About IEEE 1801-2009

The power supplied to elements in an electronic design affects the way circuits operate. Today's set of high-level design languages have not had a consistent way to concisely represent the regions of a design with different power provisions, nor the states of those regions or domains. This standard provides an HDL-independent way of annotating a design with power intent. In addition, the **level-shifting** and **isolation** between power domains may be described for a specific implementation, from high-level constraints to particular configurations.

When the logic in a power domain receives different power supply levels, the logic state of portions of the design may be preserved with various state-retention strategies. This standard provides mechanisms for the refined and specific description of intent, effect, and implementation of various retention strategies. Incorporating components into designs is greatly assisted by the encapsulation and specification of the characteristics of the power environment of the design and the power requirements and capabilities of the components; this information encapsulation mechanism is also described in this standard. The analysis of the various power modes of a design is enabled with a combination of the description of the power modes and the collection, generation, and propagation of switching information.

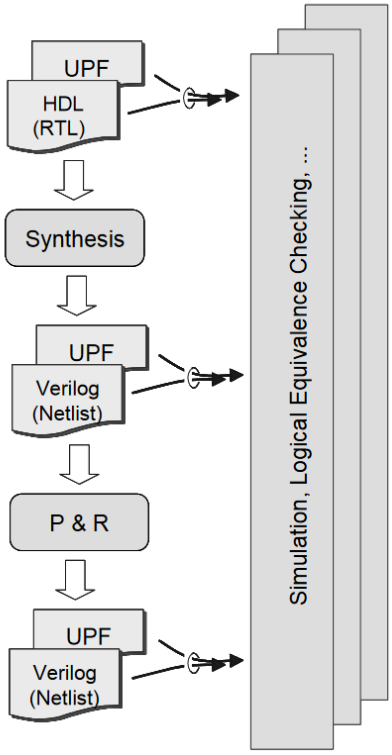


Figure 1 UPF Tool Flow [1]

Acknowledgement:

For more Information, please refer the cited document below;

[1] IEEE Standard for Design and Verification of Low Power Integrated Circuits. (n.d.). doi:10.1109/ieeestd.2009.480984

[2] https://semiengineering.com

[3] https://www.cadence.com

Definitions:

Level Shifters:

Level Shifters (LS) are special standard cells used in Multi Voltage designs to covert one voltage level to another. As Multi Voltage designs have more than one voltage domain, level shifters are used for all the signals crossing from one voltage domain to another voltage domain.

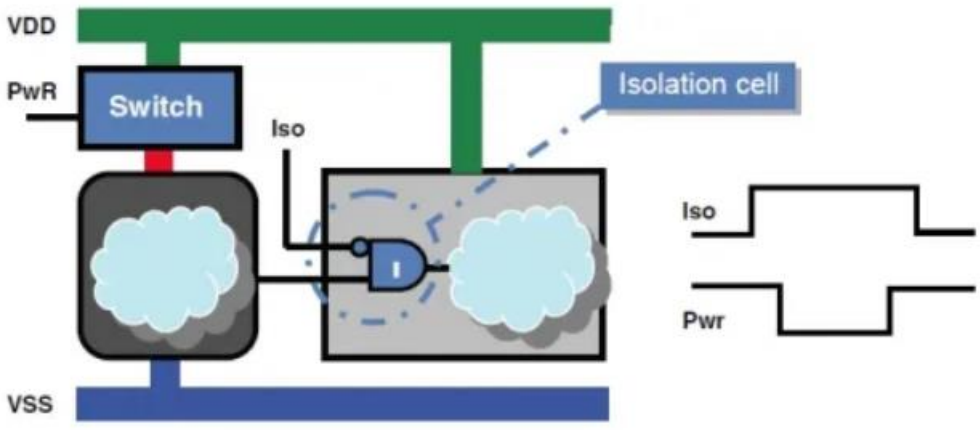
- Low to High Level Shifter
- High to Low Level Shifter
- Bidirectional Level shifters

Isolation cells (Clamp Cells):

Logic used to isolate signals between two power domains where one is switched on and one is switched off. The most common usage of such cell is to isolate signals originating in a power domain that is being switched off, from the power domain that receives these signals and that remains switched on.

Isolation logic is typically used at the output of a powered-down block to prevent floating, unpowered signals (represented by unknown or X in simulation) from propagating from powered-down blocks.

The outputs of blocks being powered down need to be isolated before power can be switched off; and they need to remain isolated until after the block has been fully powered up. Isolation cells are placed between two power domains and are typically connected from domains powered off to domains that are still powered up.



Isolation cells are also additional cells inserted by the synthesis tools for isolating the buses/wires crossing from power-gated domain of a circuit to its always-on domain. The isolation list is a list which consists of all the buses/wires that needs isolation cells.

In the isolation list we specify the clamping value of the nets as logic 0 or logic 1 and accordingly the synthesis tool will insert isolation cells.

Isolation Cell Types:

- Pull Down/ Clamp to Zero Isolation Cell
- Pull UP/ Clamp to One Isolation Cell
- Latch Type Isolation Cell

Common problems that may occur while inserting isolation cells include placing the isolation cells in the wrong power domain or hooking up the isolation power supply to the switchable power supply instead of the always-on power supply.

State Retention Power Gates (SRPGs) / Retention Cells

These cells are special flops with multiple power supply. They are typically used as a shadow register to retain its value even if the block in which its residing is shut-down.

All the paths leading to this register need to be 'always_on' and hence special care must be taken to synthesize/place/route them.

In a nut-shell, "When design blocks are switched off for sleep mode, data in all flip-flops contained within the block will be lost. If the designer desires to retain state, retention flip-flops must be used".

The retention flop has the same structure as a standard master-slave flop.

To speed power-up recovery, state retention power gating (SRPG) flops can be used. These retain their state while the power is off, provided that specific control signaling requirements are met.

State retention registers require two types of power supplies: **a switchable power supply and an always-on power supply**.

Advantages: Shutdown leakage savings, which can be independent of process variations. It allows for faster system power-on because the state is preserved in the slave latch.

Disadvantages: Increased area and die size; timing penalties - increased signal and clocking delays; increased routing resources (power routing for Vcc and a power-gating signal tree with on buffers); specialized library models for SRPG cells; additional power overhead in the active mode

Tools Used:

Innovus 19.10

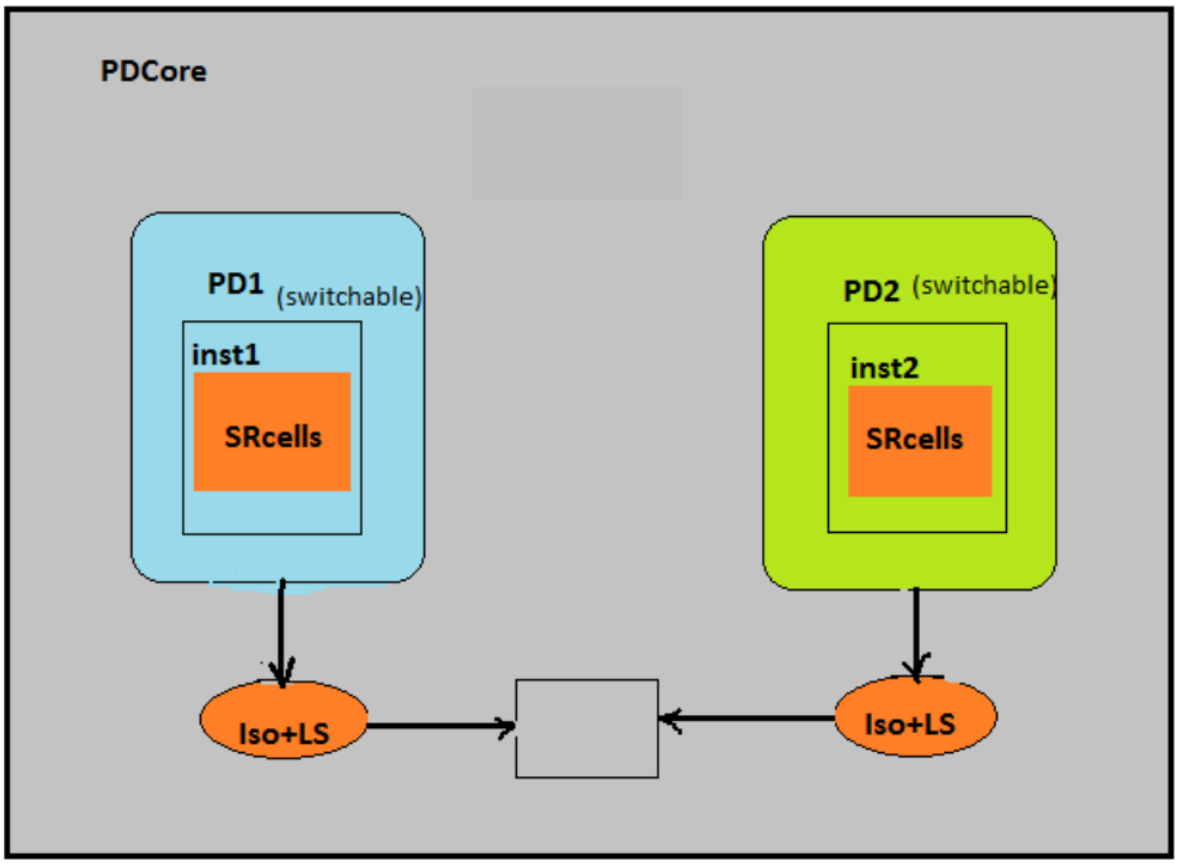
Conformal 19.10

About the Design Used here;

This lab uses a simple design (top.v); it contains two counter modules (inst1 and inst2) that are instantiated in the top module. Figure below describes the power architecture of the design.

Power Domains: The design has **three power domains**: **PDCore, PD1**, and **PD2**. PD1 and PD2 can be shut-off; therefore, **isolation** is inserted at the outputs of these power domains. PD1 has one instance of a counter and PD2 had another instance of counter.

Voltage: This is a **Multi-Voltage Design** (**PDCore operates on 1.08V; PD1, PD2 operate at 0.9V**). Therefore, in addition to **isolation cells, level-shifters** are inserted on the outputs of PD1 and PD2 domain going into PDCore.



Power Modes: The following table describes the power modes for this design.

Power Modes	PDcore	PD1	PD2
PMon	High	Low	Low
PMoff	High	Off	Off
PM1	High	Off	Low
PM2	High	Low	Off

Where, High= 1.08V, Low= 0.9V, and Off= 0V.

CONTENTS:

1. 1801-Based RTL-to-Gate Synthesis Using Genus (Legacy Mode) & Power -Aware Equivalence Checking

Synthesis of the design along with the UPF file to split the modules to operate at different power domains. [The files for doing synthesis are located in LAB4 directory]

Important Additional Commands in Genus Synthesis:

// sets the IEEE 1801 support in Genus

CMD> set enable_ieee_1801_support 1

// Read UPF File

CMD> set upf_file ../1801/top.upf

// create two power domains based on the library (Specialised Library are need for 1801 based synthesis in addition to typical, slow and fast libraries)

CMD> create_library_domain {1v08_max 0v9_max}

CMD> set_attr library {../LIBS/lib/slow_vdd1v2_extvdd1v2.lib \

../LIBS/lib/slow_vdd1v2_extvdd1v0.lib \

../LIBS/lib/slow_vdd1v2.lib} 1v08_max

CMD> set_attr library {../LIBS/lib/slow_vdd1v0_extvdd1v2.lib \

../LIBS/lib/slow_vdd1v0_extvdd1v0.lib
../LIBS/lib/slow_vdd1v0.lib} 0v9_max

// Following command generates a report that lists the number of level shifters,

isolation cells, and state retention cells available in each of the library domains

CMD> check_library
// Added to ignore level shifter for clk and reset signals (or) To not check for high-to-low voltage level shifting

CMD> set_attr clp_ignore_ls_high_to_low true

##

Read 1801

// reads in the specified 1801 file and checks it for any lint errors:

CMD> read_power_intent -1801 $upf_file -module $DESIGN

// Following command applies the power intent and associates the Power Domains in

the 1801 files with the library sets created earlier (in the same script)

CMD> apply_power_intent

CMD> set_attr library_domain 1v08_max PDcore

CMD> set_attr library_domain 0v9_max PD1

CMD> set_attr library_domain 0v9_max PD2

CMD> puts "Runtime for read_power_intent -1801 command is [expr [get_attr runtime /] - $init_time] secs"

// Following command inserts the low power cells like Isolation and level shifter cells

into the design.

CMD> commit_power_intent

After generating netlist from Synthesis using Genus, Hierarchical logical verification is performed using the Conformal LEC tool.

Note:

Hierarchical Logical Verification is the comparison of the RTL design loaded into Synthesis (Genus) against a netlist generated by Synthesis (Genus) at any stage.

2. 1801 - Based Physical Design Using Innovus:

The goal of this tutorial is to provide you a small example of 1801 implementation flow using Innovus.

The same design previously synthesized using Genus is further taken for Physical Design Implementation.

The Physical Design here involves steps like

- Design Import and Initialization
- Floor Planning
- Power Planning
- Placement
- Clock Tree Synthesis
- Routing

Understanding the IEEE 1801 coding and power intent file (Unified Power Format [UPF])

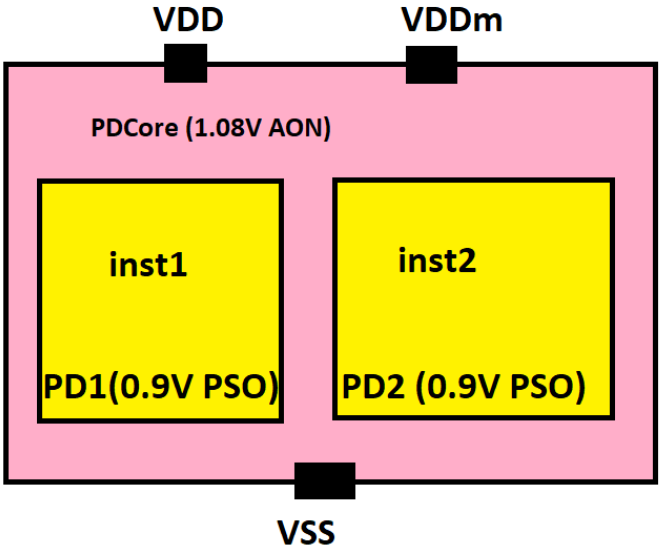
The commands to create a UPF file is explained below;

// upf version used is version 2.0

CMD> upf_version 2.0

// top module of the design is “top”

CMD> set_design_top top



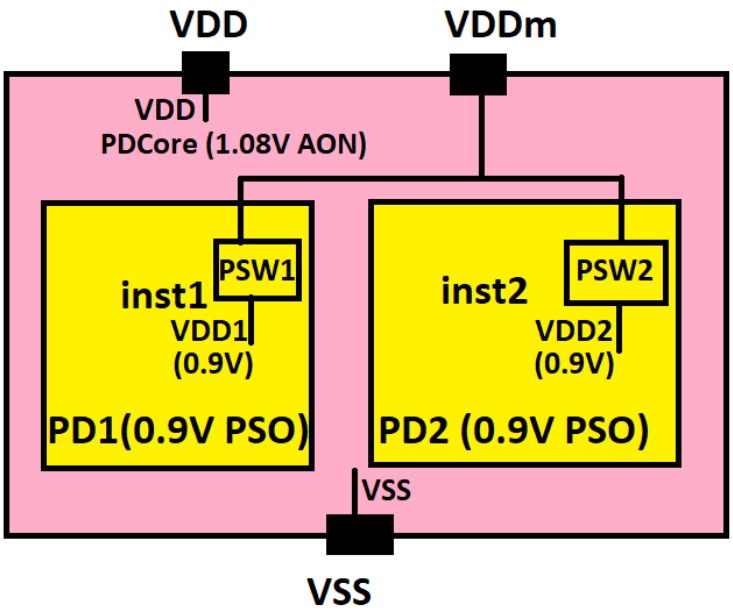
Create supply ports

// Following set of commands creates supply ports (VDD, VDDm and VSS) for the design

CMD> create_supply_port VSS

CMD> create_supply_port VDD

CMD> create_supply_port VDDm



Create supply nets

// Following set of commands creates the supply nets (VDD, VDDm and VSS).

CMD> create_supply_net VDD

CMD> create_supply_net VDDm

CMD> create_supply_net VSS

// Following set of commands connects a supply net to the specified ports

CMD> connect_supply_net VDD -ports VDD

CMD> connect_supply_net VDDm -ports VDDm

CMD> connect_supply_net VSS -ports VSS

CMD> create_supply_net VDD1 -domain PD1

CMD> create_supply_net VDD2 -domain PD2

// Following set of commands creates the supply set name within the active scope in the UPF name space

CMD> create_supply_set SS_PDcore -function {power VDD} -function {ground VSS}

CMD> create_supply_set SS_PD09 -function {power VDDm} -function {ground VSS}

CMD> create_supply_set SS_PD1 -function {power VDD1} -function {ground VSS}

CMD> create_supply_set SS_PD2 -function {power VDD2} -function {ground VSS}

// Following set of commands will create three power domains (PDcore, PD1

and PD2)

CMD> create_power_domain PDcore -include_scope

CMD> create_power_domain PD1 -elements inst1

CMD> create_power_domain PD2 -elements inst2

Note:

- This creates the power domain PDcore, and associates it with a supply set. Specify the default domain with the –include_scope option
- This creates the power domains PDmac1 and PDmac2, and associates it with a supply set.

PDCore

CMD> create_power_domain PDcore -supply {primary SS_PDcore} -update

PD1

CMD> create_power_domain PD1 -supply {primary SS_PD1} -update

PD2

CMD> create_power_domain PD2 -supply {primary SS_PD2} -update

Define an instance of a Power Switch in the scope of the -domain argument with the input/output supply ports, and the control port and on/off state.

#---

create power switches

#---

CMD> create_power_switch pwr_switch_rule1 -domain PD1 -input_supply_port {ExtVDD VDDm } -output_supply_port {VDD VDD1} -on_state {state0 ExtVDD { !PSO }} -control_port {PSO pso1}

CMD> map_power_switch pwr_switch_rule1 -domain PD1 -lib_cells HSWX1

CMD> create_power_switch pwr_switch_rule2 -domain PD2 -input_supply_port {ExtVDD VDDm } -output_supply_port {VDD VDD2} -on_state {state0 ExtVDD { !PSO }} -control_port {PSO pso2}

CMD> map_power_switch pwr_switch_rule2 -domain PD2 -lib_cells HSWX1

Power State Table (PST):

The **add_power_state** command defines a PST name and a set of supply nets to be used to create a PST.

A power state table is used for implementation—specifically for synthesis, analysis, and optimization. It defines the legal combinations of states, that is, those combinations of states that can exist at the same time during the operation of the design.

#--

Create Power State Table

#--

CMD> add_power_state SS_PDcore -state high {-supply_expr {power == `{FULL_ON, 1.08} && ground == `{FULL_ON,0.0}}}

CMD> add_power_state SS_PD09 -state low {-supply_expr {power == `{FULL_ON, 0.9} && ground == `{FULL_ON,0.0}}}

CMD> add_power_state SS_PD1 -state low {-supply_expr {power == `{FULL_ON, 0.9} && ground == `{FULL_ON,0.0}}} -state off {-supply_expr {power == `{OFF} && ground == `{FULL_ON,0.0}}}

CMD> add_power_state SS_PD2 -state low {-supply_expr {power == `{FULL_ON, 0.9} && ground == `{FULL_ON,0.0}}} -state off {-supply_expr {power == `{OFF} && ground == `{FULL_ON,0.0}}}

CMD> add_power_state PDcore -state PMon {-logic_expr {SS_PDcore == high && SS_PD09 == low && SS_PD1 == low && SS_PD2 == low}} -state PMoff {-logic_expr {SS_PDcore == high && SS_PD09 == low && SS_PD1 == off && SS_PD2 == off}} -state PM1 {-logic_expr {SS_PDcore == high && SS_PD09 == low && SS_PD1 == off && SS_PD2 == low}} -state PM2 {-logic_expr {SS_PDcore == high && SS_PD09 == low && SS_PD1 == low && SS_PD2 == off}}

Retention Registers/Cells:

Specify which objects in the domain need to be retention registers, and set the save and restore signals for the retention functionality.

#--

Set Retention Strategies

#--

CMD> set_retention SRPG_rule1 -domain PD1 -retention_supply_set SS_PD09 -save_signal {restore1 posedge } -restore_signal {restore1 negedge}

CMD> set_retention SRPG_rule2 -domain PD2 -retention_supply_set SS_PD09 -save_signal {restore2 posedge } -restore_signal {restore2 negedge}

// Specify level-shifting strategy and maps the **level-shifter cells**

set_level_shifter is used to explicitly specify a strategy for level-shifting during implementation. Level-shifters are placed on the connections between the domains operating at different supply levels.

If no level-shifter strategy applies to a connection between domains, level-shifters will be inferred based on power states.

#----------

set level shifter strategies

#----------

CMD> set_level_shifter shifter_rule1 -domain PD1 -applies_to outputs -location parent -source SS_PD1 -sink SS_PDcore -input_supply_set SS_PD1 -output_supply_set SS_PDcore

CMD> set_level_shifter shifter_rule2 -domain PD2 -applies_to outputs -location parent -source SS_PD2 -sink SS_PDcore -input_supply_set SS_PD2 -output_supply_set SS_PDcore

// Specify the **isolation strategy** and **maps isolation cells**.

#----------

Set Isolation Strategies

#----------

CMD> set_isolation iso_rule1 -domain PD1 -applies_to outputs -clamp_value 0 \

-isolation_supply_set SS_PDcore -isolation_signal {iso1_n} \

-location parent -isolation_sense low -elements {inst1/count[3]}

CMD> set_isolation iso_rule2 -domain PD2 -applies_to outputs -clamp_value 0 \

-isolation_supply_set SS_PDcore -isolation_signal {iso2_n} \

-location parent -isolation_sense low -elements {inst2/count[2]}

CMD> map_isolation_cell iso_rule1 -domain PD1 -lib_cells {LSLH_ISONL_X1_TO_ON}

CMD> map_isolation_cell iso_rule2 -domain PD2 -lib_cells {LSLH_ISONL_X1_TO_ON}

Implementation flow

Multiple Supply Voltages (MSV) and Power Shut Off (PSO):

UPF captures the following implementation techniques and will be loaded and committed before power planning in this low power design flow:

- Level shifter, isolation cell, state retention cell, power switch cell definition
- Level shifter, isolation cell, power switch cell insertion rule definition
- Power/ground net creation
- Power domain and power/ground net connection creation
- Hard macro/IP low power intent modelling

The implementation carries out **Create initial Database, Floor Planning, Power Planning, Cell Placement, Clock Tree Synthesis, Global/Detail Route, PostRoute Optimization**.

Create initial Database (Initialization / Design Planning) :

Source the global file and initialize the design as follws:

CMD> source ../INPUT/top.globals

CMD> init_design

The global file reads the netlist, LEF files and sets other additional constraints for initializing the design.

// Load and commit IEEE1801 power intent (UPF)

CMD> read_power_intent -1801 ../1801/top.upf

CMD> commit_power_intent –verbose

Use the **–verbose** option because it provides helpful information on how Innovus has interpreted the power intent

// After power intent is committed, isolation and level shifter cells are logically inserted in the netlist

CMD> verifyPowerDomain -isoNetPD

CMD> verifyPowerDomain –xNetPD

Floor Planning:

Specify the Floorplan area by specifying the Die size area, Core to Die Boundary.

Load the IO file (*.io)

CMD> floorPlan -site CoreSite -d 60 60 5 5 5 5

CMD> loadIoFile ../INPUT/top.io

Floorplan the power domains:

Besides **read_power_intent** and **commit_power_intent** commands, specify power domains for the physical-related attributes such as **minGap** and **rsExts**. These attributes are defined by the **modifyPowerDomainAttr** Innovus command.

CMD> selectObject Module inst1

CMD> setObjFPlanBox Module inst1 11.0325 11.07 22.0325 21.33CMD> modifyPowerDomainAttr PD1 -minGaps 5 5 5 5

CMD> modifyPowerDomainAttr PD1 -rsExts 5 5 5 5

MinGap [**-minGaps**] is a Halo around the domain fence and serves as a placement blockage. The row will cut in MinGap so that there is no row overlap between domains.

RouteSearchExt [**-rsExts**] is a search distance for the power router to look for a legal target to connect the power net.

Power Planning:

In the power planning, in addition to addition of rings and stripes, power switches have to be added.

Add power switches for PD1 and PD2.There are two types of power switches, the **column** and **ring switches**. Here the **column-type power switches** are used. Those switches are inserted in the switchable domains after the power domain floor plan by the **addPowerSwitch** command.

CMD> addPowerSwitch -column -powerDomain PDmac1 -enableNetOut inst1/switch_enable_out_PD1 -leftOffset 5 -bottomOffset 0 -horizontalPitch 50 -checkerBoard -loopBackAtEnd -switchModuleInstance inst1

CMD> addPowerSwitch -column -powerDomain PD2 -enableNetOut inst2/switch_enable_out_PD2 -leftOffset 5 -bottomOffset 0 -horizontalPitch 50 -checkerBoard -loopBackAtEnd -switchModuleInstance inst2

Note:

A power and ground ring are created for each power domain.

In the **addStripe** command, a combination of **-over_power_domain 1**, **-over_pin 1** and **-master HSWX1** is used to generate and connect the stripes based on the pattern of power switch cell HSWXl, in both PDmacl and PDmac2 power domains.

Add rings to PD1 and PD2.

Add stripes to Core

Add stripes (VDD1, VDDm, VSS) to PD1

Add stripes (VDD2, VDDm, VSS) to PD2

Add VDDm stripes over Power Switches.

Connect VDDm stripes together.

Special Route all the pins (Block pins & Core pins).

Verify connectivity and power domains, and **save** the design after the init stage:

CMD> verifyConnectivity -nets {VDD VDDm VDD1 VDD2 VSS} -type special -error 1000 -warning 50

CMD> verifyPowerDomain -isoNetPD

CMD> verifyPowerDomain -xNetPD

CMD> saveDesign DBS/init.enc -compress

CMD> verifyPowerDomain -bind -gconn -isoNetPD RPT/init.isonets.rpt -xNetPD RPT/init.xnets.rpt

Placement:

In the placement stage

CMD> setPlaceMode -placeIoPins false/true

// Because the IO placement file has already been provided, instruct INNOVUS not to place IO pins:

CMD> setPlaceMode -place_opt_post_place_tcl preCTS_Opt.tcl

// Add the optimization related settings in preCTS_Opt.tcl

CMD> setTrialRouteMode -maxRouteLayer Metal9

CMD> place_opt_design

// Add the tie high/low

CMD> setTieHiLoMode -cell "TIEHI TIELO" -maxfanout 10

CMD> addTieHiLo -powerDomain PDcore

CMD> addTieHiLo -powerDomain PD1

CMD> addTieHiLo -powerDomain PD2

// Perform various checks and generate reports and save the design after the placement stage:

CMD> reportIsolation -from PDmac1 -to PDcore -highlight

CMD> reportIsolation -from PDmac2 -to PDcore -highlight

CMD> checkPlace

CMD> verifyPowerDomain -isoNetPD

CMD> verifyPowerDomain -xNetPD

CMD> saveDesign DBS/place.enc -compress

CMD> verifyPowerDomain -bind -gconn -isoNetPD

CMD> RPT/place.isonets.rpt -xNetPD RPT/place.xnets.rpt

// Optimization related settings and reporting always ON buffers

CMD> setOptMode -resizeShifterAndIsoInsts true

CMD> reportAlwaysOnBuffer –all –verbose

// Check the report of always ON buffers by executing the commands

CMD> setDontUse PBUFX2 false

CMD> setDontUse PINVX1 false

CMD> set_interactive_constraint_modes [all_constraint_modes -active]

CMD> set_dont_touch [get_lib_cells PBUFX2] false

CMD> set_dont_touch [get_lib_cells PINVX1] false

CMD> set_interactive_constraint_modes { }

CMD> setDontUse CLKBUF* true

CMD> setDontUse CLKINV* true

CMD> setDontUse DLY* true

CMD> reportAlwaysOnBuffer -all

CMD> reportAlwaysOnBuffer -all –verbose

It reports how many always-on buffers optDesign can see per domain. Always-on buffering has a big impact on the optimization QoR. It is useful to check the always-on buffer availability during always-on buffering.

Clock Tree Synthesis (CTS):

Settings and Command to execute CTS clock concurrent optimization (CCOPT) stage are as follows;

CMD> setDontUse CLKBUF* false

CMD> setDontUse CLKINV* false

CMD> setDontUse PBUFX2 false

CMD> setDontUse PINVX1 false

CMD> reportAlwaysOnBuffer -all

CMD> set_ccopt_property buffer_cells {CLKBUFX4 CLKBUFX8 CLKBUFX16 PBUFX2}

CMD> set_ccopt_property inverter_cells {CLKINVX4 CLKINVX8 CLKINVX16 PINVX1}

CMD> setNanoRouteMode -quiet -routeTopRoutingLayer 9

CMD> ccopt_design

// Saving the design after the CTS stage;

CMD> saveDesign DBS/cts.enc -compress

CMD> verifyPowerDomain -bind -gconn -isoNetPD RPT/cts.isonets.rpt -xNetPD RPT/cts.xnets.rpt

Check the report of verifyPowerDomain in RPT/cts.xnets.rpt.

// Post-CTS hold optimization and saving the design after post-CTS optimization

CMD> setDontUse DLY* false

CMD> optDesign -postCTS -hold -outDir RPT -prefix

CMD> postcts_hold

CMD> saveDesign DBS/postcts_hold.enc -compress

CMD> verifyPowerDomain -bind -gconn -isoNetPD RPT/postcts_hold.isonets.rpt -xNetPD RPT/postcts_hold.xnets.rpt

Check the report of verifyPowerDomain in RPT/postcts_hold.xnets.rpt.

ROUTING:

Settings and commands for routing the design is shown below;

CMD> setPGPinUseSignalRoute PBUFX2:ExtVDD PINVX1:ExtVDD LSLH_ISONL_X1_TO_ON:ExtVDD SRDFF*:ExtVDD

CMD> setNanoRouteMode -routeStripeLayerRange 4:8

CMD> setNanoRouteMode -drouteUseMultiCutViaEffort medium

CMD> #setNanoRouteMode -drouteEndIteration 5

CMD> routePGPinUseSignalRoute

CMD> setNanoRouteMode -routeStripeLayerRange ""

CMD> verifyConnectivity -nets {VDD VDDm VDDau VDDlu VSS} -type special -error 1000 -warning 50

CMD> setNanoRouteMode -routeWithTimingDriven true

CMD> routeDesign

CMD> setExtractRCMode -engine postRoute

//Save the design after Global and Detailed routing:

CMD> saveDesign DBS/route.enc -compress

CMD> verifyPowerDomain -bind -gconn -isoNetPD RPT/route.isonets.rpt -xNetPD RPT/route.xnets.rpt

check the report of verifyPowerDomain in RPT/route.xnets.rpt.

// **Post-routing optimization** and saving the design after post-route optimization

CMD> setAnalysisMode -analysisType onChipVariation -cppr none

CMD> setDelayCalMode -siAware false

CMD> optDesign -postRoute -outDir RPT -prefix postroute

CMD> saveDesign DBS/postroute.enc -compress

CMD> verifyPowerDomain -bind -gconn -isoNetPD RPT/postroute.isonets.rpt -xNetPD RPT/postroute.xnets.rpt

Check the report of verifyPowerDomain in RPT/postroute.xnets.rpt

Verifying Connectivity:

CMD> verifyConnectivity -nets {VDD VDDm VDDau VDDlu VSS} -type special -error 1000 -warning 50