

# **Voltus IC Power Integrity Solution Foundation Flows User Guide**

**Product Version 21.10  
May 2021**

© 2022 Cadence Design Systems, Inc. All rights reserved.  
Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

**Trademarks:** Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 1-800-862-4522.

All other trademarks are the property of their respective holders.

**Restricted Print Permission:** This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

The publication may be used solely for personal, informational, and noncommercial purposes;

The publication may not be modified in any way;

Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and

Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

**Disclaimer:** Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence's customer in accordance with, a written agreement between Cadence and its customer. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

**Restricted Rights:** Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.



---

# Contents

---

About This Manual	5
Related Documents	5
1	6
Getting Started With Voltus Foundation Flows	6
Overview	6
Setting up the Foundation Flow Environment	7
Understanding the Foundation Flow Framework	8
Foundation Flow Scripts	9
Guidelines for Editing Scripts	10
Defining Variables for the Power Environment	11
setup.tcl	11
Example of setup.tcl	14
voltus_config.tcl	15
Example of voltus_config.tcl	24
Customizing the Plug-in Scripts	25
Launching the Foundation Flow	28

# About This Manual

This manual describes the recommended flows using the Voltus IC Power Integrity Solution software to perform the Basic Power and IR Drop analysis.

The Voltus software encompasses the following products:

- Voltus IC Power Integrity Solution - L
- Voltus IC Power Integrity Solution - XL

For information about these products, see the "[Product and Licensing Information](#)" chapter of the *Voltus User Guide*.

## Related Documents

- [Voltus Known Problems and Solutions](#)  
Describes important Cadence Change Requests (CCRs) for Voltus, including solutions for working around known problems.
- [Voltus Text Command Reference](#)  
Describes the Voltus text commands, including syntax and examples.
- [Voltus Menu Reference](#)  
Provides information specific to the forms and commands available from the Voltus graphical user interface.
- [Voltus User Guide](#)  
Provides information on using various Voltus features.

---

# Getting Started With Voltus Foundation Flows

---

- [Overview](#)
- [Setting up the Foundation Flow Environment](#)
- [Understanding the Foundation Flow Framework](#)
  - [Foundation Flow Scripts](#)
- [Guidelines for Editing Scripts](#)
- [Defining Variables for the Power Environment](#)
  - [setup.tcl](#)
  - [Example of setup.tcl](#)
  - [voltus\\_config.tcl](#)
  - [Example of voltus\\_config.tcl](#)
- [Customizing the Plug-in Scripts](#)
- [Launching the Foundation Flow](#)

## Overview

The Foundation Flows are the Cadence-recommended procedures for performing Basic Power and IR Drop analysis using the Voltus IC Power Integrity Solution software. These flows support both CPF and non-CPF based designs. Cadence provides a set of scripts that give you an easy way to automate these flows. This document describes these scripts, and how to set up, customize, and run the scripts.

The Foundation Flows are a starting point for building Power and Rail Analysis Signoff environment, but you can augment them with design-specific content.

In Voltus, you have four stand-alone analysis foundation flows and a PG library generation flow. These are:

- Static Power Analysis
- Dynamic Power Analysis
- Static Rail Analysis
- Dynamic Rail Analysis (including power-up analysis)
- PG library generation (used when no PG library is provided)

Each flow can either load an Innovus database or can source some other input files. For non-Innovus based flow, various input files such as design data and library information are sourced and the design is loaded. You also need to load some other input files such as SDF, TWF, or IR drop files for all type of flows.

The requirements to load an Innovus database are:

- Innovus design database
- Parasitic information file
- Power-grid views for all cells and macros

The requirements to load a non-Innovus or third party design database are:

- Design DEF file
- Design Verilog netlist file
- Parasitic information file
- CPF file (only for low power designs)
- Timing and Physical libraries (.lib and .lef)
- Power-grid views for all cells and macros

## Setting up the Foundation Flow Environment

Using the command prompt, you can create a foundation flow environment by performing the following steps:

1. Set the run-time environment by using the following command:

```
set path = (<install_dir>/tools/bin $path)
```

2. Launch the Voltus console in the non-GUI environment using the -nowin parameter.

```
voltus -nowin
```

3. Run the `write_flow_template` command.

```
write_flow_template -directory .
```

This command copies the Voltus Foundation Flows templates into the directory you specify (or to the current directory if you do not specify a directory with the `-directory` parameter).

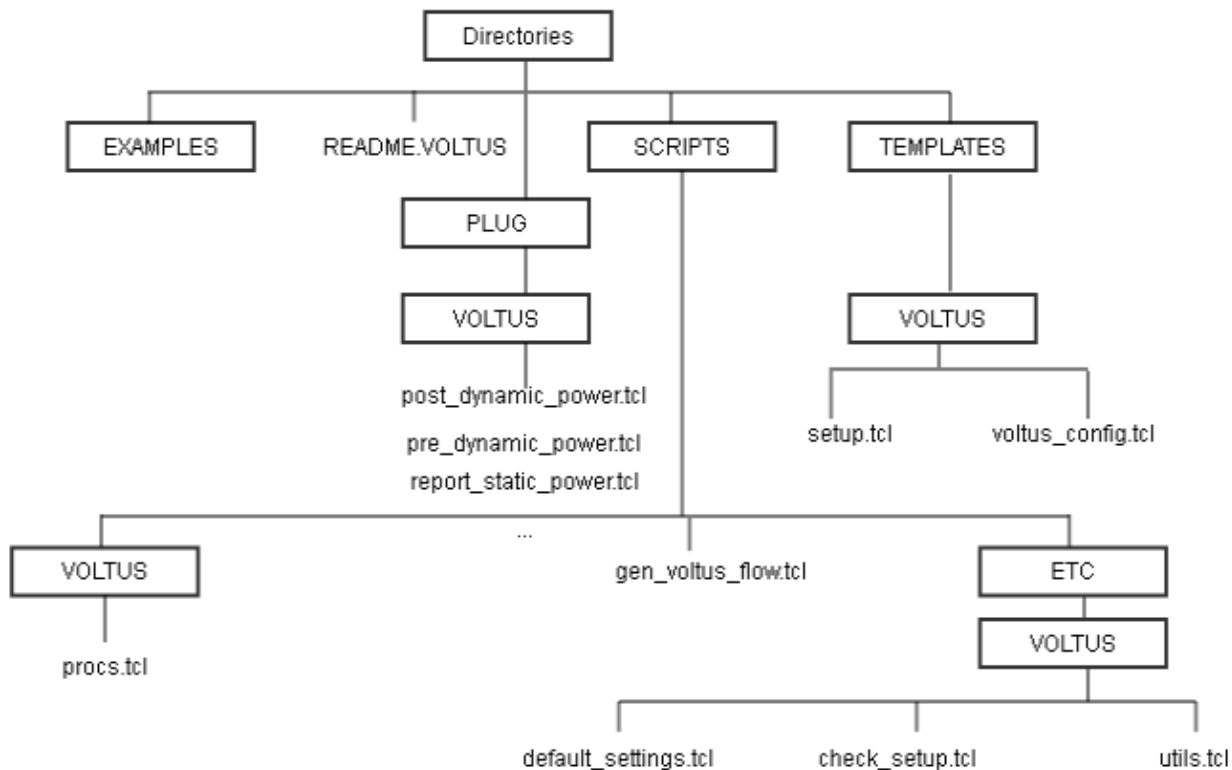
4. Exit the Voltus software.

```
exit
```

You can access the Voltus foundation flow directory to verify if the README file, and the PLUG, EXAMPLES, SCRIPTS, and TEMPLATES folders are created.

## Understanding the Foundation Flow Framework


The following diagram illustrates the directory structure of the Voltus foundation flows:





The Voltus foundation flow directory consists of:

- **EXAMPLES** - a folder containing example scripts for a sample design. These scripts have design specific content that explains how to set up the scripts.
- **README.Voltus** - a file containing the directory structure of the foundation flow, the content of each directory, and how to use the foundation flows.
- **PLUG** - Contains Voltus plug-in scripts. These scripts provide an easy method to expand and customize the basic flows.
- **SCRIPTS** - a folder containing the Voltus foundation flow related scripts.
- **TEMPLATES** - a folder containing the `setup.tcl` and `voltus_config.tcl` files that are required to create and configure the flow environment. You must copy these two files to your run directory and edit them.

 You can complete the flow by specifying the variables in the `voltus_config.tcl` file. To customize the flow, enable the plug-ins in the `voltus_config.tcl` file, and update the corresponding plug-in scripts in the PLUG folder.

## Foundation Flow Scripts


This section discusses the Voltus foundation flow scripts:

- `TEMPLATES/Voltus/setup.tcl` - A single source of all design data required to run your design.
- `TEMPLATES/Voltus/voltus_config.tcl` - A configuration file to set up the flow related variables, to enable plug-in variables, and to configure the setup for multi-CPU processing.
- `SCRIPTS/gen_voltus_flow.tcl` - A file used to generate the Voltus flow (flattened scripts) based on the settings made in the `setup.tcl`, `voltus_config.tcl`, and plug-in script files.
- `SCRIPTS/ETC/Voltus/check_setup.tcl` - A script to perform sanity checking and verification of the `setup.tcl` and `voltus_config.tcl` files.
- `SCRIPTS/ETC/Voltus/utils.tcl` - Common procedures for all foundation flows (Innovus/Tempus/Voltus)
- `SCRIPTS/ETC/Voltus/default_settings.tcl` - The default settings for the Voltus flow.
- `PLUG/Voltus` - This folder contains the plug-in scripts used to customize a flow as all commands/parameters are not covered in `voltus_config.tcl`.

# Guidelines for Editing Scripts

Follow the guidelines below to edit the `setup.tcl`, `voltus_config.tcl`, and plug-in script files for your environment:

- The hash (#) symbol indicates that the line (variable/command/parameter) is a comment and does not get read by the software. All the mandatory variables/commands/parameters have been left uncommented. You can uncomment a variable/command/parameter by deleting the hash (#) at the beginning of the line.
- The script for each flow has the default or recommended settings. You can modify these default settings.
- Any content within angled brackets or curly braces indicate information for which you must edit or substitute a value.


 For content within angled brackets, you must ensure that you delete the angled brackets after editing the value.

You can edit the following example script to set the transition density of the pin i1/A to 100 transitions per second:

```
#set_switching_activity
# -pin <pinname>
# -activity <activity factor>
# -period <period value>
# -density <density factor>
# -duty <duty factor>
```

The modified example is given below:

```
set_switching_activity
-pin i1/A
-density 100
```

 For content within curly braces, you must ensure that you retain the curly braces after editing the value.

You can edit the following example script to add the list of decap cells and pg nets:

```
set_power_analysis_mode
-method static
```

```
-create_binary_db true
-binary_db_name StaticPower.db
-write_static_currents true
-decap_cell_list {list of decap cells}
-off_pg_nets {list of off pgnets}
```

The modified example is given below:

```
set_power_analysis_mode
-method static
-create_binary_db true
-binary_db_name StaticPower.db
-write_static_currents true
-decap_cell_list {DCAP* }
-off_pg_nets {VDD}
```

## Defining Variables for the Power Environment

- setup.tcl
- voltus\_config.tcl

### setup.tcl

Defines the variables that are used 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, MMMC configuration, and PGV.

The following variables are defined in setup.tcl. They are put into an array named vars.

### Defining variables to point data, libraries, reports and scripts

```
set vars(script_root) SCRIPTS
set vars(plug_root) PLUG/Voltus
set vars(rpt_dir) reports
set vars(log_dir) LOG
set vars(data_root) "DATA"
set vars(libs_root) "DATA/libs"
```

## Defining the design data

- **Innovus Design** - set this variable for an Innovus design database with routed information. You can set this variable by uncommenting it and giving the path of the Innovus database.

```
# set vars(edi_db_name)      <EDI_design_path>
# set vars(load_edi_db)      "true"
```

- **Non-Innovus Design** - set the following variables for a non-Innovus design database with third-party information.

```
# set vars(netlist)          <verilog netlist file>
# set vars(def_files)        <def file>
# set vars(sdc_files)        <timing constraints>
# set vars(design)           <top_design>
```

**Note:** If you are providing an Innovus database, you should comment these variables.

## Ignore undefined cells while loading the design

```
# set vars(ignore_undefined_cell) "<true/false>"; # default is "false"
# set vars(ignore_timing_library_check) "<true/false>"; # must be false for power
```

## Specifying the SPEF file, if the design is non-MMMC

```
# set vars(spef) <spef only for non-mmmc design>
```

## Supported flows -> default or mmmc

- **Default**

```
#set vars(flow)      "default"
```

- **MMMC** - set the following variables only for an MMMC design:

```
# set vars(cpf_file) <CPF_file> # only for CPF-based MMMC design
# set vars(mmmc_setup_file) <view_definations_file>
# set vars(rc_corners) "<corner1> <corner2> ..."
# set vars(<corner1>,spef) <corner1_spef>
# set vars(<corner2>,spef) <corenr2_spef>
```

```
# set vars(analysis_views) "<view1> <view2>"
```

## Defining an active analysis view, if only one view is required to be active

```
#set vars(active_analysis_view) "<active view>"
```

## Defining library sets for a non-Innovus database

```
# set vars(library_sets) "max min"
# set vars(max,timing) <list of lib files>
# set vars(min,timing) <list of lib files>

# set vars(lef_files) <list of lef files> #Define LEF files
# set vars(cl_views) <list of .cl files> # Define Power-Grid views
```

**Note:** If you are providing an Innovus database, you should comment these variables.

## Defining PGV sets required for creation of power-grid views

- Variables for common power-grid creation

```
# set vars(extraction_tech_file)      <technology file>
# set vars(lef_layermap)              "<file>"
```

- Specify advanced options of the command `set_advanced_pg_library_mode`, like `"-default_frequency"`

```
# set vars(advanced_pg_library_mode)  "<advanced options>" ;
```

- Variables for techonly pg

```
# set vars(techonly_pg_creation)      "<true/false>"      ;# set to true to create
techonly power-grid library based on defined variables
# set vars(techonly_ground_pins)      "<ground_pin_list>"
# set vars(techonly_power_pins)       "<pin1 voltage1 ... pinN voltageN>"
# set vars(techonly_other_options)    "<other options>"      ;
# set vars(techonly_outdir)           "<dir_name>"
# set vars(techonly_prefix)           "<prefix>"            ;
```

- Variables for std cells

```
# set vars(stdcells_pg_creation)      "<true/false>"      ;# set to true to create
```

### stdcells power-grid library based on defined variables

```
# set vars(stdcells_ground_pins)      "<ground_pin_list>"
# set vars(stdcells_power_pins)       "<pin1 voltage1 ... pinN voltageN>"
# set vars(stdcells_spice_models)     "<file_list>" ;
# set vars(stdcells_spice_subckts)    "<file_list>"
# set vars(stdcells_other_options)    "<other options>" ; # specify optional options
# set vars(stdcells_outdir)           "<dir_name>"
# set vars(stdcells_prefix)           "<prefix>" ;
```

- Variables for macro cells

```
# set vars(macros_pg_creation)        "<true/false>" # set to true to create MACRO
```

### power-grid library based on defined variables

```
# set vars(macros_ground_pins)        "<ground_pin_list>"
# set vars(macros_power_pins)         "<pin1 voltage1 ... pinN voltageN>"
# set vars(macros_cell_list_file)     "<built_cells_list>"
# set vars(macros_gds_files)          "<gds_file_list>"
# set vars(macros_gds_layermap)       "<gds_layermap_file>"
# set vars(macros_spice_models)       "<file_list>"
# set vars(macros_spice_subckts)      "<file_list>"
# set vars(macros_other_options)      "<other options>" ; # specify optional options
# set vars(macros_outdir)             "<dir_name>"
# set vars(macros_prefix)             "<prefix>" ;
```

## Example of setup.tcl

The following example shows the minimum design data settings required to run a non-Innovus design:

```
set vars(script_root) SCRIPTS
set vars(plug_root)    PLUG/Voltus
set vars(rpt_dir)      RPT
set vars(log_dir)      LOG
set vars(data_root)    "DATA"
set vars(libs_root)    "DATA/libs"

set vars(edb_name)      $vars(data_root)/srio_mac_assembled.enc
#set vars(load_edb)     "true"
set vars(netlist)       $vars(data_root)/srio_mac.v.gz
set vars(def_files)     $vars(data_root)/srio_mac_assembled.def
set vars(sdc_files)     $vars(data_root)/db_default_4.2-USR2.sdc
set vars(design)        "srio_mac"
set vars(spef)          $vars(data_root)/srio_mac_assembled_rc_worst.spef.gz
```

```
#set vars(cpf_file)           $vars(data_root)/chip_top_1.0e_integrated_for_backend.cpf
set vars(mmmc_setup_file)     $vars(data_root)/viewDefinition.tcl
set vars(rc_corners)          "rc_worst rc_best"
set vars(rc_worst,spef)       $vars(data_root)/srio_mac_assembled_rc_worst.spef.gz
set vars(rc_best,spef)        $vars(data_root)/srio_mac_assembled_rc_best.spef.gz
set vars(analysis_views)      "setup_view1 hold_view1 setup_view2 hold_view2"

#set vars(active_analysis_view) [lindex $vars(analysis_views) 0]

set vars(library_sets) "max min"
set vars(max,timing) "\
    $vars(libs_root)/tse_rf256x8p1_slow_syn.lib \
    $vars(libs_root)/tse_ra280x67p2_slow_syn.lib \
"
set vars(min,timing) "\
    $vars(libs_root)/tse_rf256x8p1_fast-40C_syn.lib \
    $vars(libs_root)/tse_ra280x67p2_fast-40C_syn.lib \
"
set vars(lef_files) "\
    $vars(libs_root)/tse_ra280x67p2.vclef \
    $vars(libs_root)/tse_rf256x8p1.vclef \
"
set vars(cl_views) "\
    $vars(libs_root)/fast_allcells.cl \
"
puts "<FF> Finished loading setup.tcl"
```

## **voltus\_config.tcl**

This is an optional configuration file containing flow related information that is unique to a particular block or run. The following variables are defined in `voltus_config.tcl`. They are put into an array named `vars`.

## Specifying the analysis type

```
#set vars(generate_pg)      "true" # set to true to add PG library generation flow before power
and rail analysis
#set vars(pg_only)          "true" # set to true to run only the PG library generation flow
#set vars(static_power)     "true"
#set vars(static_rail)      "true"
#set vars(dynamic_power)    "true"
#set vars(dynamic_rail)     "true"
```

## Specifying the Power Analysis Parameters

Specify the global switching activity for all primary inputs, nets, and other instances whose activity has not been defined through TCF or VCD files. The following are the options to define the switching activity of the design:

```
# set vars(input_activity)    "<primary input activity>" (OPTIONAL)
# set vars(seq_activity)      "<activity at sequential logic output>" (OPTIONAL)
# set vars(global_activity)    "<activity for all unset nodes>" (OPTIONAL)
```

## Specifying the Power Output Directory

# Specify the directory path to write the power reports and current files for both the static and dynamic. The default directory for static power analysis: \$vars(rpt\_dir)/static\_power. The default directory for dynamic power analysis: \$vars(rpt\_dir)/dynamic\_power.

```
# set vars(static_power_reports) "<static_power_directory_name>" (OPTIONAL)
# set vars(dynamic_power_reports) "<dynamic_power_directory_name>" (OPTIONAL)
```

## Specifying the Static Power Calculation Parameters

```
# set vars(static_power,analysis_view) "$vars(active_analysis_view)" # specify the power
analysis view for the MMMC design
# set vars(static_power,corner) "min/max" # for a non-MMMC design, define the power
analysis library corner, default:max
# set vars(static_power,create_binary_db) "true/false" # set true to create power analysis
binary db, default:false
# set vars(static_power,transition_time_method) "min/avg/max" # set the transition time
method, default:max
# set vars(static_power,write_static_currents) "true/false" # set true to write the static
```



current files, default:false

## Specifying the Dynamic Power Calculation Parameters

```
# set vars(dynamic_power,method) "dynamic_vectorless/dynamic_vectorbased" # Specify the
method of analysis to be performed, default:dynamic_vectorless
# set vars(dynamic_power,analysis_view) "$vars(active_analysis_view)" # specify the
power analysis view for MMMC setup
# set vars(dynamic_power,corner) "min/max" # For non-MMMC designs define the library
corner, default:max
# set vars(dynamic_power,create_binary_db) "true/false" # set true to create power analysis
binary db, default:false
# set vars(dynamic_power,transition_time_method) "min/avg/max" # set the transition time
method, default:max
# set vars(dynamic_power,disable_static) "true/false" # To perform Static Power Analysis
during Dynamic, default:true
# set vars(dynamic_power,write_static_currents) "true/false" # To write the static current
files, default:false
```

## Power Include File for Dynamic Power Calculation (Optional)

```
# set vars(dynamic_power_inc_file) "<power_include_file.inc>" # All additional Power
Analysis options, which do not have an equivalent option in Voltus, can be given through this
include file. For more information, refer to "Power Analysis Options" in Voltus Text Command
Reference.
```

## Common Parameters Required for Rail Analysis

```
# set vars(rail_analysis_temp_dir)  "./tmp" # specify the analysis temporary directory path
# set vars(power_domains)           "<domain1 domain2>" # specify the power domains in the design
# set vars(<domain1>,pwr_nets) "<pwr_net1 pwr_net2 ...>" # specify list of power & ground
nets under each power domain (Required)
# set vars(<domain1>,gnd_nets) "<gnd_net1 gnd_net2 ...>"
# set vars(<pg_net1>,voltage) "<supply_voltage>" # specify the supply voltage for the power
nets
# set vars(threshold_percent) "<threshold_percent>" # specify the voltage threshold in
percentage (Optional) default is 5
# set vars(<pg_net1>,threshold) "<user_specified_threshold_value>" # specify the
threshold voltage per P/G net (Optional) else the tool evaluates
# set vars(supply_tolerance_percent) "<supply_tolerance>" # specify power supply
tolerance in percentage (Optional) default is 30
# set vars(<pg_net1>,tolerance) "<user_specified_tolerance_value>" # specify the
supply tolerance per P/G net (Optional) else the tool evaluates
```

## Static Rail Analysis Parameters

```
# set vars(static_rail,analyze_domains) "<domain1 domain2 ....>" # specify the list of
power domains to be analyzed for static rail (Required)
# set vars(static_rail,analyze_type) "<net/domain>" # specify the type of rail analysis to
be performed, default:domain
# set vars(static_rail,accuracy) "fast/fast_accurate/accurate" # specify the
accuracy of the analysis, default:fast
# set vars(static_rail,analysis_view) "$vars(active_analysis_view)" # specify the
analysis view only for MMMC designs
# set vars(static_rail,temperature) "<user_set_temp>" # specify the analysis
temperature, default:25
# set vars(static_rail,dist_solver_processing) "true/false" # enable parallel distributed
process for solver, default:false (Optional)
# set vars(static_rail,gen_bb_voltage_file) "<list of instance names>" # write the block
boundary interface nodes voltage to the file specified (Optional)
# set vars(static_rail,ignore_shorts) "true/false" # ignore shorts during signoff mode of
analysis, default:false (Optional)
# set vars(static_rail,cell_ignore_file) "<cell_ignore_file>" # cells to be ignored
during rail analysis (Optional)
# set vars(static_rail,fast_views_list) "<fast_views_cell_file>" # list of fast PGVs to
be used for rail analysis (Optional)
# set vars(static_rail,fast_accurate_views_list) "<fast_accurate_views_cell_file>"
# list of fast_accurate PGVs to be used for rail analysis (Optional)
# set vars(static_rail,accurate_views_list) "<accurate_views_cell_file>" # list of
accurate PGVs to be used for rail analysis (Optional)
# set vars(static_rail,ext_inc_file) "<extractor_include_file>" # specify extractor
include file to control ZX extractor (Optional)
# set vars(static_rail,dis_analysis_types) "<list analysis types to be disabled>" #
disable analysis types (Optional)
# set vars(static_rail,<pwr_net1>,pti_file) "<pti_current_file>" # specify PTI
current files for all the PG nets (Optional)
# set vars(static_rail,<pwr_net1>,ascii_file) "<ascii_file>" # specify ASCII format
instance power file (Optional)
# set vars(static_rail,<pwr_net1>,scale) "<real_number>" # specify scaling of the
power/current data provided (Optional) default is 1.00
```

## Vstorm Include File for Static Rail Analysis (Optional)

The Vstorm include file with additional Vstorm options, which are not supported by Voltus. All additional Vstorm options can be given through this include file.

```
# set vars(static_rail,vstorm2_begin_file) "<vstorm_begin_file.inc>" # The Vstorm
options included before "analyze_rail"
# set vars(static_rail,vstorm2_end_file) "<vstorm_end_file.inc>" # The Vstorm options
included after "analyze_rail"
```

## ElectroMigration (EM) Model File (Optional)

```
# set vars(em_models_file) "<em_models_file>" # This file is required for
electromigration(em)/current_density(rj) analysis
```

## Power Pads Location Files

```
# set vars(<pg_net1>,format) "defpin" # Specify the format of the pad locations
"defpin/xy/padcell/boundary"
# set vars(<pg_net1>,pad_file) "<net1_pp_location_file1.pp \
# net1_pp_location_file2.pp \
# net1_pp_location_file3.pp>" # Specify the power pad location
files of all the analysis nets
```

## Dynamic Rail Analysis Parameters

```
# set vars(dynamic_rail,analyze_domains) "<domain1 domain2 ....>" # specify the list of
power domains to be analyzed for dynamic rail (Required)
# set vars(dynamic_rail,analyze_type) "net/domain" # specify the type of rail analysis to be
performed, default:domain
# set vars(dynamic_rail,accuracy) "fast/fast_accurate/accurate" # specify the accuracy of
the analysis, default:fast
# set vars(dynamic_rail,analysis_view) "$vars(active_analysis_view)" # specify the
analysis view only for MMMC designs
# set vars(dynamic_rail,temperature) "<user_set_temp>" # specify the analysis
temperature, default:25
# set vars(dynamic_rail,dist_solver_processing) "true/false" # enable parallel distributed
process for solver, default:false (Optional)
# set vars(dynamic_rail,powering_up_nets) "<switchable_PGnets>" # specify the list of
```

powering up nets for dynamic rail and power-up analysis

```
# set vars(dynamic_rail,gen_power_switch_eco) "true/false" # generate power switch Eco
file, default:false (Optional)
# set vars(dynamic_rail,gen_decap_eco) "true/false" # generate decap Eco file,
default:false (Optional)
# set vars(dynamic_rail,decap_opt_method) "feasibility" # decap opt method, can be
"area/feasibility/timing/removal/feasibility_removal" (Optional)
# set vars(dynamic_rail,decap_removal_method) "<user_specified>" # decap removal
method, can be "conservative/aggressive" (Optional)
# set vars(dynamic_rail,save_voltage_waveforms) "false" # save voltage waveforms, can be
"true/false" (Optional)
# set vars(dynamic_rail,gen_bb_voltage_file) "<list of instance names>" # write the block
boundary interface nodes voltage to the file specified (Optional)
# set vars(dynamic_rail,save_current_files) "true/false" # save the current files for
dynamic hierarchical power_view creation, default:false (Optional)
# set vars(dynamic_rail,ignore_shorts) "true/false" # ignore shorts during signoff mode of
analysis, default:false (Optional)
# set vars(dynamic_rail,decap_eco_file) "<decap_eco_file>" # decap ECO file to be
considered for dynamic ir drop analysis (Optional)
# set vars(dynamic_rail,cell_ignore_file) "<cell_ignore_file>" # cells to be ignored
during rail analysis (Optional)
# set vars(dynamic_rail,fast_views_list) "<fast_views_cell_file>" # list of fast PGVs to
be used for rail analysis (Optional)
# set vars(dynamic_rail,fast_accurate_views_list) "<fast_accurate_views_cell_file>"
# list of fast_accurate PGVs to be used for rail analysis (Optional)
# set vars(dynamic_rail,accurate_views_list) "<accurate_views_cell_file>" # list of
accurate PGVs to be used for rail analysis (Optional)
# set vars(dynamic_rail,ext_inc_file) "<extractor_include_file>" # specify extractor
include file to control ZX extractor (Optional)
# set vars(dynamic_rail,dis_analysis_types) "<list analysis types to be disabled>" #
disable analysis types (Optional)
# set vars(dynamic_rail,<pwr_net1>,pti_file) "<pti_current_file>" # specify pti
current files for all the PG nets (Optional)
# set vars(dynamic_rail,<pwr_net1>,scale) "<real_number>" # specify scaling of the
power/current data provided (Optional) default is 1.00
```

## Vstorm Include File for Dynamic Rail Analysis (Optional)

The Vstorm include file with additional Vstorm options, which are not supported by Voltus. All additional Vstorm options can be given though this include file.

```
# set vars(dynamic_rail,vstorm2_begin_file) "<vstorm_begin_file.inc>" # The Vstorm
options included before "analyze_rail"
# set vars(dynamic_rail,vstorm2_end_file) "<vstorm_end_file.inc>" # The Vstorm options
included after "analyze_rail"
```

## Voltus Tool Control Settings

```
# set vars(distribute) "local/rsh/lsf/custom" # set the distribution process
type, default:local
# set vars(local_cpus) "<num_local_cpus>" # set the number of local CPUs, set
any integer_number or "max", default:0
# set vars(remote_hosts) "<number_of_hosts>" # set number of remote hosts for
distribute type non-local
# set vars(cpu_per_remote_host) "<number_of_cpus_per_host>" # set number of CPUs per
host for distribute type non-local
# set vars(rsh,host_list) "<host1 host2 host2 .....>" # set host list for distribute
type "rsh"
# set vars(lsf,queue) "<queue_name>" # set lsf queue name for distribute type "lsf"
# set vars(lsf,resource) "<lsf_resource_string>" # set lsf resource string for
distribute type "lsf"
# set vars(lsf,args) "<lef_args>" # set lsf args for distribute type "lsf"
# set vars(custom,script) "<specify_lsf_custom_script>" # specify the lsf custom
scripts for distribute type "custom"
# set vars(keep_license) "true/false" # specify whether to keep the multiple CPU-
licenses, default:true (Optional)
```

## Supported Plug-ins

```
#set vars(set_advanced_pg_library_mode_tcl)
$vars(plug_root)/set_advanced_pg_library_mode.tcl
#set vars(user_load_design_tcl) $vars(plug_root)/user_load_design.tcl
#set vars(pre_static_power_tcl) $vars(plug_root)/pre_static_power.tcl
#set vars(report_static_power_tcl)
$vars(plug_root)/report_static_power.tcl
#set vars(post_static_power_tcl) $vars(plug_root)/post_static_power.tcl
#set vars(pre_dynamic_power_tcl) $vars(plug_root)/pre_dynamic_power.tcl
#set vars(post_dynamic_power_tcl)
$vars(plug_root)/post_dynamic_power.tcl
#set vars(set_static_rail_mode_tcl)
$vars(plug_root)/set_static_rail_mode.tcl
#set vars(set_static_rail_pg_nets_tcl)
$vars(plug_root)/set_static_rail_pg_nets.tcl
#set vars(set_static_rail_power_data_tcl)
$vars(plug_root)/set_static_rail_power_data.tcl
#set vars(set_static_rail_pad_location_tcl)
$vars(plug_root)/set_static_rail_pad_location.tcl
#set vars(set_static_rail_analysis_domain_tcl)
$vars(plug_root)/set_static_rail_analysis_domain.tcl
#set vars(pre_static_rail_tcl) $vars(plug_root)/pre_static_rail.tcl
#set vars(post_static_rail_tcl) $vars(plug_root)/post_static_rail.tcl
#set vars(set_dynamic_rail_mode_tcl)
$vars(plug_root)/set_dynamic_rail_mode.tcl
#set vars(set_dynamic_rail_pg_nets_tcl)
$vars(plug_root)/set_dynamic_rail_pg_nets.tcl
#set vars(set_dynamic_rail_power_data_tcl)
$vars(plug_root)/set_dynamic_rail_power_data.tcl
#set vars(set_dynamic_rail_pad_location_tcl)
$vars(plug_root)/set_dynamic_rail_pad_location.tcl
#set vars(set_dynamic_rail_analysis_domain_tcl)
$vars(plug_root)/set_dynamic_rail_analysis_domain.tcl
#set vars(pre_dynamic_rail_tcl) $vars(plug_root)/pre_dynamic_rail.tcl
#set vars(post_dynamic_rail_tcl) $vars(plug_root)/post_dynamic_rail.tcl

Puts "<FF> Finished loading voltus_config.tcl"
```

## Example of voltus\_config.tcl

The following example shows the analysis types that are to be run, and the settings for running the multi-CPU flow:

```
set vars(input_activity) "0.25"
set vars(seq_activity)   "0.25"

#set vars(static_power_reports)      "RPT/STATIC_POWER"
#set vars(dynamic_power_reports)     "RPT/DYNAMIC_POWER"

set vars(static_power,analysis_view) $vars(active_analysis_view)
set vars(static_power,write_static_currents) "true"

set vars(dynamic_power,analysis_view) $vars(active_analysis_view)
#set vars(dynamic_power,method) "dynamic_vectorbased"
#set vars(dynamic_power_inc_file) $vars(script_root)/dynamic_power.inc

set vars(rail_analysis_temp_dir)      "./tmp"
set vars(power_domains)               "domain1"
set vars(domain1,pwr_nets)            "vdd\!"
set vars(domain1,gnd_nets)            "gnd\!"
set vars(vdd\!,voltage)               1.08
#set vars(threshold_percent)           2.5
#set vars(supply_tolerance_percent)    40

set vars(static_rail,analyze_domains) "domain1"
#set vars(static_rail,analyze_type)    "net"
set vars(static_rail,analysis_view)    $vars(active_analysis_view)
#set vars(static_rail,accuracy)         "fast"
#set vars(static_rail,temperature)      "125"
#set vars(static_rail,vdd\!,scale)      "5.00"
#set vars(static_rail,gnd\!,scale)      "10.00"
#set vars(static_rail,gnd\!,pti_file)
$vars(rpt_dir)/STATIC_POWER/static_gnd\!.ptiavg
#set vars(static_rail,vdd\!,ascii_file)
$vars(rpt_dir)/static_power/Instance_pin_power.rpt
#set vars(static_rail,ext_inc_file)      ./extractor_include_file.txt
# set vars(static_rail,vstorm2_begin_file) "<vstorm_begin_file.inc>"
# set vars(static_rail,vstorm2_end_file)  "<vstorm_end_file.inc>"

set vars(vdd\!,format)                 "defpin"
```



```
set vars(gnd\!,format)      "defpin"

set vars(dynamic_rail,analyze_domains)      "domain1"
set vars(dynamic_rail,analysis_view)        $vars(active_analysis_view)
#set vars(dynamic_rail,accuracy)            "fast"
#set vars(dynamic_rail,temperature)         125
#set vars(dynamic_rail,vdd\!,scale)         "10.00"
#set vars(dynamic_rail,gnd\!,scale)         "10.00"
#set vars(dynamic_rail,cell_ignore_file)    ./cell_ignore_list.txt

#set vars(distribute)            lsf
#set vars(remote_hosts)          2
#set vars(cpu_per_remote_hosts)  2
#set vars(lsf,queue)             "rnd"
#set vars(remote_hosts)          3
#set vars(cpu_per_remote_host)   2
#set vars(rsh,host_list)         "kitsopt11 kitsopt11 kitsopt12 kitsopt12 kitsopt13
kitsopt13"
#set vars(keep_license)          "false"
#set vars(local_cpus)            "4"
Puts "<FF> Finished loading voltus_config.tcl"
```

## Customizing the Plug-in Scripts

Based on the flow you want to run, you can customize the following plug-in scripts that contain the software-level directives:

Plug-in Script	Description
user_load_design.tcl	This plug-in loads the design. You can specify all the design loading commands in this file to load the design for analysis.
pre_static_power.tcl	This plug-in is called before executing the <code>report_power</code> command for specifying settings before static power calculation.
report_static_power.tcl	This plug-in executes the static power run and writes out various text-based reports

post_static_power.tcl	This plug-in is called after executing the <code>report_power</code> command. It includes commands to view the static power analysis reports for debugging.
pre_dynamic_power.tcl	This plug-in is called before executing the <code>report_power</code> command for specifying setting before dynamic power calculation.
post_dynamic_power.tcl	This plug-in is called after executing the <code>report_power</code> command. It includes commands to view the dynamic power analysis reports for debugging.
set_static_rail_mode.tcl	This plug-in is called in place of the <code>set_rail_analysis_mode</code> command.
set_static_rail_pg_nets.tcl	This plug-in is called in place of the <code>set_pg_nets</code> command.
set_static_rail_power_data.tcl	This plug-in is called in place of the <code>set_power_data</code> command.
set_static_rail_pad_location.tcl	This plug-in is called in place of the <code>set_power_pads</code> command.
set_static_rail_analysis_domain.tcl	This plug-in is called in place of the <code>set_rail_analysis_domain</code> command.
pre_static_rail.tcl	This plug-in is called before executing the <code>analyze_rail</code> command for specifying settings before static rail analysis.
post_static_rail.tcl	This plug-in is called after executing the <code>analyze_rail</code> command. It includes commands to view the static rail analysis reports for debugging.
set_dynamic_rail_mode.tcl	This plug-in is called in place of the <code>set_rail_analysis_mode</code> command.
set_dynamic_rail_pg_nets.tcl	This plug-in is called in place of the <code>set_pg_nets</code> command.
set_dynamic_rail_power_data.tcl	This plug-in is called in place of the <code>set_power_data</code> command.

set_dynamic_rail_pad_location.tcl	This plug-in is called in place of the <code>set_power_pads</code> command.
set_dynamic_rail_analysis_domain.tcl	This plug-in is called in place of the <code>set_rail_analysis_domain</code> command.
pre_dynamic_rail.tcl	This plug-in is called before executing the <code>analyze_rail</code> command for specifying settings before dynamic rail analysis.
post_dynamic_rail.tcl	This plug-in is called after executing the <code>analyze_rail</code> command. It includes commands to view the dynamic rail analysis reports for debugging.

When the plug-in scripts are enabled in `voltus_config.tcl`, the scripts are either inserted into the `run_voltus.tcl` (the final run scripts), or will replace some of the commands in `run_voltus.tcl`, without any check by foundation flows. This means if there are any incorrect settings in the plug-in, the error is reported by the tool only when the script is being executed, rather than by the foundation flows in advance. This is why the plug-ins do not support the incremental mode.

All possible commands/options are listed in the plug-in with commented commands for reference, as shown below:

```
#####

# set_power can be used to specify design/cell/instance power
# set_power 1500mw. This will set the total chip power to 1500mw
# set_power -type cell NAND* 5mw. This will set the power of all cells which match the
pattern NAND* to 5mw
# set_power -type instance ram_256x12 15mw. This will set the power of instance
ram_256x12 to 15mw
# set_power -type cell DCAP* -leakage 2mw. This will set the leakage power for cells
which match the pattern DCAP* as 2mw

#####

#set_power -reset
#set_power <chip power>mw
#set_power -cell <cellname> <power>mw
#set_power -instance <instname> <power>mw
#set_power -cell <decap cell names> -leakage <power>mw
```

Because of the plug-in feature mentioned above, these lines will be copied to `run_voltus.tcl` without any filter. If you do not need these commands/options and want a clean `run_voltus.tcl`, you can remove them from the plug-in.

# Launching the Foundation Flow

1. Copy the content in ./TEMPLATES/ to your run directory.

```
cp -r ./Voltus_FF/TEMPLATES/* <Run_directory>
```

2. Modify the following files before generating the code for Voltus analysis:

- a. `setup.tcl` - Update the script with the design data. You can edit this file to set up the design file variables.
- b. `voltus_config.tcl` - Select the flows (Voltus analysis variables) that you want to run. You need to explicitly set the variables to `true` to run the analysis type. If these flows do not meet your requirements, you can enable the plug-ins variables at the end of the file by uncommenting the required variable.
- c. (Optional step) If you enable a plug-in variable, you must copy the plug-in directory to your run directory, as shown below:

```
cp -r ./PLUG/ <Run_directory>.
```

Then, modify the corresponding plug-in files in PLUG/Voltus. You must add full Voltus commands in the appropriate plug-in files for the controls that were not covered in `voltus_config.tcl`.

**Note:** You can change your plug-in directory using the `setup.tcl` variable `vars(plug_root)`.

3. Use the following command to generate the flattened code.

```
tclsh SCRIPTS/gen_voltus_flow.tcl
```

This command generates the "FF/Voltus" directory in the run directory. This directory contains the flattened Voltus Foundation Flow scripts to run the flow. This command also generates the "Makefile" to run the flow.

Check the "FF/voltus.check.rpt" file for the setup status. If this code generation report file ends with "Setup Check Passed", you can proceed with the next step. You can also browse the "FF/Voltus/run\_Voltus.tcl" to understand how and when the plug-in is applied.

**Note:** The Foundation Flow will detect most of the errors in the variables you defined, and error out in this step.

4. Use the following commands to run the selected Voltus analysis:

- a. Link the Voltus makefile to Makefile:

```
ln -s Makefile.Voltus Makefile
```

- b. Run the Voltus analysis on the design for the analysis set in the "voltus\_config.tcl" file:

```
make flow
```

You can also run the optional command `make help` to show the usage of the make file.

This command generates the following directories in the run directory.

- LOG - contains log reports for Voltus analysis
- RPT - contains Voltus analysis output reports