the pre_place plug-in. You can
  modify these commands according to your design needs.

Rings Creations for Blocks and for Power Domains

Ring Around PDw Power Domain

```
selectObject Group PDw

addRing -spacing_bottom 1.0 -width_left 4 -width_bottom 4 \

-width_top 4 -spacing_top 1.0 -layer_bottom M5 \

-stacked_via_top_layer M9 -width_right 4 -around power_domain \

-jog_distance 0.1 -offset_bottom 2 -layer_top M5 -threshold 0.1 \

-offset_left 4 -spacing_right 1.0 -spacing_left 1.0 \

-type block_rings -offset_right 4 -offset_top 4 -layer_right M4 \

-nets { VDDw VSSsoc VDDdvfs} -stacked_via_bottom_layer M1 \

-layer_left M4 -snap_wire_center_to_grid Grid
```

Ring Around PLL Macro

```
selectInst i_PLL

addRing -spacing_bottom 1.5 -width_left 4 -width_bottom 4 \

-width_top 10 -spacing_top 3.0 -layer_bottom M5 \

-stacked_via_top_layer M9 -width_right 4 -around selected \

-jog_distance 0.1 -offset_bottom 56 -layer_top M5 \
```

```
-threshold 0.1 -offset_left 60 -spacing_right 1.5 \

-spacing_left 1.5 -type block_rings -offset_right 4 \

-offset_top 20 -layer_right M4 -nets {VDDpll VSSsoc} \

-stacked_via_bottom_layer M1 -layer_left M4 \

-snap_wire_center_to_grid Grid


addRing -spacing_bottom 1.5 -width_left 5 -width_bottom 5 \

-snap_wire_center_to_grid Grid -width_top 5 -left 0 -top 0 \

-spacing_top 1.5 -layer_bottom M5 -stacked_via_top_layer M5 \

-width_right 5 -around selected -jog_distance 0.1 -offset_bottom 70 \

-layer_top M5 -threshold 0.1 -right 0 -offset_left 78 \

-spacing_right 1.5 -spacing_left 1.5 -type block_rings \

-offset_right 78 -offset_top 78 -layer_right M4 -nets VDDdvfs \

-stacked_via_bottom_layer M1 -layer_left M4
```

PLL Pad to PLL Ring Connection

```
sroute -noBlockPins -noPadRings -noCorePins -noStripes \

-padPinAllGeomsConnect -padPinMaxLayer 2 \

-straightConnections { straightWithDrcClean straightWithChanges } \
```

```
-nets {VDDpll} -area { 185 185 2815 2815 }
```

PLL Pins to PLL Ring Connection

```
sroute -blockPinRouteWithPinWidth -noPadRings -noCorePins \

-noPadPins -noStripes -straightConnections { straightWithDrcClean }\

-blockPinTarget { routeToRingOnly } -nets { VDDpll VSSsoc } \

-blocks named -blockNames { i_PLL }
```

Stripe Creations for Power Domains

PDsmc Power Domain Stripe Creation

```
selectObject Group PDsmc

addStripe -block_ring_top_layer_limit M5 \

-max_same_layer_jog_length 0.8  -over_power_domain 1 \

-padcore_ring_bottom_layer_limit M1 -set_to_set_distance 70 \

-skip_via_on_pin {} -stacked_via_top_layer M5 \

-padcore_ring_top_layer_limit M5 -spacing 1.6 -xleft_offset 0.905 \

-xright_offset 10 -merge_stripes_value 0.1 -layer M4 \

-block_ring_bottom_layer_limit M1 -width 6.4 -nets \
```

```
{VSSsoc VDDdvfs VDDsmc} -stacked_via_bottom_layer M1 \

-snap_wire_center_to_grid Grid
```

```
addStripe -block_ring_top_layer_limit M5 \

-max_same_layer_jog_length 0.8 -over_power_domain 1 \

-padcore_ring_bottom_layer_limit M1 -set_to_set_distance 35 \

-skip_via_on_pin {} -stacked_via_top_layer M5 \

-padcore_ring_top_layer_limit M5 -spacing 1.6 -merge_stripes_value \

0.1 -direction horizontal -layer M5 -block_ring_bottom_layer_limit M1 \

-width 6.4 -nets {VSSsoc VDDdvfs VDDsmc} -stacked_via_bottom_layer M1 \

-ytop_offset 2 -ybottom_offset 6 -snap_wire_center_to_grid Grid
```

Full-Chip/Top Level Ring and Stripe Creation

```
addRing -spacing_bottom 2 -width_left 11 -width_bottom 11  \

-width_top 11 -spacing_top 2 -layer_bottom M9 \

-stacked_via_top_layer M9 -width_right 11 -around core \

-jog_distance 0.1 -offset_bottom 2.5 -layer_top M9 -threshold 0.1 \

-offset_left 2.5 -spacing_right 2 -spacing_left 2 -offset_right 2.5 \

-offset_top 2.5 -layer_right M8 -nets {VSSsoc VDDdvfs} \
```

```
-stacked_via_bottom_layer M8 -layer_leftM8



addRing -spacing_bottom 2 -width_left 11 -width_bottom 11 \

-width_top 11 -spacing_top 2 -layer_bottom M9 \

-stacked_via_top_layer M9 -width_right 11 -around core \

-jog_distance 0.1 -offset_bottom 28.5 -layer_top M9 \

-threshold 0.1 -offset_left 28.5 -spacing_right 2 -spacing_left 2 \

-offset_right 28.5 -offset_top 28.5 -layer_right M8 -nets \

{VSSsoc VDDdvfs} -stacked_via_bottom_layer M8 -layer_left M8



addStripe -block_ring_top_layer_limit M9 \

-max_same_layer_jog_length 0.8 -over_power_domain 1 \

-snap_wire_center_to_grid Grid -padcore_ring_bottom_layer_limit M5 \

-set_to_set_distance 35 -stacked_via_top_layer M9 \

-padcore_ring_top_layer_limit M9 -spacing 1.6 -xleft_offset 54 \

-xright_offset 25 -merge_stripes_value 0.1 -layer M8 \

-block_ring_bottom_layer_limit M5 \

-width 6.4 -nets {VSSsoc VDDdvfs} -stacked_via_bottom_layer M5 \

-allow_jog_block_ring 0
```

```
addStripe -area { 193 245 2807 2755 } -block_ring_top_layer_limit M9 \

-max_same_layer_jog_length 0.8 -snap_wire_center_to_grid Grid \

-padcore_ring_bottom_layer_limit M5 -set_to_set_distance 35 \

-stacked_via_top_layer M9 -padcore_ring_top_layer_limit M9 \

-spacing 1.6 -merge_stripes_value 0.1 -direction horizontal -layer M9 \

-block_ring_bottom_layer_limit M5 -width 6.4 -nets {VSSsoc VDDdvfs} \

-stacked_via_bottom_layer M5 -allow_jog_block_ring 0 -ytop_offset 18 \

-ybottom_offset 18
```

For welltap/Bias/Power-Switch cells PG pin connection, you can run strap over/off-set with regard to these cells so that the PG connection is established easily. For example, in the below stripe creation command, the stripe is created over welltap cells (FILLBIAS2A9TH) so that PG connection is easily established.

```
addStripe -nets {VDDcpu} -layer ME4 -direction vertical -width 0.2 \

-spacing 1 -over_pins 1 -pin_layer ME1 -master FILLBIAS2A9TH \

-over_power_domain1 -block_ring_top_layer_limit ME7 \

-block_ring_bottom_layer_limit ME4 -stacked_via_top_layer ME7 \

-stacked_via_bottom_layer ME1 -skip_via_on_pin {} \

-switch_layer_over_obs 1
```

To handle always-on net synthesis as part of preCTS optimization, you have to enable these:

```
setDontUse PTBUFFD* false
```

```
setOptMode -resizeShifterAndIsoInsts true
```

```
setvar dpgOptSupportAOFeedThru 1
```

Add Placement Blockages

```
createObstruct 2727.626 1098.3065 2755.178 1143.1175
```

```
createObstruct 1992.592 245.018 2012.6265 437.8765
```

```
createObstruct 2748.0225 2132.4865 2755.371 2192.543
```

```
createObstruct 2750.585 1827.1725 2755.96 1886.794
```

7

Hierarchical Flow

Innovus Implementation System Hierarchical Flow is a more complex implementation of the flat foundation flow.
In this section we cover:

- Using Interface Logic Models

- About the Hierarchical One-Pass ILM Flow and Diagram

- Hierarchical Foundation Flow with FlexILM

- Results for Hierarchical Flow Implementation

Introduction to Hierarchical Implementation Flow

This chapter describes the tasks in the recommended flow using the Innovus Implementation System (Innovus) software to implement a hierarchical flat chip from a completed floorplan, while meeting timing and physical design requirements and resolving signal integrity (SI) problems. In addition, for designs with multiple operating modes or corners that require optimization, and for more accurate timing, the flow supports multi-mode multi-corner (MMMC) timing analysis and on-chip variation (OCV) mode. The flow also takes advantage of the Innovus software's multiple-CPU processing capabilities to accelerate the design process and runs features in multi-threading or distributed processing mode where appropriate.

The flow uses the Innovus super commands. For example, `place_opt_design`, `routeDesign`, and `optDesign` with as few non-default options as possible for the main design tasks. It is a starting point for you to use with designs for a hierarchical chip. Ultimately, your design might have a different final set of commands, but this flow is the recommended starting point, before customization for your specific design or technology needs.

When the tasks described in this guide are complete, you should have a design that is ready for DRC and LVS checks and other sign-off tasks.

Overview of Hierarchical Implementation Flow

The hierarchical implementation flow starts with a completed floorplan, including partitions. For the top level and for each block, clock tree estimation is performed for budgeting and CTS.

When implementing a hierarchical flow where the design is divided into partitions, you can treat the partitions as blackboxes and derive timing estimations for these blackboxes. For a more accurate representation, instead of using a blackbox at the top level, you can create an Interface Logic Model (ILM) at the block level and use an ILM as you would use a blackbox.

This flow uses ILM's for the top-level implementation. Steps for using the blackbox flow are noted as comments but not actually used in the flow.

At the end of each main task, there is an optimization step, and the task is complete when violations are fixed and the timing is acceptable. The flow is complete and the design is ready for sign-off tasks when the design meets timing and physical design requirements.

One-Pass and Two -Pass Hierarchical Flows

The hierarchical one-pass ILM flow assumes a single iteration of top-level budgeting. This flow can be used for designs that do not have complex inter-partition timing flows.

For complex inter-partition timing flows, a single iteration of top-level budgeting might not be sufficient. In such cases, you can use the hierarchical two-pass ILM flow, which uses a two-step iteration of top-level budgeting. The two-step flow results in more accurate budgeting and ensures high accuracy block closure. The two-pass hierarchical flow is provided as reference. The Foundation Flow kits do not currently provide automation to execute this flow.

Before You Begin - Hierarchical Implementation Flow

The hierarchical implementation flow assumes that timing and routing constraints are feasible.

Make sure the following tasks are complete before starting this flow:

- Floorplanning, including the following tasks:
  - Partition definition
  - Hard block placement

- Power grid creation

- Region definition

- I/O pad placement

- Creation of placement and routing blockages

Note: Leakage power and dynamic power optimization are outside the scope of this flow.

Inputs for Hierarchical Implementation Flow

- Innovus configuration file (setup.tcl) with the following information specified:

  - Timing libraries (*.lib)

  - LEF libraries (*.lef)

  - Timing constraints (*.sdc)

  - Capacitance table or QRC technology file

  - SI libraries (*.cdb or *.udn) (recommended but not required)

- Verilog netlist (*.v)

- Floorplan file (*.fp or *.def)

- Clock tree specification file

- Scan chain information (*.tcl or *.def)In addition, depending on the design and technology, you may need to supply values for the following items:

- RC scaling factors

- OCV derating factors

- Metal fill parameters

- Filler cell names

- Tie cell names

- Clock gating cell names

- Spare instance names

- JTAG instance names

- JTAG rows

- LSF queue

- Don't Use Cells

- Delay Cells

- CTS Cells

Hierarchical Partition Flow

With the hierarchical partition flow, standard cell placement, feedthrough insertion, pin assignment, timing budgeting, and partition steps are performed. It provides capabilities for defining and handling partition blocks, blackboxes, various orientations, rectilinear shapes, multiply-instantiated partitions, multiple-levels of partitions, partition guard-bands, and so on. It creates different directories for the top-level and the blocks – which can then be independently implemented by different design teams.

The following capabilities are supported:

- Placement or route based partition (feedthrough insertion/pin assignment) flow. You also have the option to disable feedthrough buffer insertion step.

- Channelless and/or design with master/clones.

- Option to use prototyping net delay (pico-second per micron) model for quick timing estimation during partition flow.

- FlexModel based hierarchical partition flow for reducing netlist size.

- Different virtual optimization engines for timing budgeting

    - Trial IPO

    - gigaOptVirtual

    - Pico-second per micron model (psPM)

Implementing Top-Level and Blocks Concurrently

The hierarchical flow lets you implement the top-level design and the blocks concurrently. When you partition a design, the Innovus software creates LEF files and budgeted timing models for the blocks. You can treat the blocks as blackboxes from the top-level perspective— and thus implement the top-level and the blocks concurrently.

During the flow, as the block implementation progresses, the budgeted timing models for the blocks are often replaced with Interface Logic Models (ILM's) or with more accurate budgeted timing models. You can also add new LEF files that contain the antenna information for the blocks.

See also,

Using Interface Logic Models

About the Hierarchical One-Pass ILM Flow and Diagram

Hierarchical Foundation Flow with FlexILM

Results for Hierarchical Flow Implementation

Using Interface Logic Models

An Interface Logic Model (ILM) is a structural representation of the timing characteristics of a block, specifically a subset of the block's structure including instances along the I/O timing paths, clock-tree instances, and instances or net coupling affecting the signal integrity (SI) on I/O timing paths. Instead of using a blackbox to represent a block at the top level, you create an ILM at the block level and use it as you would use a blackbox. Using ILM's to represent the timing characteristics of a block provides more accurate analysis than using the blackbox flow.

The flow described in this document uses ILM's for representing the partition blocks during the top-level implementation.

See also,

Hierarchical Flow

About the Hierarchical One-Pass ILM Flow and Diagram

Using the Flattened and Unflattened ILM States

ILM's can be in either of the following two states:

- Flattened state
  In this state, the spef file or the Verilog netlist for the ILM is flattened and merged into the top level. The flattened state is used for accurate delay calculation and Static Timing Analysis. The flattened state is required by the CTE commands such as `report_timing`.

- Unflattened state
  In this state, the spef file or the Verilog netlist of the ILM model is not visible at the top level. The ILM appears as a blackbox. The unflattened state is used for physical implementation tasks such as routing.
  Here are some commonly used commands that require an unflattened state:

    - power routing commands such as `addStripe`

    - physical cell insertion commands such as `addFiller`

    - scan commands such as `scanReorder`
      The Innovus super commands `place_opt_design`, `routeDesign`, and `optDesign` automatically manage the flattened and unflattened ILM states. For other commands, you should flatten or unflatten the design as required with the `flattenIlm` and the `unflattenIlm` commands.

See also,

Performing Multi-Mode Multi-Corner Timing Analysis and Optimization

For designs that contain multiple modes and/or corners, it becomes necessary to configure the software to support multiple combinations of modes and corners and to evaluate them concurrently. Multi-Mode Multi-Corner (MMMC) Timing Analysis provides the ability to determine and validate the timing requirements for such designs.
Multi-Mode Multi-Corner (MMMC) Timing Analysis and Optimization is also recommended for better timing correlation.
Using MMMC timing analysis, you can concurrently perform the following activities:

- Minimum and maximum analysis for each library corner

- On-chip variation specification

- Push-out and pull-in SDF annotation

    - Setup analysis uses push-out for launch, pull-in for capture

    - Hold analysis uses pull-in for launch, and push-out for capture

Note: defineRCCorner flows are not supported with MMMC

Note: For the ILM flow, temperature scaling for resistance must be done using MMMC.

See also,

Using the Flattened and Unflattened ILM States

Managing Clock Latencies

Deriving Clock Tree Estimation for Budgeting and CTS

Preventing Hierarchical Signal Integrity Issues

Managing Clock Latencies

The ILM models can be used during pre-CTS and post-CTS stages without modification. However, for blackboxes, the following considerations related to clock latency apply:

- Clock latency is reflected directly in the timing arcs. Any included latency is not calculated.

- During the pre-CTS stage, the delays of the clock tree are assumed to be 0. To specify clock latency, create SDC constraints with the `set_clock_latency` command.

- If the delays of the clock tree are included, during pre-CTS optimization, the latency of the clock pins should be included in the SDC file to account for the top-level clock latency (source latency)

- During the post-CTS stage, the delays of the clock tree are calculated and latencies asserted on pins or with -source are retained.

- The `deriveTimingBudget` command and `savePartition`command will generate a budgeted library for use at the top level:

    - This model assumes zero delay on the clock tree

    - For pre-CTS optimization, any clock latencies applied to the clock pins of the model should represent the full clock insertion (top and block)

    - This model cannot be used in the post-CTS stage. Use the `saveModel` command to create new models for each partition in the post-CTS stage.

See also,

Using the Flattened and Unflattened ILM States

Performing Multi-Mode Multi-Corner Timing Analysis and Optimization

Deriving Clock Tree Estimation for Budgeting and CTS

Preventing Hierarchical Signal Integrity Issues

Deriving Clock Tree Estimation for Budgeting and CTS

You need to derive the clock tree estimation for the blocks in the design for timing budgeting and for Clock Tree Synthesis (CTS). Derive the CTS files and the clock latencies for the top level and for each block separately as follows:

1. For each block and for the top-level:

    - create a CTS specification file with the appropriate minimum delay and maximum delay based on the achievable insertion delays

    - use the `set_clock_latency` command to create a latency SDC with for each clock. The latency should match the minimum- and maximum-delays set in the CTS specification file.

2. At the top level, specify the latency SDC's with the `setPtnUserCnsFile` command.

When you perform timing budgeting, latencies in the block- and top-level will be adjusted based on the latency SDC's. MacroModels for top-level CTS will be used in a .lib flow.

3. Control the latency manually in the following cases:

   ○ when the design contains clock feedthroughs

   ○ when clocks are generated within a partition

4. Derive the budgeted latencies by performing a prototyping pass through the flow including CTS

Example

The following example illustrates the use flow for deriving CTS files and latencies separately for the top level and for each block.

For this example, assume a single clock FCLK that attaches to pin pll/clk at the top level and enters partitions A and B at port clk
The insertion delay target for flat full chip insertion 5 ns. The insertion delay target for partition A is 3 ns and that for partition B is 2 ns.

CTS Specification File

```
# Ptn A


AutoCTSRootPin clk


MinDelay 3ns


MaxDelay 3ns
```

```
# Ptn B

AutoCTSRootPin clk

MinDelay 2ns

MaxDelay 2ns

# Top Level

# Only required for Black Box flow

# MacroModel port PtnA/clk 3ns 3ns 3ns 3ns 10ff

# MacroModel port PtnB/clk 2ns 2ns 2ns 2ns 10ff

AutoCTSRootPin pll/clk

MinDelay 5ns

MaxDelay 5ns
```

Latency SDC File

```
# Ptn A

set_clock_latency 3 [get_clocks {FCLK}]

# Ptn B

set_clock_latency 2 [get_clocks {FCLK}]

# Full Chip

set_clock_latency 5 [get_clocks {FCLK}]
```

Note: MacroModels are not required in the ILM flow because CTS automatically calculates the latency based on the spef file. The `saveModel` command produces the MacroModels automatically

Latencies for the Full Chip and the Partitions

Before you specify partitions and perform timing budgeting and partitions, the full chip latencies must either be in the original timing constraints or loaded in a separate SDC
The latency SDC's for each partition are applied using the `setPtnUserCnsFile` command as follows

```
setPtnUserCnsFile \

-fileName PartitionLatencySDCFile

-ptnName PartitionName
```

In the budgeted SDC's for each partition, the IO arrival and departure times reflect the assigned latencies. The source and network latencies are also assigned
For this example, the source latency will be 2 ns for partition A (5 ns – 3 ns) and 3 ns for partition B (5 ns -2 ns)
The same budgeted SDC's can be used for pre-CTS and post-CTS optimization. The network latencies will be ignored once the tree is synthesized
Related Topics:

- Creating a Clock Tree Specification File in the "Synthesizing Clock Trees" chapter in the Innovus User Guide

- Timing Budgeting chapter in the Innovus User Guide

See also,

Using the Flattened and Unflattened ILM States

Performing Multi-Mode Multi-Corner Timing Analysis and Optimization

Managing Clock Latencies

Preventing Hierarchical Signal Integrity Issues

Preventing Hierarchical Signal Integrity Issues

When working on hierarchical designs, preventing hierarchical signal integrity (SI) issues between partitions and the top level can become important. You can use many strategies to prevent hierarchical SI issues. Most of these strategies are performed during floorplanning.
These strategies are not specifically covered in this guide, but include the following:

- adding placement halos to the partitions to limit the need for routing close to the partition boundaries

- adding routing halos to the partition to limit cross-coupling between routes in the partitions and top level
  Some methodologies such as routing over, or through, partitions using partition feedthroughs can increase the SI sensitivity. In such cases, you can reduce the SI sensitivity by attaching buffers to the I/O ports of each partition—this reduces SI sensitivity by limiting the I/O nets to short routes and filtering injected noise.

See also,

Using the Flattened and Unflattened ILM States

Performing Multi-Mode Multi-Corner Timing Analysis and Optimization

Managing Clock Latencies

Deriving Clock Tree Estimation for Budgeting and CTS

About the Hierarchical One-Pass ILM Flow and Diagram

The hierarchical one-pass ILM flow documented in this chapter utilizes a single iteration of top-level budgeting. This flow is useful for designs that do not have complex inter-partition timing.
For designs that have more complex inter-partition flows, you can use the hierarchical two-pass ILM flow, which utilizes a two-step iteration of top-level budgeting to generate more accurate budgets after clock insertion.

The following figure illustrates the hierarchical one-pass ILM flow.

See also,

Hierarchical Flow

Using Interface Logic Models

Hierarchical Foundation Flow with FlexILM

Results for Hierarchical Flow Implementation

Hierarchical Foundation Flow with FlexILM

The hierarchical foundation flow with FlexILM (preCTS) creates reduced netlist and physical model from each partition. With automatic netlist and placement ECO after top-level paths optimization, memory reduction is done in both timing graph and physical data. With this, the need for strict timing budget accuracy is reduced, resulting in no requirement for using flattenILM or unflattenILM during the flow.

Setting Up the Flow

To enable the FlexILM (preCTS) hierarchical flow, set `vars(enable_flexilm) true`

The files and folders listed below are created after `tclsh SCRIPTS/gen_flow.tcl -m hier all`:

`DBS/ FF/ LOG/ make/ Makefile PARTITION/ PARTITION_PRECTS/ RPT/ setup.tcl`

Single Pass Hierarchical Flow

Make –f Makefile

- Initial partitioning and budgeting

- Run block to preCTS and generate FlexILM

- Top FlexILM prectsopt , re-assign ptn pin , timing rebudget and FlexILM-ECO

- Assemble updated preCTS DB and check timing

- Block implementation from CCOPT/CTS to signoff , generate ILM

- Top implementation from CCOPT/CTS to signoff with ILM

- Final chip assembly

FlexILM Generation and Top preCTS Optimization with FlexILM

Block FlexILM Generation After preCTS-opt

```
optDesign –preCTS –outDir RPT –prefix prects
createInterfaceLogic –dir block.flexilm –useType flexIlm –optStage preCTS
```

Top preCTS Optimization with FlexILM

```
###commit flexIlm####
set dirs ""
foreach name $vars(partition_list) { append dirs " –flex_ilm
{${name} ../${name}/${name}.flexilm}" }
eval commit_module_model $dirs –mmmc_file ../../FF/view_definition.tcl

###optDesign ####
setTrialRouteMode –honorPin false –handlePartitionComplex true
setHierMode –optStage preCTS
setOptMode –handlePartitionComplex true
optDesign –preCTS –outDir RPT –prefix flexilm
```

Reassigning PTN Pin and Rebudgeting Timing

```
###reassign ptn pin and rebudget###
assignPtnPin
checkPinAssignment
setTrialRouteMode -honorPin true
trialRoute
deriveTimingBudget

###flexIlm ECO to update block DB####
set dirs ""
foreach block $vars(partition_list) { append dirs " -goldenBlockDir
../${block}/DBS/prects.enc.dat" }
eval update_partition -flexIlmECO $dirs -flexIlmDir ../../PARTITION_FLEXILM -
postECOSuffix postECO -pinLocation
```

See also,

[Hierarchical Flow](#)

[Using Interface Logic Models](#)

[About the Hierarchical One-Pass ILM Flow and Diagram](#)

[Results for Hierarchical Flow Implementation](#)

Results for Hierarchical Flow Implementation

The design is now ready for sign-off physical and timing verification. The flow outputs the following files:

- Design data
  - Verilog netlist
  - SPEF file
  - GDS file
  - Innovus log file

Use the [viewLog](#) command to open the log file in a new window.

- Reports
  - Timing reports

- Clock tree reports

- Verify connectivity

- Verify geometry

- Verify metal density

- Verify process antenna

See also,

Hierarchical Flow

Using Interface Logic Models

About the Hierarchical One-Pass ILM Flow and Diagram

Hierarchical Foundation Flow with FlexILM

8

Tags for Flow

The following list depicts the tags for the Innovus foundation flow.

```
vars(init,set_distribute_host,tag)                    value

vars(init,set_multi_cpu_usage,tag)                    value

vars(init,set_rc_factor,tag)                          value

vars(init,derate_timing,tag)                          value

vars(init,create_rc_corner,tag)                       value

vars(init,create_library_set,tag)                     value

vars(init,create_delay_corner,tag)                    value

vars(init,create_constraint_mode,tag)                 value

vars(init,create_analysis_view,tag)                   value

vars(init,update_delay_corner,tag)                    value

vars(init,update_library_set,tag)                     value

vars(init,set_default_view,tag)                       value

vars(init,set_power_analysis_mode,tag)                value

vars(init,load_config,tag)                            value
```

```
vars(init,load_floorplan,tag)                              value

vars(init,add_tracks,tag)                          value

vars(init,load_cpf,tag)                                    value

vars(init,commit_cpf,tag)                                  value

vars(init,read_activity_file,tag)                          value

vars(init,specify_ilm,tag)                                 value

vars(init,load_ilm_non_sdc_file,tag)                       value

vars(init,initialize_timing,tag)                           value

vars(init,load_scan,tag)                                   value

vars(init,specify_spare_gates,tag)                         value

vars(init,set_dont_use,tag)                                value

vars(init,set_max_route_layer,tag)                         value

vars(init,set_design_mode,tag)                             value

vars(init,insert_welltaps_endcaps,tag)                     value

vars(init,load_config,tag)                                  value

vars(init,time_design,tag)                                  value

vars(init,check_design,tag)                                 value

vars(init,check_timing,tag)                                 value
```

```
vars(init,report_power_domains,tag)                          value

vars(place,set_distribute_host,tag)                          value

vars(place,set_multi_cpu_usage,tag)                          value

vars(place,restore_design,tag)                               value

vars(place,initialize_step,tag)                              value

vars(place,set_design_mode,tag)                              value

vars(place,set_delay_cal_mode,tag)                           value

vars(place,set_place_mode,tag)                               value

vars(place,set_opt_mode,tag)                                 value

vars(place,cleanup_specify_clock_tree,tag)                   value

vars(place,specify_clock_tree,tag)                           value

vars(place,specify_jtag,tag)                                 value

vars(place,place_jtag,tag)                                   value

vars(place,place_design,tag)                                 value

vars(place,add_tie_cells,tag)                                value

vars(place,time_design,tag)                                  value

vars(place,save_design,tag)                                  value

vars(place,report_power,tag)                                 value
```

```
vars(place,verify_power_domain,tag)                         value

vars(place,run_clp,tag)                                     value

vars(prects,set_distribute_host,tag)                        value

vars(prects,set_multi_cpu_usage,tag)                        value

vars(prects,initialize_step,tag)                            value

vars(prects,set_design_mode,tag)                            value

vars(prects,set_ilm_type,tag)                               value

vars(prects,cleanup_specify_clock_tree,tag)                 value

vars(prects,create_clock_tree_spec,tag)                     value

vars(prects,specify_clock_tree,tag)                         value

vars(prects,set_useful_skew_mode,tag)                       value

vars(prects,set_opt_mode,tag)                               value

vars(prects,set_design_mode,tag)                            value

vars(prects,set_delay_cal_mode,tag)                         value

vars(prects,set_dont_use,tag)                               value

vars(prects,opt_design,tag)                                 value

vars(prects,ck_clone_gate,tag)                              value

vars(prects,save_design,tag)                                value
```

```
vars(prects,report_power,tag)                          value

vars(prects,verify_power_domain,tag)                   value

vars(prects,run_clp,tag)                               value

vars(cts,set_distribute_host,tag)                      value

vars(cts,set_multi_cpu_usage,tag)                      value

vars(cts,initialize_step,tag)                          value

vars(cts,set_design_mode,tag)                          value

vars(cts,set_cts_mode,tag)                             value

vars(cts,set_nanoroute_mode,tag)                       value

vars(cts,enable_clock_gate_cells,tag)                  value

vars(cts,clock_design,tag)                             value

vars(cts,disable_clock_gate_cells,tag)                 value

vars(cts,run_clock_eco,tag)                            value

vars(cts,update_timing,tag)                            value

vars(cts,time_design,tag)                              value

vars(cts,save_design,tag)                              value

vars(cts,report_power,tag)                             value

vars(cts,verify_power_domain,tag)                      value
```

`vars(cts,run_clp,tag)`	value
`vars(postcts,set_distribute_host,tag)`	value
`vars(postcts,set_multi_cpu_usage,tag)`	value
`vars(postcts,initialize_step,tag)`	value
`vars(postcts,set_design_mode,tag)`	value
`vars(postcts,set_delay_cal_mode,tag)`	value
`vars(postcts,set_analysis_mode,tag)`	value
`vars(postcts,set_opt_mode,tag)`	value
`vars(postcts,opt_design,tag)`	value
`vars(postcts,save_design,tag)`	value
`vars(postcts,report_power,tag)`	value
`vars(postcts,verify_power_domain,tag)`	value
`vars(postcts,run_clp,tag)`	value
`vars(postcts_hold,set_distribute_host,tag)`	value
`vars(postcts_hold,set_multi_cpu_usage,tag)`	value
`vars(postcts_hold,initialize_step,tag)`	value
`vars(postcts_hold,set_dont_use,tag)`	value
`vars(postcts_hold,set_opt_mode,tag)`	value

```
vars(postcts_hold,opt_design,tag)                          value

vars(postcts_hold,save_design,tag)                         value

vars(postcts_hold,report_power,tag)                        value

vars(postcts_hold,verify_power_domain,tag)                 value

vars(postcts_hold,run_clp,tag)                             value

vars(route,set_distribute_host,tag)                        value

vars(route,set_multi_cpu_usage,tag)                        value

vars(route,initialize_step,tag)                            value

vars(route,set_nanoroute_mode,tag)                         value

vars(route,add_filler_cells,tag)                           value

vars(route,route_secondary_pg_nets,tag)                    value

vars(route,check_place,tag)                                value

vars(route,route_design,tag)                               value

vars(route,run_clock_eco,tag)                              value

vars(route,spread_wires,tag)                               value

vars(route,initialize_timing,tag)                          value

vars(route,time_design,tag)                                value

vars(route,save_design,tag)                                value
```

```
vars(route,report_power,tag)                              value

vars(route,verify_power_domain,tag)                       value

vars(route,run_clp,tag)                                   value

vars(postroute,set_distribute_host,tag)                   value

vars(postroute,set_multi_cpu_usage,tag)                   value

vars(postroute,initialize_step,tag)                       value

vars(postroute,set_design_mode,tag)                       value

vars(postroute,set_extract_rc_mode,tag)                   value

vars(postroute,set_analysis_mode,tag)                     value

vars(postroute,set_delay_cal_mode,tag)                    value

vars(postroute,add_metalfill,tag)                         value

vars(postroute,delete_filler_cells,tag)                   value

vars(postroute,opt_design,tag)                            value

vars(postroute,add_filler_cells,tag)                      value

vars(postroute,trim_metalfill,tag)                        value

vars(postroute,save_design,tag)                           value

vars(postroute,report_power,tag)                          value

vars(postroute,verify_power_domain,tag)                   value
```

`vars(postroute,run_clp,tag)`	value
`vars(postroute_hold,set_distribute_host,tag)`	value
`vars(postroute_hold,set_multi_cpu_usage,tag)`	value
`vars(postroute_hold,initialize_step,tag)`	value
`vars(postroute_hold,set_dont_use_mode,tag)`	value
`vars(postroute_hold,set_opt_mode,tag)`	value
`vars(postroute_hold,delete_filler_cells,tag)`	value
`vars(postroute_hold,opt_design,tag)`	value
`vars(postroute_hold,add_filler_cells,tag)`	value
`vars(postroute_hold,trim_metalfill,tag)`	value
`vars(postroute_hold,save_design,tag)`	value
`vars(postroute_hold,report_power,tag)`	value
`vars(postroute_hold,verify_power_domain,tag)`	value
`vars(postroute_hold,run_clp,tag)`	value
`vars(postroute_si_hold,set_distribute_host,tag)`	value
`vars(postroute_si_hold,set_multi_cpu_usage,tag)`	value
`vars(postroute_si_hold,initialize_step,tag)`	value
`vars(postroute_si_hold,set_design_mode,tag)`	value

```
vars(postroute_si_hold,set_dont_use,tag)                    value

vars(postroute_si_hold,set_opt_mode,tag)                    value

vars(postroute_si_hold,set_extract_rc_mode,tag)             value

vars(postroute_si_hold,set_si_mode,tag)                     value

vars(postroute_si_hold,set_delay_cal_mode,tag)              value

vars(postroute_si_hold,set_analysis_mode,tag)               value

vars(postroute_si_hold,add_metalfill,tag)                   value

vars(postroute_si_hold,delete_filler_cells,tag)             value

vars(postroute_si_hold,opt_design,tag)                      value

vars(postroute_si_hold,add_filler_cells,tag)                value

vars(postroute_si_hold,trim_metalfill,tag)                  value

vars(postroute_si_hold,save_design,tag)                     value

vars(postroute_si_hold,report_power,tag)                    value

vars(postroute_si_hold,verify_power_domain,tag)             value

vars(postroute_si_hold,run_clp,tag)                         value

vars(postroute_si,set_distribute_host,tag)                  value

vars(postroute_si,set_multi_cpu_usage,tag)                  value

vars(postroute_si,initialize_step,tag)                      value
```

```
vars(postroute_si,set_design_mode,tag)                        value

vars(postroute_si,set_extract_rc_mode,tag)                    value

vars(postroute_si,set_si_mode,tag)                            value

vars(postroute_si,set_analysis_mode,tag)                      value

vars(postroute_si,set_delay_cal_mode,tag)                     value

vars(postroute_si,add_metalfill,tag)                          value

vars(postroute_si,delete_filler_cells,tag)                    value

vars(postroute_si,opt_design,tag)                             value

vars(postroute_si,add_filler_cells,tag)                       value

vars(postroute_si,trim_metalfill,tag)                         value

vars(postroute_si,save_design,tag)                            value

vars(postroute_si,report_power,tag)                           value

vars(postroute_si,verify_power_domain,tag)                    value

vars(postroute_si,run_clp,tag)                                value

vars(signoff,set_distribute_host,tag)                         value

vars(signoff,set_multi_cpu_usage,tag)                         value

vars(signoff,initialize_timing,tag)                           value

vars(signoff,initialize_step,tag)                             value
```

```
vars(signoff,set_analysis_mode,tag)                        value

vars(signoff,set_extract_rc_mode,tag)                      value

vars(signoff,extract_rc,tag)                               value

vars(signoff,dump_spef,tag)                                value

vars(signoff,time_design_setup,tag)                        value

vars(signoff,time_design_hold,tag)                         value

vars(signoff,stream_out,tag)                               value

vars(signoff,save_oa_design,tag)                           value

vars(signoff,create_ilm,tag)                               value

vars(signoff,summary_report,tag)                           value

vars(signoff,verify_connectivity,tag)                      value

vars(signoff,verify_geometry,tag)                          value

vars(signoff,verify_metal_density,tag)                     value

vars(signoff,verify_process_antenna,tag)                   value

vars(signoff,save_design,tag)                              value

vars(signoff,report_power,tag)                             value

vars(signoff,verify_power_domain,tag)                      value

vars(signoff,ru_clp,tag)                                   value
```

```
vars(partition_list)                                   value

vars(partition,initialize_timing,tag)                  value

vars(partition,load_cpf,tag)                           value

vars(partition,commit_cpf,tag)                         value

vars(partition,run_clp_init,tag)                       value

vars(partition,save_init_dbs,tag)                      value

vars(partition,set_budgeting_mode,tag)                 value

vars(partition,update_constraint_mode,tag)             value

vars(partition,set_ptn_user_cns_file,tag)              value

vars(partition,set_place_mode,tag)                     value

vars(partition,place_design,tag)                       value

vars(partition,save_place_dbs,tag)                     value

vars(partition,trial_route,tag)                        value

vars(partition,assign_ptn_pins,tag)                    value

vars(partition,check_pin_assignment,tag)               value

vars(partition,report_unaligned_nets,tag)              value

vars(partition,set_ptn_pin_status,tag)                 value

vars(partition,derive_timing_budget,tag)               value
```

```
vars(partition,save_budget_dbs,tag)                      value

vars(partition,run_clp,tag)                              value

vars(partition,partition,tag)                             value

vars(partition,save_partition,tag)                       value

vars(assemble,assemble_design,tag)                       value

vars(assemble,specify_ilm,tag)                           value

vars(assemble,load_ilm_non_sdc_file,tag)                 value

vars(assemble,load_cpf,tag)                              value

vars(assemble,commit_cpf,tag)                            value

vars(assemble,initialize_timing,tag)                     value

vars(assemble,update_timing,tag)                         value

vars(assemble,pre_pac_verify_connectivity,tag)           value

vars(assemble,pre_pac_verify_geometry,tag)               value

vars(assemble,set_module_view,tag)                       value

vars(assemble,delete_filler_cells,tag)                   value

vars(assemble,opt_design,tag)                            value

vars(assemble,add_filler_cells,tag)                      value

vars(assemble,post_pac_verify_connectivity,tag)          value
```

```
vars(assemble,post_pac_verify_geometry,tag)                value
```

9

Sample Script - Code Generator

- [Executing the Flow](#)
- [Sample Single Script Flow](#)

Executing the Flow

```
init_design FF/run.conf

generateTracks

readActivityFile -format TCF $vars(activity_file)

create_rc_corner -name rc_min \

-cap_table $vars(rc_min,cap_table) \

-preRoute_res $vars(rc_min,pre_route_res_factor) \

-preRoute_cap $vars(rc_min,pre_route_cap_factor) \

-preRoute_clkres $vars(rc_min,pre_route_clk_res_factor) \

-preRoute_clkcap $vars(rc_min,pre_route_clk_cap_factor) \

-postRoute_res $vars(rc_min,post_route_res_factor) \

-postRoute_cap $vars(rc_min,post_route_cap_factor) \

-postRoute_clkres $vars(rc_min,post_route_clk_res_factor) \
```

```
-postRoute_clkcap $vars(rc_min,post_route_clk_cap_factor) \

-postRoute_xcap $vars(rc_min,post_route_xcap_factor) \

-T $vars(rc_min,T) \

-qx_tech_file $vars(rc_min,qx_tech_file)

create_rc_corner -name rc_max \

-cap_table $vars(rc_max,cap_table) \

-preRoute_res $vars(rc_max,pre_route_res_factor) \

-preRoute_cap $vars(rc_max,pre_route_cap_factor) \

-preRoute_clkres $vars(rc_max,pre_route_clk_res_factor) \

-preRoute_clkcap $vars(rc_max,pre_route_clk_cap_factor) \

-postRoute_res $vars(rc_max,pre_route_res_factor) \

-postRoute_cap $vars(rc_max,pre_route_cap_factor) \

-postRoute_clkres $vars(rc_max,pre_route_clk_res_factor) \

-postRoute_clkcap $vars(rc_max,pre_route_clk_cap_factor) \

-postRoute_xcap $vars(rc_max,pre_route_xcap_factor) \

-T $vars(rc_max,T) \

-qx_tech_file $vars(rc_max,qx_tech_file)

create_library_set \
```

```
-name fast \

-timing [list {$vars(fast,timing)} ]

create_library_set \

-name slow \

-timing [list {$vars(slow,timing)} ]

create_delay_corner -name fast_min -library_set fast \

-rc_corner rc_min

create_delay_corner -name slow_max -library_set slow \

-rc_corner rc_max

create_constraint_mode -name mode1 \

-sdc_files [list {$vars(mode1,pre_cts_sdc)}]

create_constraint_mode -name mode2 \

-sdc_files [list {$vars(mode2,post_cts_sdc)}]

if {[lsearch [all_analysis_views] hold_view] == -1} {

create_analysis_view -name hold_view \

-constraint_mode $vars(hold_view,constraint_mode) \

-delay_corner fast_min

}
```

```tcl
if {[lsearch [all_analysis_views] setup_view] == -1} {

create_analysis_view -name setup_view \

-constraint_mode $vars(setup_view,constraint_mode) \

-delay_corner slow_max

}

update_library_set -name fast -si [list $vars(fast,si)]

update_library_set -name slow -si [list $vars(slow,si)]

set active_corners [all_delay_corners]

if {[lsearch $active_corners slow_max] != -1} {

# begin derate.tcl

set_timing_derate -clock \

-cell_delay \

-early \

-delay_corner fast_min $vars(fast_min,clock_cell_early)

set_timing_derate -clock \

-cell_delay \

-late \

-delay_corner slow_max $vars(slow_max,clock_cell_late)
```

```
# <end derate.tcl>


}


set_analysis_view \


-setup [list setup_view] \


-hold [list hold_view]


set_default_view -setup setup_view -hold hold_view


set_power_analysis_mode -analysis_view setup_view


exec /bin/touch make/init


saveDesign DBS/init.enc -compress -def


report_power -outfile RPT/init.power.rpt


setDontUse $vars(dont_use_list) true


setDontUse $vars(use_list) false


setMaxRouteLayer 6


setDesignMode -process 90


addWellTap -cell [list $vars(welltaps)] \


-prefix WELLTAP \


-maxGap $vars(welltaps,max_gap) \


-inRowOffset $vars(welltaps,row_offset) \
```

```
-checkerboard

timeDesign -preplace -prefix preplace -outDir RPT

checkDesign -all check_timing

set vars(step) place
```

Variables affecting this step are listed under the "Place" section on the Example Settings for Each Script page.

```
setDesignMode -process 90

setPlaceMode -congEffort $vars(congestion_effort) \

-clkGateAware $vars(clock_gate_aware) \

-placeIoPins $vars(place_io_pins)

setOptMode -preserveAssertions false \

-powerEffort $vars(power_effort) \

-leakageToDynamicRatio medium$vars(leakage_to_dynamic_ratio) \

-clkGateAware $vars(clock_gate_aware) \

-criticalRange $vars(critical_range)

setTieHiLoMode -cell {$vars(tie_cells)} \

-maxDistance $vars(tie_cells,max_distance) \

-maxFanout $vars(tie_cells,max_fanout) cleanupSpecifyClockTree\ specifyClockTree -file
$vars(cts_spec) specifyJtag -inst $vars(jtag_cells)\ placeJtag -nrRow $vars(jtag_rows)
place_opt_design -inPlaceOpt foreach cell\ {$vars(tie_cells)} {
```

```
setDontUse $cell false

}

addTieHiLo

foreach cell {$vars(tie_cells)} {

setDontUse $cell true

}

exec /bin/touch make/place

saveDesign DBS/place.enc -compress

report_power -outfile RPT/place.power.rpt

set vars(step) prects
```

The variables affecting this step are: PreCTS on page 49

```
setDesignMode -process 90

cleanupSpecifyClockTree

specifyClockTree -file $vars(cts_spec)

specifyClockTree -update "AutoCTSRootPin * RouteClkNet YES

setUsefulSkewMode -delayPreCTS true \

-useCells [list $vars(skew_buffers)]
```

```
setOptMode -preserveAssertions false \

-powerEffort $vars(power_effort) \

-leakageToDynamicRatio medium$vars(leakage_to_dynamic_ratio) \

-clkGateAware $vars(clock_gate_aware) \

-criticalRange $vars(critical_range) \

-usefulSkew false

optDesign -preCTS -outDir RPT -prefix prects

exec /bin/touch make/prects

saveDesign DBS/prects.enc -compress

report_power -outfile RPT/prects.power.rpt

set vars(step) cts
```

The variables affecting this step are listed under the "CTS" section on the Example Settings for Each Script page.

```
setDesignMode -process 90

setCTSMode -routeClkNet true

setNanoRouteMode \

-routeWithLithoDriven $vars(litho_driven_routing) \

-drouteMultiCutViaEffort $vars(multi_cut_effort)

setOptMode -preserveAssertions false \
```

```
-powerEffort $vars(power_effort) \

-leakageToDynamicRatio medium$vars(leakage_to_dynamic_ratio) \

-clkGateAware $vars(clock_gate_aware) \

-criticalRange $vars(critical_range) \

-usefulSkew false

optDesign -preCTS -outDir RPT -prefix prects

exec /bin/touch make/prects

saveDesign DBS/prects.enc -compress

report_power -outfile RPT/prects.power.rpt

setDesignMode -process 90

setCTSMode -routeClkNet true

setNanoRouteMode \

-routeWithLithoDriven $vars(litho_driven_routing) \

-drouteMultiCutViaEffort $vars(multi_cut_effort)

set vars(step) postcts
```

The variables affecting this step are listed under the "PostCTS" section on the Example Settings for Each Script page.

```
setOptMode -postCtsClkGateCloning true

setAnalysisMode -cppr both
```

```
optDesign -postCTS -outDir RPT -prefix postcts


exec /bin/touch make/postcts


saveDesign DBS/postcts.enc -compress


report_power -outfile RPT/postcts.power.rpt
```

Sample Single Script Flow

```
source FF/vars.tcl


source FF/procs.tcl


 setDistributeHost -local


setMultiCpuUsage -localCpu max


 set vars(step) init


exec mkdir -p $env(VPATH)
```

See also,

Executing the Flow