Validation of Unified Power Format and Low Power Features

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Varun K. Shah
(13MECV26)

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May 2015

Validation of Unified Power Format and Low Power Features

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Varun K. Shah
(13MECV26)

Under the guidance of

External Project Guide:

Mrs. Bharathi V
Engineering Manager,
Intel Technologies India Pvt. Ltd.,
Bangalore.

Internal Project Guide:

Prof. Vijay Savani
Associate Professor
Institute of Technology,
Nirma University, Ahmedabad.

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May 2015

Declaration

This is to certify that

1. The thesis comprises my original work towards the degree of Master of Technology in VLSI Design at Nirma University and has not been submitted elsewhere for a degree.

2. Due acknowledgment has been made in the text to all other material used.

- Varun K. Shah
13MECV26

Certificate

This is to certify that the Major Project entitled **"Validation of Unified Power Format and Low Power Features"** submitted by **Varun K. Shah (13MECV26)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.The results embodied in this major project, to the best of our knowledge,haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Vijay Savani
Internal Project Guide

Dr. N. M. Devashrayee
PG Coordinator (VLSI Design)

Dr. P. N. Tekwani
Head of EE Dept.

Dr. Ketan Kotecha
Director, IT-NU

Date:

Place : Ahmedabad

Certificate

This is to certify that the Project entitled **"Validation of Unified Power Format and Low Power Features"** submitted by **Varun K. Shah (13MECV26)**, towards the submission of the Project for requirements for the degree of Master of Technology in VLSI Design, Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.

Date: Place:Bangalore

Mrs. Bharathi V.
Engineering Manager
Intel Technologies India Pvt. Ltd.,
Bangalore

Acknowledgements

I would like to express my gratitude and sincere thanks to my thesis supervisors, **Mrs Bharathi V., Engineering Manager at Intel Technologies India Pvt. Ltd.** and **Prof. Vijay Savani, Professor, EC Department, Nirma University** for their constant guidance and motivation. I also wish to thank **Mr. Vadivel Ramalingam, Employee, Intel Technologies India Pvt. Ltd.** and all other team members at Intel for their constant help and support. Without their experience and insights, it would have been very difficult to do quality work.

I am deeply indebted to **Dr. P.N.Tekwani, Head of Electrical Engineering Department**, and **Dr. N. M. Devashrayee, Coordinator of M.Tech VLSI Design program** for allowing me to undertake this thesis work and for his guidelines during the review process.

I wish to thank my friends of my class for their delightful company which kept me in good humor throughout the journey.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

- Varun Shah
13MECV26

Abstract

Increasing functionality of battery operated devices challenge power as a key design metric along with area and timing. While dynamic power played a major role for power dissipation until now, with technology shrinking static power is equally important. Designers are using different techniques to improve power dissipation. With increasing complexity of devices, it is must essential to use EDA (Electronic Design Automation) tools.

Design houses are widely using tools from EDA vendors rather than making their own. EDA companies are challenged enough to improve on algorithms for better optimizations on power, area or timing. They are adding more and more features to make designs more better. With the advent of Unified Power Format, designers are using UPF to provide power intent separated from the design functionality in RTL.

So with this increasing complexity and functionality of different tools, it is must required to validate tools properly with different scenarios. If anything missed during validation, it will create an issue during design phase and it will impact on chip quality. To validate different features, it must require some base design on which experiments can be done. With a goal to create a reference design for low power validation, the thesis work covers aspects of creating UPFs with UPF 1.0 standard for different partitions. Different design phases like synthesis, floorplaning, placement, Clock Tree Synthesis (CTS), routing, power calculation are done on this design. This design will be used to experiment different low power experiments and EDA updates from multiple vendors. Also this design will be used to test flows which are developed on top of tools for low power activities.

It is mentioned above designers are using UPF to provide power intent file. It is IEEE standard. Designers are using UPF 1.0 standard. But latest IEEE standard for low power is UPF 2.x. It has some enhancements from UPF 1.0 standard. Major enhancement is supply sets, isolation strategy and power state table. As second step of this project, it covers conversion of UPF 1.0 to UPF 2.0 with same power intent and validate UPF 2.0 constructs. One partiton from reference design is used for UPF 2.0 validation. Different scenarios are made in UPF to validate different tools. This UPF is passed through different tools starting from low power lint checking to synthesis, P&R tool, formal equivalence tool and power calculation tool. Some of issues captured with behaviour of tools with UPF 2.0.

Contents

List of Figures

Abbreviation Notation and Nomenclature

EDA	Electronic Design Automation
UPF	Unified Power Format
RTL	Register Transfer Level
VTCMOS	Variable Threshold CMOS
GIDL	Gate Induced Drain Leakage
PG	Power and Ground
CTS	Clock Tree Synthesis
LP	Low Power

Chapter 1

Introduction

In modern era, consumers are using more and more portable devices like mobiles, tablets, laptops, palmtops, smart watches and many more. Most of these devices are battery operated. With more and more features added in these devices, power consumption is increasing and drain on the batteries. So power efficiency and energy savings become extremely important issues for designers. Designs are needed that can consume less power while maintaining comparable performance. With this complexity, designers are now dependent on EDA tools to make better designs. EDA companies are improving their tools by adding different features to implement different techniques for reduction of power. These features are first to be evaluated in the design context and then introduced in the flow.

Next thing is how to implement different low power techniques. Before going into implementation, below is a diagram which shows different type of powers. Power consumption in the

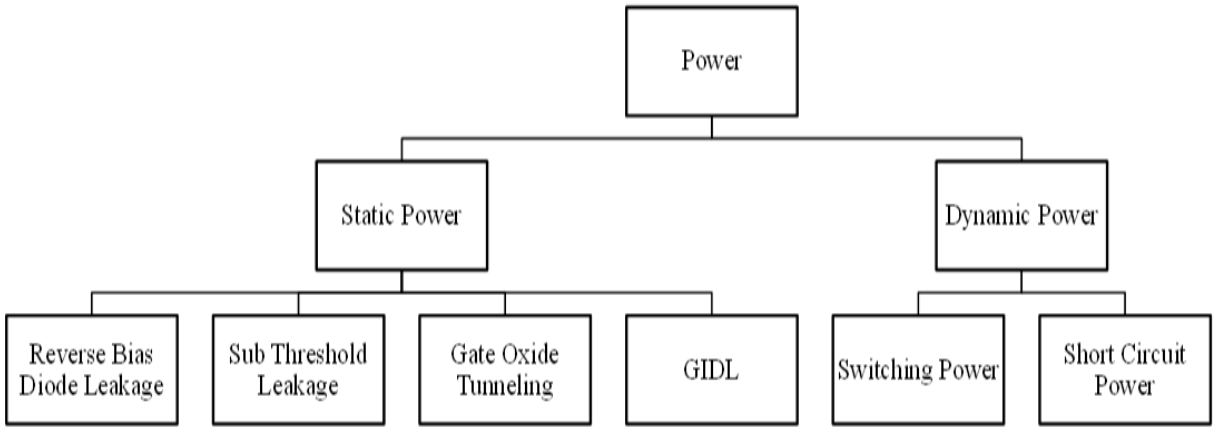


Figure 1.1: Types of Power

conventional CMOS digital circuit can be separated into three types of power dissipation (i) switching power (ii) short-circuit power and (iii) leakage power. Switching power represents the power dissipated because of the signal transitions at inputs charging or discharging the load capacitance. Short-circuit power is produced because of short circuit current which is due to the lower transition rate at input signals. This will turn on both the PMOS network and the NMOS network simultaneously in CMOS logic. It will generate current from VDD to GND to generate short circuit power. The MOSFETs in CMOS logic normally will have some non-zero reverse leakage and sub-threshold current, which causes the leakage power consumption.

Both switching power and short circuit power occur during active state of circuit so it is called as dynamic power dissipation and the leakage power occurs during idle mode of circuit so it is called static power dissipation. The static power increases faster than dynamic power with the shrinking of feature size.

There are many techniques to reduce power. Below diagram is showing different techniques which can be implemented at design level and technology level.

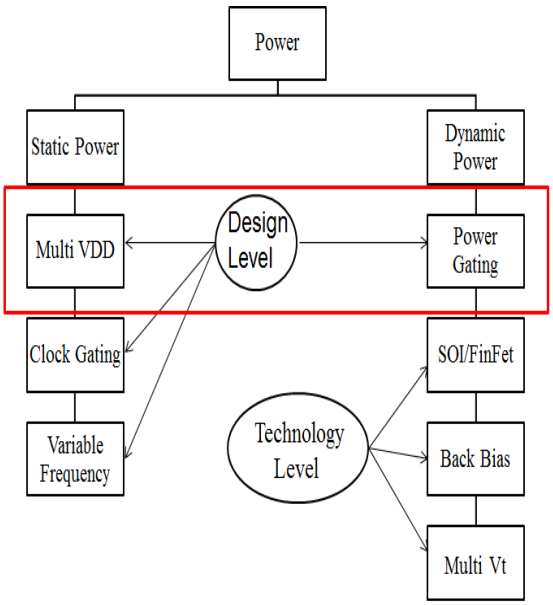


Figure 1.2: Power Reduction Techniques

Out of these different reduction techniques, focus of this thesis work is on Multi VDD Techniques and Power Gating Techniques.

1. Multi VDD:
   As name suggests, Out of complete chip, some modules can work on one voltage and another modules can work on another voltage. Also, same module can work on two different voltage.It is shown in below diagram:

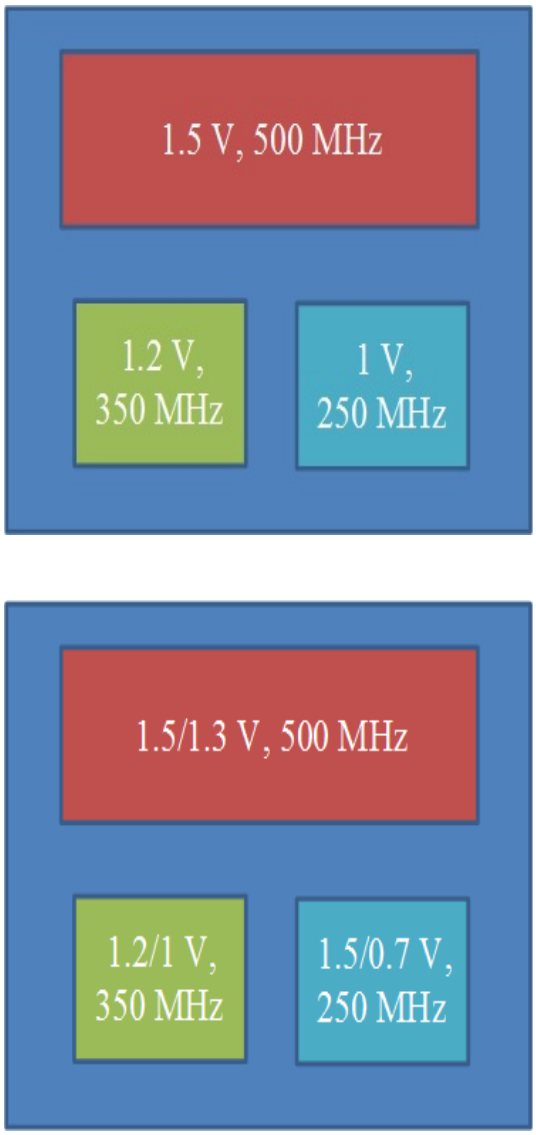


Figure 1.3: Multi VDD Technique

2. Power Gating:
   The other technique is to shut down the power supply to a block of logic when it is not active. This approach is known as power gating. As shown in diagram left side, high throughput processor which is working on 2 GHz. In this case, in any of task whole processor will work and it will dissipate more power even in case where resources are not required much more to complete task. Another approach of this which is shown in right side in figure is to make 3 cores working on 1 GHz speed so when high throughput work is requried all 3 cores will work together and for lower throughput work is requried, it can turn off cores which will reduce power.

To introduce this technique in design, Unified Power Format (UPF) is used. RTL files are used to describe functionality of design, same thing UPF file is used to describe low

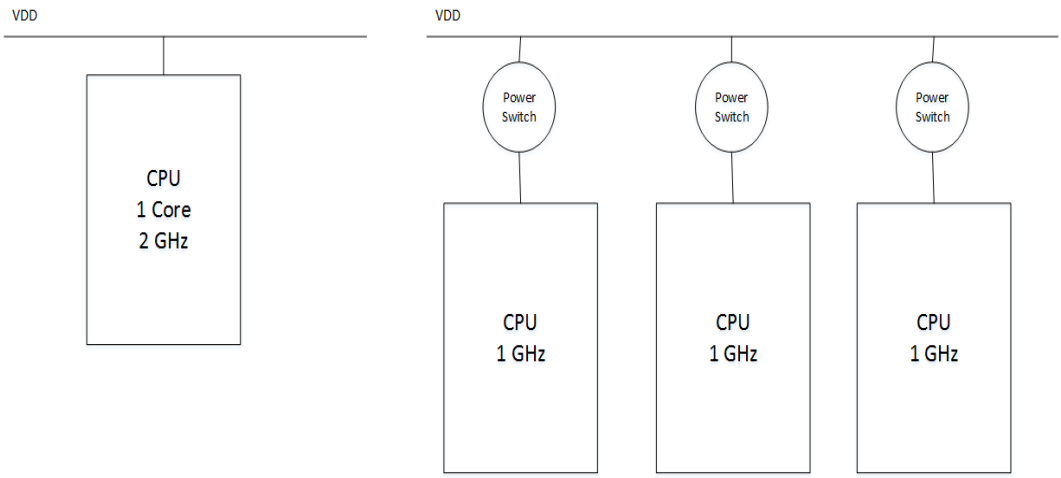


Figure 1.4: Multi VDD Technique

power specification and implement above techniques. Details about UPF is described in next chapter.

1.1 Motivation

EDA tools are coming with new features hence complexity increased.It is challenging task to validate tools for different scenarios. This project will give a good overview of low power design flow and provides an exposure to tools used in VLSI design answering some of the questions like how to resolve different low power issues,How to define UPF architecture and create UPF along with how to check different scenarios in tools.

It is also challenging to make different scenarios for validation and find proper usage of features. During validation, if something is missed, it will lead in design issue and impact on chip. So It is important that validation will cover different scenarios.

1.2 Problem Statement

There are two phases of this project.
1. In first phase, take RTL, create UPFs (UPF 1.0) and complete physical design flow through different tools. So this reference design can be used for experimenting different power related activities like to validate different new features which will be introduced by EDA companies in EDA tools in feature. Also validate extra features which will be added in flows which are made on top of these tools.

This includes all the steps that come across ASIC flow i.e. synthesis, floorplan, placement, CTS, routing, formal equivalence checking, power calculation and violation correction.

This whole implementation will be done with different EDA tools and flows developed by Intel teams on top of these tools.

2. In second phase,convert UPFs from UPF 1.0 standard to UPF 2.0 standard without changing power intent and pass this UPF to different tools and check behavior of different tools.

1.3 Thesis Organization

Thesis is organized as follows:
Chapter 2: UPF and Power Management Cells,describes about Unified Power Format and power management cells used to implement different power reduction techniques.
Chapter 3: Reference Design Creation,describes different steps of low power design, information about different tools and architectures of UPFs.
Chapter 4: UPF 2.0 Validation,describes UPF 2.0 standard and validation of UPF 2.0 standard across different tools.
Chapter 5: Conclusion

Chapter 2

UPF and Power Management Cells

Next thing is how to introduce low power intent in design? So for that UPF i.e. Unified Power Format is introduced. Unified Power Format (UPF) is an IEEE standard for specifying power intent. An UPF committee was formed by the Accellera organization, chaired by Stephen Bailey of Mentor Graphics. This version 1.0 was approved to be published on February 26, 2007. This UPF was donated to IEEE in 2006. On March 27, 2009, the "Standard for Design and Verification of Low Power Integrated Circuits" was published as IEEE Std 1801-2009. It is called as UPF 2.0. IEEE 1801-2013 was published in March 2013 and it is called as UPF 2.1.

Power gating and voltage scaling can be implemented using UPF. Mainly there are two reasons behind introducing UPF as separate standard instead of introducing as part of RTL. One is - today designers are reusing most of RTL. RTLs are updated o for adding in new functionality. Another reason is due to challenges associated to verification. If we introduce low power intent as part of RTL then for every power intent change we have to verify both functional and also low power intent. Whereas if we introduce low power intent as separate part that once functional verification is done we don't need to check functionality again. Only needs to check low power intent.

2.1 Basic Terms of UPF

Some basic terms related to UPF described below:

1. Power Domain: A collection of design elements that share a primary supply set. So generally different partitions or modules which are working on same supply can be grouped together and considered as power domain. [1]

2. Power Domain State: The state of supply net, supply port or power domain. The power state of supply net or supply port is the state and voltage values that supply net or supply port. [1]

3. Power State Table: A table that captures the legal combinations of power states for a set of supply nets. [1]

4. Supply Net: A net with power state semantics. This will provide supply to different power domain. [1]

5. Supply port: A port with power state semantics. This is used to connect power supply net with power domain. [1]

Above components are shown in below figure: Box with pink boundary is power domain. Square box at that boundary is supply port and blue and yellow lines are supply nets. There are also some special power management cells are required to implement power intent. Those are shown in figure 2.1: Declaration of all above components in UPF:

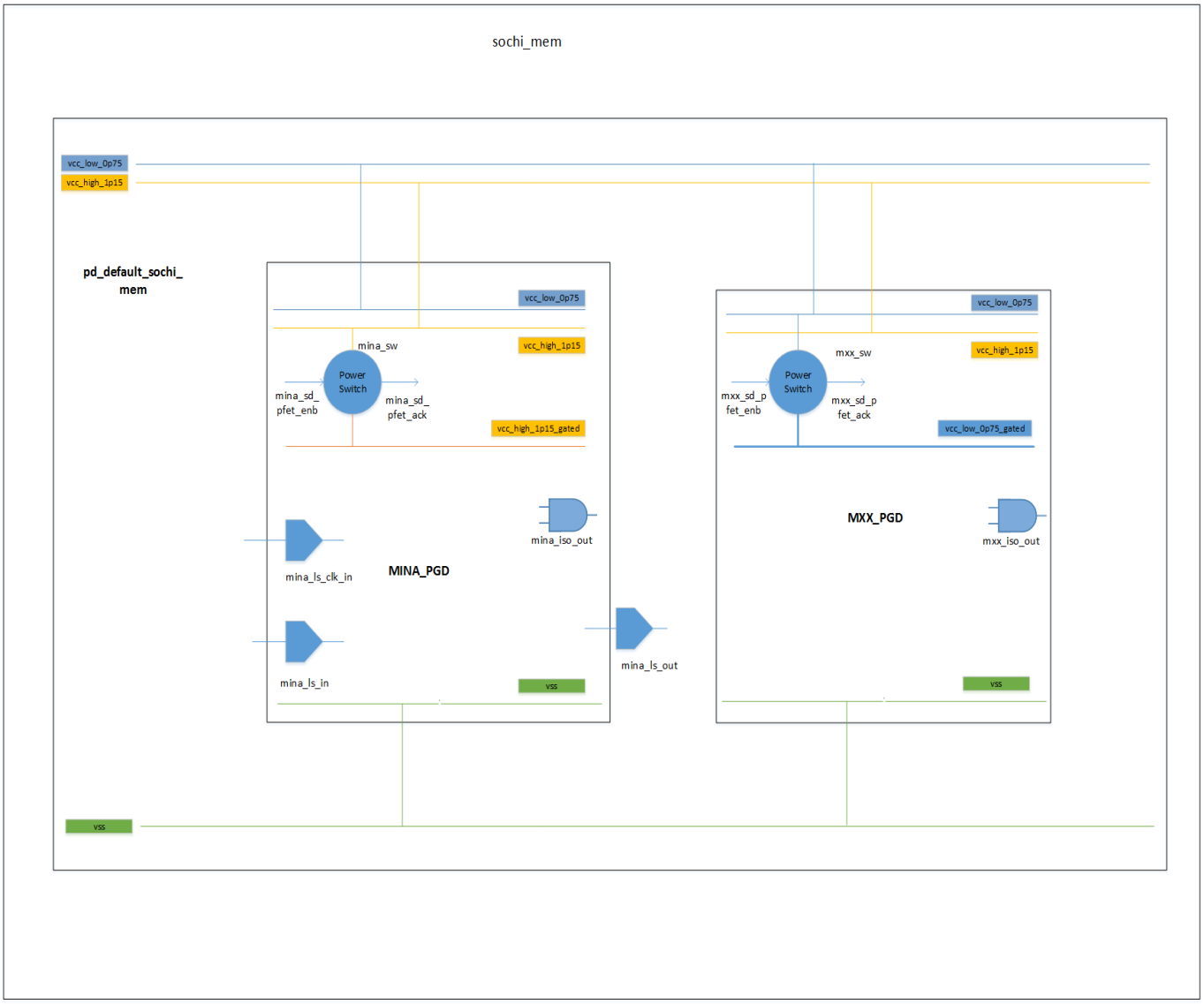


Figure 2.1: UPF Diagram

1. Power Domain:
   Syntax:
   create_power_domain domain_name
   -elements { element list }
   -exclude_elements { element list }
   -scope instance_name

   This is syntax for creating power domain which has domain name and all elements list belongs to that power domain. Also it is possible to declare all elements list which are not belong to this power domain. Here, scope is a instance name which is in logical hierarchy. [1]

2. Supply Net:
   Syntax:
   create_supply_net net_name
   -domain domain_name
   -reuse

   This is syntax for creating supply net in power domain which has net name and domain name means which domain has this net. And one more option is reuse so if supply net is coming from top level domain and it is used in below domains than it is declared as reuse. It will reuse supply net from one level above power domain. [1]

3. Supply Port:
   Syntax:
   create_supply_port port_name
   -domain domain_name
   -direction in/out/inout

   This syntax is for creating supply port through which supply net is connected to power domain or we can say that supply net will enter in power domain through this port. It has port name and domain name also direction of port is one more option. If we don't specify direction, by default it is considered as input port. [1]

4. Connection of supply port and supply net:
   Syntax:
   connect_supply_net net_name
   -ports port_list
   -domain domain_name

   This syntax is for declaring supply net and supply port connection. Here, we will

declare connection of supply net and supply port. It has net name and port list to which this supply net is connected. Also we can specify power domain name which has this supply net and supply port. [1]

5. Declaration of primary supply net:
   Syntax:
   set_domain_supply_net domain_name
   -primary_supply_net net_name
   -pimary_ground_net net_name

   This syntax is for declaring primary supply net and primary ground net for power domains. Sometimes, power domains has multiple power nets. It is required to declare primary supply net and primary ground net so all elements which are belong to this power domain, their pins have this primary supply net connection. Here, we need to define domain name, it's primary power net and ground net. Supply net and ground net must be available in that power domain. [1]

6. Declaration of supply nets voltage value:
   Syntax:
   add_port_state port_name
   -state name voltage_value
   -state name voltage_value

   This syntax is for declaring voltage value of supply nets. Here we declare supply port name which is connected to supply net. Also we declare state name and its voltage value. [1]

7. Declaration of power state table:
   Syntax:
   create_pst table_name
   -supplies supply_list

   This syntax is for creating power state table. Here we declare power state table name and all supply list which are available in different power domain. [1]

8. Declaration of power states:
   Syntax:
   add_pst_state state_name
   -pst table_name
   -state supply_states

   This syntax is for adding power states in power state table. Here we declare power

state name list, power state table name and all different combinations of supply list. [1]

2.2 Power Management Cells

1. Isolation Cells: As name suggests, isolation cell isolate output of shut down domain to always on domain. When a net is coming from one domain and that domain can be on or off. And net is going to another domain's input. So whenever first domain is turn off and second domain is turn on, net coming from first domain drives high impedance value which will create problem for on domain. Because it will get high impedance value as input and functionality of this domain will collapse. So to avoid this situation, we require isolation cell. Isolation cell provides normal value during active mode of power domain and it provides either 0 or 1 during turn off mode of power domain based on type of isolation cell. During insertion of isolation cell, care must be taken. Isolation cell can be inserted in power domain (self) or outside of power domain (parent). If always on net is available in power domain, we can insert isolation cell as self. So in this case primary supply of isolation cell is power domain's primary supply and secondary supply is always on net. If we insert as parent, it will connect to always on net of parent domain if it is available. So here we don't require secondary supply net if that parent domain is not gated.[9]
   There are two types of isolation cells:
   1. AND type isolation cell that clamp value to 0.
   2. OR type isolation cell that clamp value to 1.

   Isolation cell is show in figure 2.2.

   (a) Declaration of isolation cell:
   Syntax:
   set_isolation isolation_name
   -domain domain_name
   -elements elements_list
   -applies_to_output/input/both
   -clamp_value 0/1/Z
   -isolation_power_net net_name
   -isolation_ground_net net_name
   -no_isolation

   This syntax is for declaring isolation strategy for power domain. Here, we have to declare isolation strategy name, domain name which has this isolation strategy. Also elements list is required. If we will not declare elements list by default it

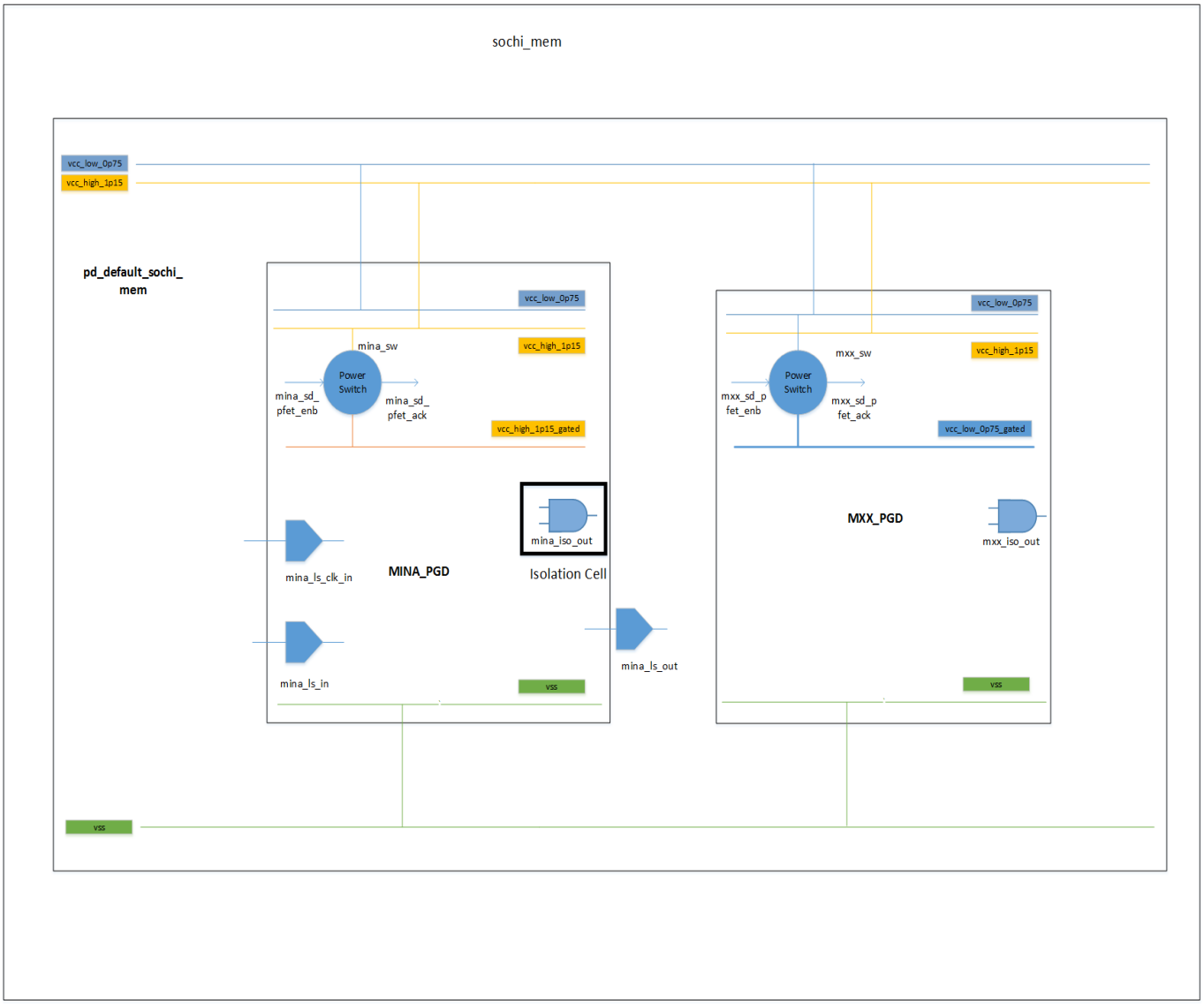


Figure 2.2: Isolation Cells in UPF

will consider all outputs or inputs. But there may be situation where we don't require isolation cells like any net which is coming from always on domain to this domain but this net is directly passing to output means it is not going to input of any cell directly connected to output port. This is called as feed through net. In this situation we don't require isolation cell on this net. So during declaration of elements, we can omit this port name. Or another way of doing this is to declare all the things and at the end declare no isolation for this element. We have to declare also that this isolation strategy is applied to inputs of all elements which are belongs to this power domain or outputs or both. Also we declare clamp value of this isolation strategy. If clamp value is one than it is OR gate isolation strategy and if clamp value is zero than it is AND gate isolation strategy. [1]

(b) Declaration of isolation control signal:
Syntax:
set_isolation_control isolation_name
-domain domain_name
-isolation_signal signal_name
-isolation_sense high/low
-location self/parent/sibling/fanout/automatic
This syntax is for declaring isolation control signal which will enable or disable isolation strategy. Here we declare isolation name for which this isolation control signal is available. Also domain name is one more required thing. Then isolation signal name, sense of isolation signal. So generally, if and gate isolation strategy is there then isolation sense is low so whenever this isolation signal is low and gate isolation is on and it will clamp value to 0. And in normal operation this signal is high. So it will pass data. Generally this signal will come from power controller or directly from input port. Another option here is location of isolation strategy. Here multiple options are available for location. [1]
Self : isolation cell is placed inside cell being isolated.
Parent: isolation cell is placed in parent of cell being isolated.
Sibling: a new sibling parallel to power this power domain is created in which isolation cells are placed
Fanout: isolation cell occurs at all fanout location of ports
Automatic: tool is free to choose location of isolation cell

(c) Declaration of isolation cell mapping to library cell:
Syntax:
map_isolation_cell isolation_name
-domain domain_name
-elements elements list
-lib_cells lib_cells_list
This syntax is for mapping isolation cell to library cell. This is not compulsory. If we will not declare this tool will automatically choose library cell for isolation cell. If we want to force tool to pick up cell from library, we need to give this map isolation cell. Here we have to give isolation cell name, domain name and library cell list. [1]

2. Level Shifters: As name suggests, level shifters are changing voltage level of signals. There may be a situation in design, where different blocks can operate at different voltage. Also in one state one power domain can operate at some voltage and in another state that power domain can operate at some another voltage. In this situation,

we require level shifters. Level shifters are inserted at two sides at inputs and also at outputs.[9]
There are 3 types of level shifters.
1. When one power domain is working at low voltage than another power domain then level shifter from low to high is required.
2. When one power domain is working at high voltage than another power domain then level shifter from high to low is required.
3. When one power domain is working at high and low voltage than another power domain in different power states then level shifter from low to high and high to low i.e. both required.
There is one more type of level shifter available. This level shifter is called as enable level shifter. So normal level shifters are able to convert only voltage level but this type of level shifter is able to isolate signal and also able to convert voltage level. Means we can say that it is combination of isolation cell and level shifter. [9]
Level shifters are shown in figure 2.3:

(a) Declaration of level shifter:
Syntax:
set_level_shifter level_shifter_name
-domain domain_name
-elements elements list
-applies_to inputs/outputs/both
-location self/parent/sibling/fanout/both
-rule low_to_high/high_to_low/both
-no_shift
This syntax is for level shifter declaration. Here we will give level shifter strategy name, domain name which has this level shifter strategy, elements list to whom this strategy is applied, also where to apply this inputs, outputs or both. Location is also required for level shifter strategy. Most important is rule. If signals are coming from high voltage module to low voltage module, we need high to low type level shifter. If signal are coming from low voltage to high voltage module, we need low to high rule for level shifter. And if signals coming from another module and that module is working on multiple supply voltage, we need both high to low or low to high i.e. both as rule. Sometimes we don't want to give level shifters for some elements than we declare them as no shift. [1]

(b) Declaration of level shifter mapping with library cell:
Syntax:
map_level_shifter_cell level_shifter_strategy

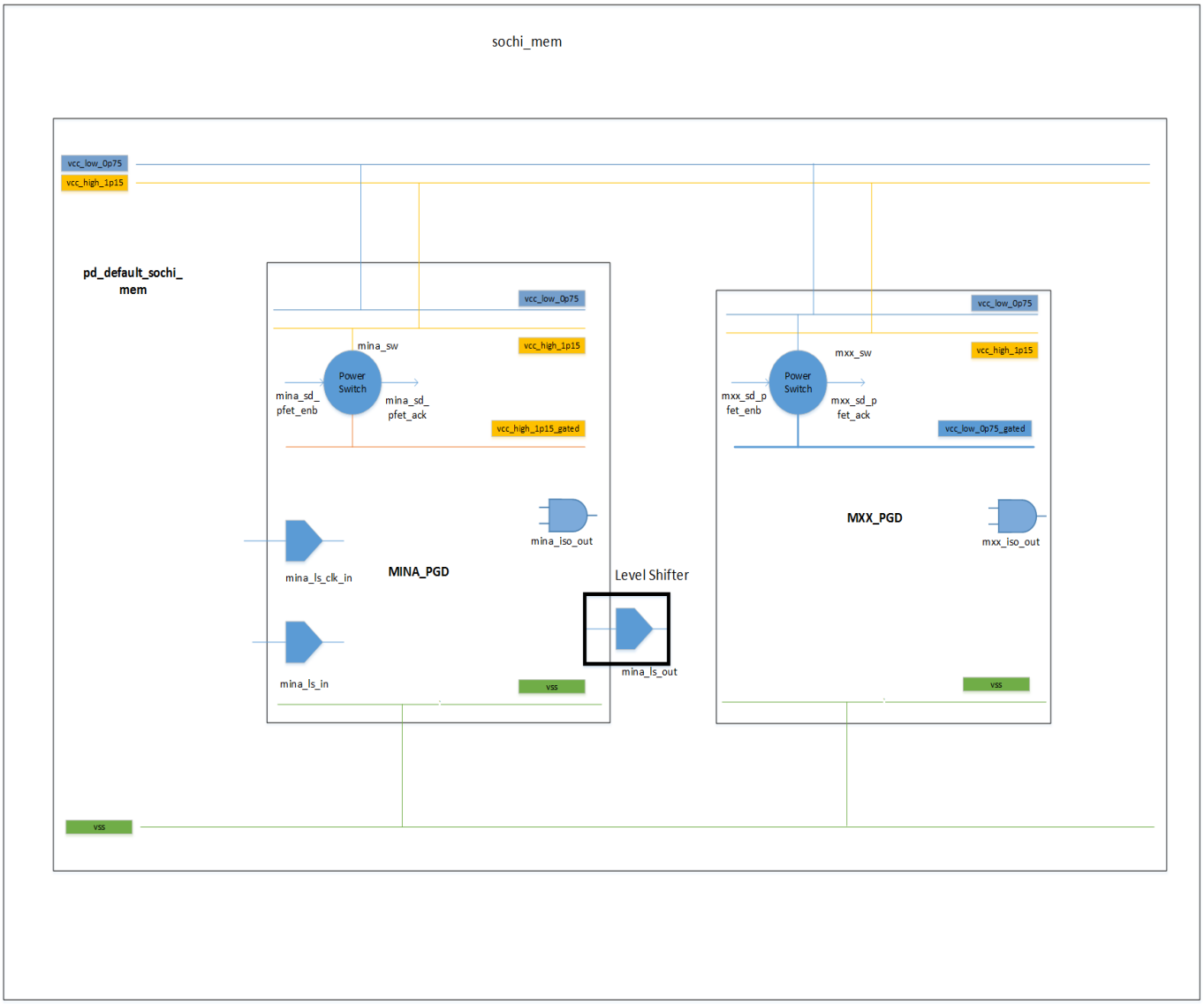


Figure 2.3: Level Shifters in UPF

-domain domain_name
-elements elements list
-lib_cells list
This syntax is for mapping level shifter to library cell. Here we need to declare level shifter strategy name, domain name, lib cells list to which this level shifter strategy is mapped. If we will not declare this tool will automatically pick up library cell from library. [1]

3. Power Switch:: As name suggests, power switch is connecting and disconnecting power supplies. Using power switch, we can turn on and turn off power domain. Normally, when switch is on, input supply net and output supply net of switch is connected. And when switch is off, input supply net and output supply net is disconnected. And

output supply net can not get voltage value. So it will turn off that power domain and all the cells belong to that power domain. We can turn off power domain in two ways. One way is disconnect vcc supply and another way is disconnect vss supply.[9] Power switch is shown in figure 2.4.

Based on this, There are two types of power switches:

1. Header switch: Header switch is used to connect or disconnect power supply net.
2. Footer switch: Footer switch is used to connect or disconnect ground net from logic.

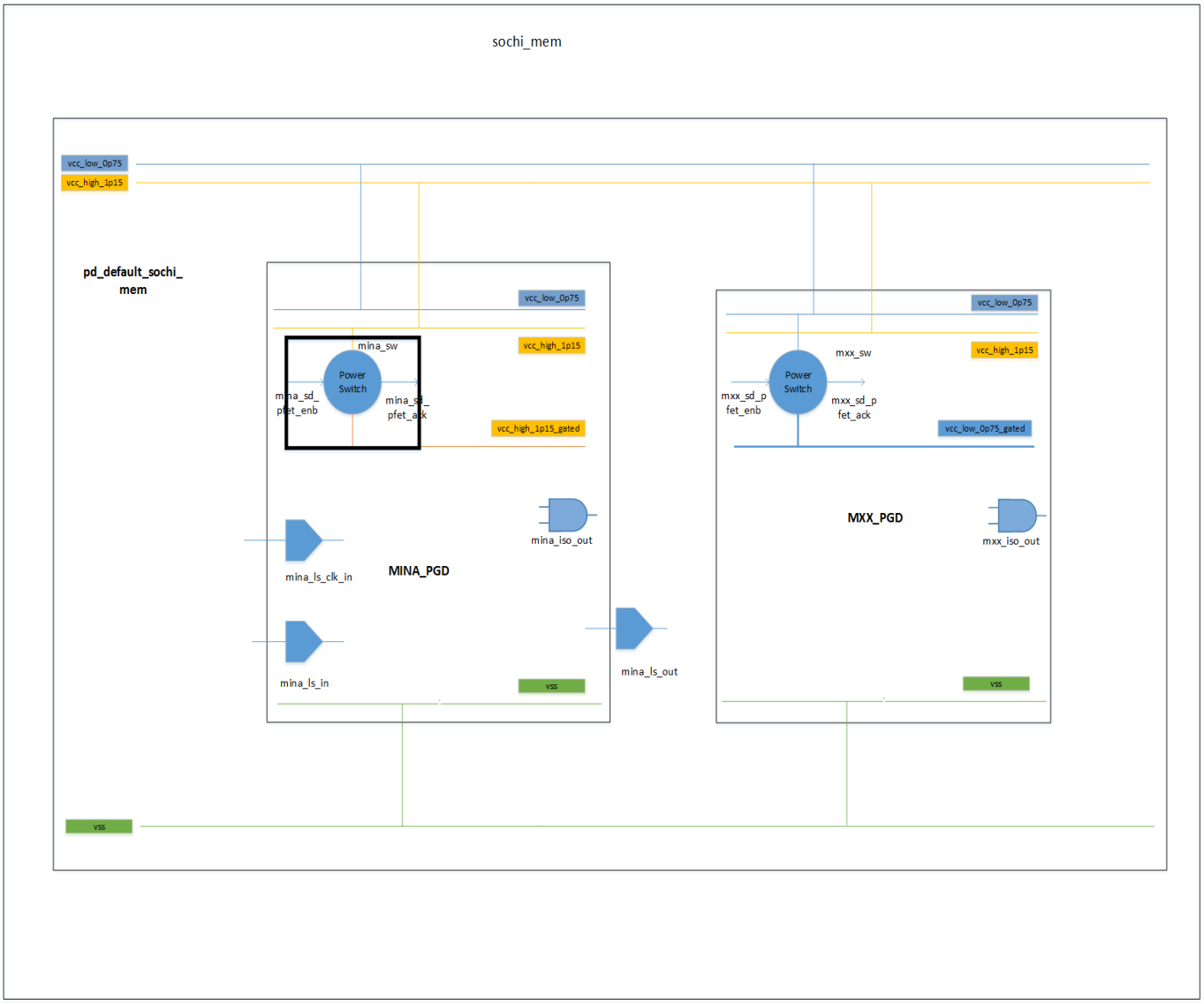


Figure 2.4: Power Switch in UPF

(a) Declaration of power switch:
Syntax:
create_power_switch switch_name
-domain domain_name

-output_supply_port port_name supply_net_name
-input_supply_port port_name supply_net_name
-control_port port_name net_name
-on_state state_name input_supply_port boolean_function
-off_state state_name Boolean_function
-ack_port port_name net_name boolean_function
This syntax is for power switch declaration. Here, we give power switch name, domain name which has this power switch. Power switch is used to connect to supply nets so we give output supply net and input supply net. Also we give control port which will turn on and turn off power switch and also acknowledgement port which will give response after tuning on and off power switch. Also we declare function to identify when to turn on power switch and turn off power switch. So when switch is on, output supply net is connected to input supply net and when switch is off output supply net is disconnected with input supply net. [1]

(b) Declaration of power switch mapping with library cell:
Syntax:
map_power_switch switch_name
-domain domain_name
-lib_cells list
This syntax is for mapping power switch to library cell. Here we need to declare power switch name, domain name, lib cells list to which this power switch is mapped. If we will not declare this tool will automatically pick up library cell from library. [1]

4. Retention Register: As name suggests, retention register is used to retain logic of output. There may be situation, when we turn off domain and again turn on domain, we need previous value to start operation. In this case, isolation cells will not work. For this, we need retention registers. So in active mode it will transfer data and when power domain is turn off, it will save last value of output. And again when power domain is turn on, it will restore that previously saved value. Retention cells are sequential cells and are of two types: a retention flip-flop and a retention latch. A retention cell is made up of normal flip flop (or a latch) for active mode working and extra save latch for turn off mode which holds the state when primary power is not available or power domain is off. A retention cell has both a primary power and a backup power supply (which is kept on). This retention register is also called as ballon type register. [9] Retention register is shown in figure 2.5 and 2.6.

(a) Declaration of retention register:

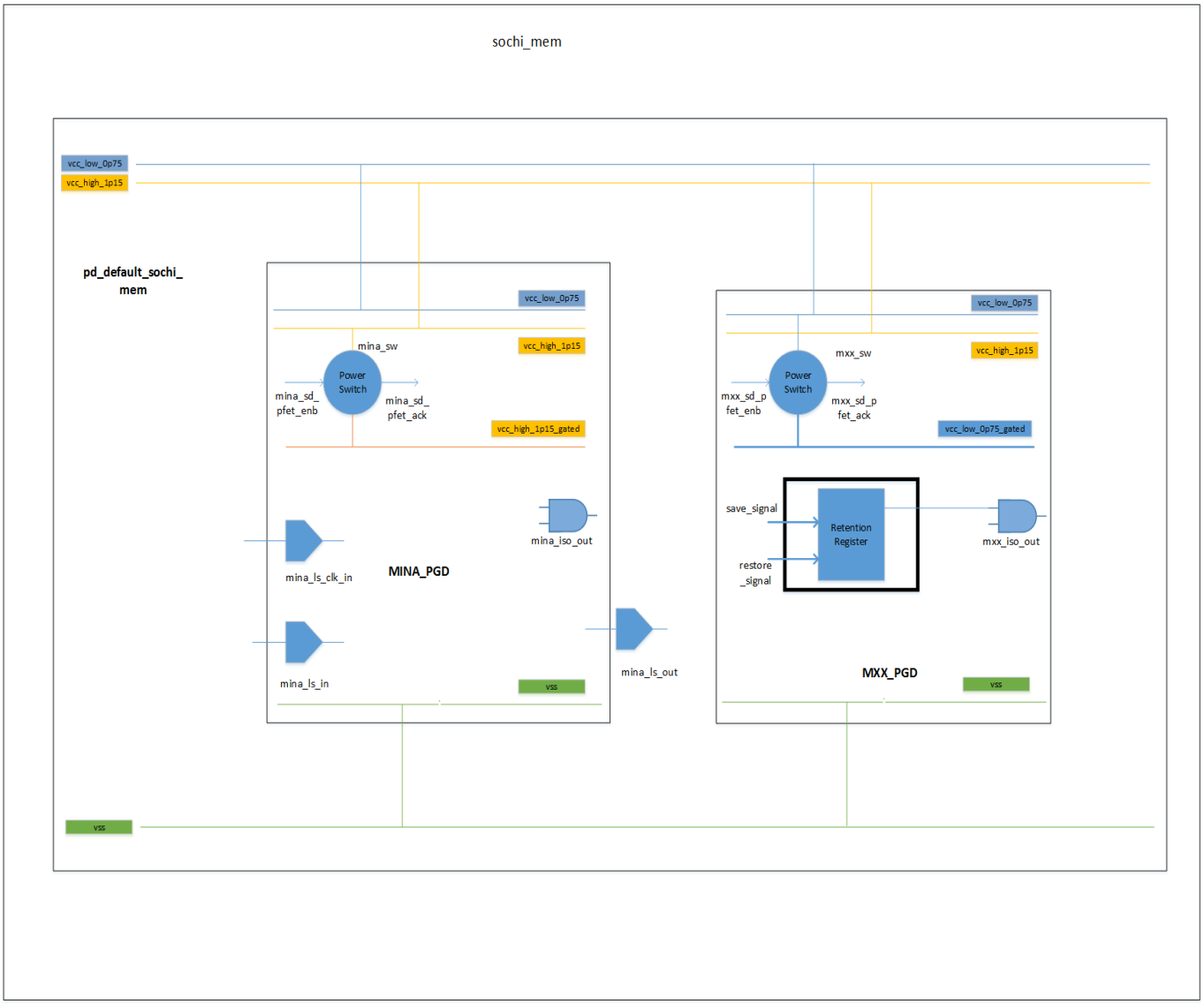


Figure 2.5: Retention Register in UPF

Syntax:
set_retention retention_name
-domain domain_name
-elements elements list
-exclude_elements elements list
-retention_power_net net_name
-retention_ground_net net_name
-save_signal logic_net high/low/posedge/negadge
-restore_signal logic_net high/low/posedge/negadge
-save_condition Boolean_function
-restore_condition Boolean_function
-retention_condition Boolean_function

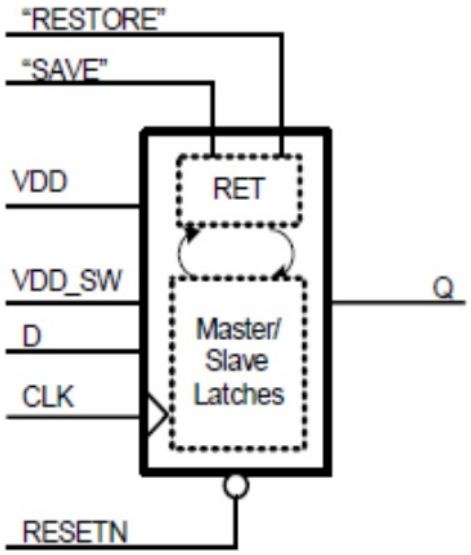


Figure 2.6: Retention Register [10]

-no_retention

This syntax is for declaring retention register strategy. Here we declare retention strategy name, domain name which has this strategy, elements list. If we want to exclude some elements, we declare these elements as exclude elements so no retention strategy on this or another option is declare all these elements and no retention on these elements. Also we declare power net and ground net connected to this retention register. Save signal and restore signal are also required to save content or restore content back and also their condition is required. This save and restore signal come from power controller or from top level ports. [1]

(b) Declaration of retention cell mapping with library cell:

Syntax:

map_retention_cell retention_name

-domain domain_name

-elements elements_list

-exclude_elements elements_list

-lib_cells list

This syntax is for mapping retention cell to library cell. Here we need to declare retention cell name, domain name, lib cells list to which this retention cell is mapped. If we will not declare this tool will automatically pick up library cell from library. [1]

2.3 UPF Example

Below is one example of UPF which is written for mobile device. As we know mobile has different modules and not all modules working in all conditions. So we can turn on and turn off modules in different modes. Mobile has many modules which work as per different conditions. But for simplicity we will consider graphics module which is used during playing of game, internet module for connecting with internet and using internet, display module which is used for displaying and Tx-Rx module for transmission and receiving messages or calls. These are different modules we will use to make UPF. UPF diagram is shown in figure 2.7.

Different modes are Normal mode, Game mode, Call mode, Flight mode, Off mode. Comlete power state table is shown below:

Mode/Units	Graphics	Display	Internet	Tx-Rx
Normal	OFF	ON(1.2)	OFF	ON(0.7)
Internet	OFF	ON(1.2)	ON(1.2)	ON(0.7)
Game	ON(1.2)	ON(1.2)	OFF	ON(0.7)
Call	OFF	OFF	OFF	ON(0.7)
Flight	ON(1.2)	ON(1.2)	OFF	OFF
Off	OFF	OFF	OFF	OFF

Architecture of UPF:
Power Domains:
Here, there are 5 power domains.

1. PD_DEFAULT: Default power domain
2. PD_GRAPHICS: Power domain for Graphics Module
3. PD_DISPLAY: Power domain for display module
4. PD_INTERNET: Power domain for internet module
5. PD_TX_RX: Power domain for transmission-reception module

Supply Nets:

1. vcc_1p2 : 1.2 volt supply net for default domain
2. vcc_1p2_gated: 1.2 volt gated supply net fot graphics, display, internet module
3. vcc_0p7 : 0.7 volt supply net for default domain

4. vcc_0p7_gated: 0.7 volt gated supply net for tx-rx module
5. vss : ground supply net

Power Switch:

1. graphics_sw : power switch for graphics module
2. display_sw : power switch for display module

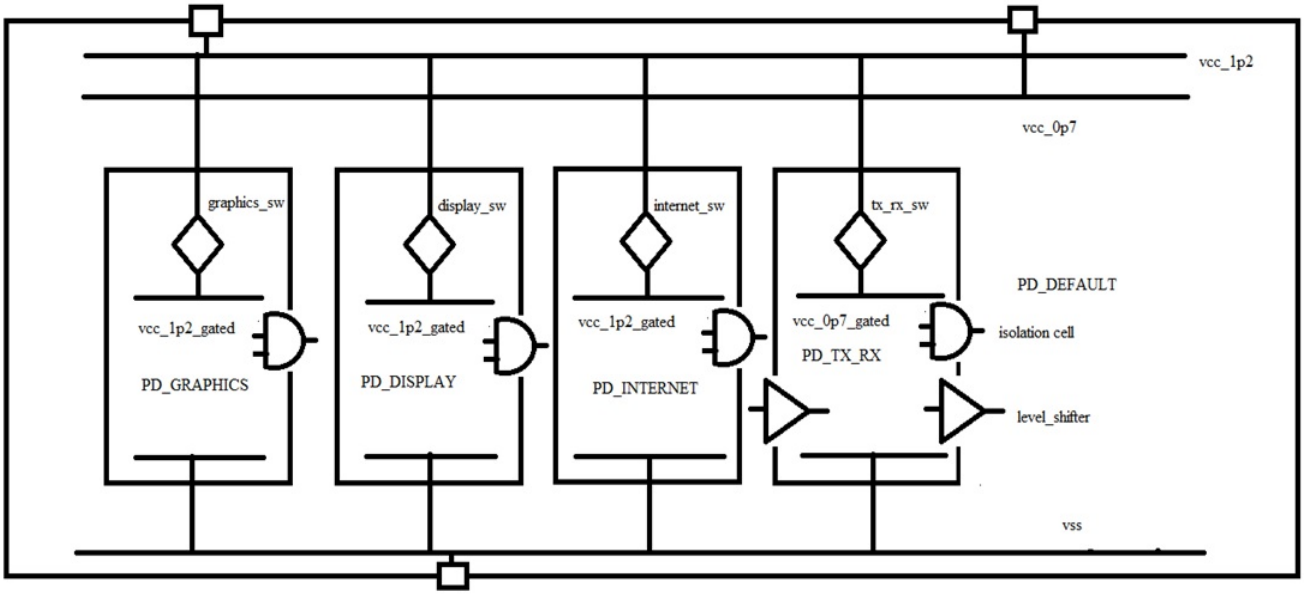


Figure 2.7: UPF diagram for Mobile

3. internet_sw : power switch for internet module
4. tx_rx_sw : power switch for tx_rx module

Isolation Strategy:

1. graphics_iso : isolation strategy for graphics module
2. display_iso : isolation strategy for display module
3. internet_iso : isolation strategy for internet module
4. tx_rx_iso : isolation strategy for tx_rx module

Level Shifter Strategy:

1. tx_rx_ls : level shifter strategy for tx_rx module

UPF:

```
#Power Domain
create_power_domaim PD_DEFAULT
create_power_domain PD_GRAPHICS –elements { graphics_module }
create_power_domain PD_INTERNET –elements { internet_module }
create_power_domain PD_DISPLAY –elements { display_module }
create_power_domain PD_TX_RX –elements { tx_rx_module }

#Supply Net and Port (1.2 v)
create_supply_net vcc_1p2 –domain PD_DEFAULT
create_supply_port vcc_1p2 –domain PD_DEFAULT
connect_supply_net vcc_1p2 –ports vcc_1p2
create_supply_net vcc_1p2 –domain PD_GRAPHICS –reuse
create_supply_net vcc_1p2 –domain PD_INTERNET -reuse
create_supply_net vcc_1p2 –domain PD_DISPLAY -reuse

#Supply Net and Port (0.7 v)
create_supply_net vcc_0p7 –domain PD_DEFAULT
create_supply_port vcc_0p7 –domain PD_DEFAULT
connect_supply_net vcc_0p7 –ports vcc_0p7
create_supply_net vcc_0p7 –domain PD_TX_RX –reuse

#supply net (1.2 gated)
create_supply_net vcc_1p2_gated –domain PD_GRAPHICS
create_supply_net vcc_1p2_gated –domain PD_INTERNET
create_supply_net vcc_1p2_gated –domain PD_DISPLAY

#supply net (0.7 gated)
create_supply_net vcc_0p7_gated –domain PD_TX_RX

#Primary net
set_domain_supply_net PD_DEFAULT –primary_power_net vcc_1p2 –primary_ground_net vss
set_domain_supply_net PD_INTERNET –primary_power_net vcc_1p2_gated –primary_ground_net
vss
set_domain_supply_net PD_GRAPHICS –primary_power_net vcc_1p2_gated –primary_ground_net
vss
set_domain_supply_net PD_DISPLAY –primary_power_net vcc_1p2_gated –primary_ground_net
vss
set_domain_supply_net PD_TX_RX –primary_power_net vcc_0p7_gated –primary_ground_net
```

vss

#Power Switch
create_power_switch graphics_sw
–domain PD_GRAPHICS
-output_supply_port {gtdout vcc_1p2_gated }
-input_supply_port {vcc_in vcc_1p2 }
-control_port {a graphics_sw_en }
-on_state { SW_ON vcc_in {!a} }
-off_state {SW_OFF {a} }
create_power_switch internet_sw
–domain PD_INTERNET
-output_supply_port {gtdout vcc_1p2_gated }
-input_supply_port {vcc_in vcc_1p2 }
-control_port {a internet_sw_en }
-on_state { SW_ON vcc_in {!a} }
-off_state {SW_OFF {a} }
create_power_switch display_sw
–domain PD_DISPLAY
-output_supply_port {gtdout vcc_1p2_gated }
-input_supply_port {vcc_in vcc_1p2 }
-control_port {a display_sw_en }
-on_state { SW_ON vcc_in {!a} }
-off_state {SW_OFF {a} }
create_power_switch tx_rx_sw
–domain PD_TX_RX
-output_supply_port {gtdout vcc_1p2_gated }
-input_supply_port {vcc_in vcc_0p7 }
-control_port {a tx_rx_sw_en }
-on_state { SW_ON vcc_in {!a} }
-off_state {SW_OFF {a} }

#Isolation Strategy
set_isolation graphics_iso
–domain PD_GRAPHICS
–applies_to outputs
–clamp_value 0
–isolation_power_net vcc_1p2
–isolation_gound_net vss

set_isolation_control graphics_iso
–domain PD_GRAPHICS
–isolation_signal iso_graphics_enb
–isolation_sense low
–location self
set_isolation internet_iso
–domain PD_INTERNET
–applies_to outputs
–clamp_value 0
–isolation_power_net vcc_1p2
–isolation_gound_net vss
set_isolation_control internet_iso
–domain PD_INTERNET
–isolation_signal iso_internet_enb
–isolation_sense low
–location self
set_isolation display_iso
–domain PD_DISPlAY
–applies_to outputs
–clamp_value 0
–isolation_power_net vcc_1p2
–isolation_gound_net vss
set_isolation_control display_iso
–domain PD_DISPLAY
–isolation_signal iso_display_enb
–isolation_sense low
–location self
set_isolation tx_rx_iso
–domain PD_TX_RX
–applies_to outputs
–clamp_value 0
–isolation_power_net vcc_1p2
–isolation_gound_net vss
set_isolation_control tx_rx_iso
–domain PD_TX_RX
–isolation_signal iso_tx_rx_enb
–isolation_sense low –location self

#Level shifter strategy

set_level_shifter tx_rx_ls_in –domain PD_TX_RX –applies_to inputs –location self –rule high_to_low
set_level_shifter tx_rx_ls_out –domain PD_TX_RX –applies_to outputs –location self –rule low_to_high

#Port States
add_port_state vcc_1p2 –state { ON 1.2 } –state { OFF off }
add_port_state vcc_0p7 –state { ON 0.7 } –state { OFF off }
add_port_state vss –state {ON 0} –state { OFF off }
add_port_state graphics_sw/gtdout –state { ON 1.2 } –state { OFF off }
add_port_state internet_sw/gtdout –state { ON 1.2 } –state { OFF off }
add_port_state display_sw/gtdout –state { ON 1.2 } –state { OFF off }
add_port_state tx_rx_sw/gtdout –state { ON 0.7 } –state { OFF off }

#Power State Table
create_pst table1 supplies {vcc_1p2 vcc_0p7 graphics_sw/gtdout display_sw/gtdout internet_sw/gtdout tx_rx_sw/gtdout vss}
add_pst_state normal –pst table –state {ON ON OFF ON OFF ON ON }
add_pst_state game –pst table –state { ON ON ON ON OFF ON ON }
add_pst_state call –pst table –state { ON ON OFF OFF OFF ON ON}
add_pst_state flight –pst table –state { ON ON ON ON OFF OFF ON }
add_pst_state off –pst table –state { OFF OFF OFF OFF OFF OFF OFF }

This chapter provided a broad overview of different types of cells required for the specific requirements of power management. Specifically, it described scenarios when different types of cells are needed. For gating off clocks, clock gating cells are required. [9]

In multi-voltage designs, level-shifters are required. [9]

For a design that has a shutdown domain with no state retention, isolation cells and power switches are required. [9]

In a design with multi-voltage domains and shutdown domains with no state retention, level shifters, isolation cells and power switches are required. [9]

In a design with multi-voltage domains including shutdown domains with state retention, level shifters, isolation cells, power switches, retention registers are required. [9]

Chapter 3

Reference Design Creation

3.1 Low Power Design Flow

First step of the flow is converting specification to Register transfer level (RTL) description which shows logic of design. With RTL files, UPF is also given which contains power intent.The RTL and UPF descriptions are contained in separate files so that they can be maintained and modified separately. To make RTL files, some standard language like VHDL, Verilog or System Verilog are used. Proper care is required during RTL Design otherwise it will lead to problems in later steps in the flow. With this RTL, UPF is also written. For this UPF IEEE standard is used. In UPF, architecture of whole system is defined like blocks which are always on, blocks which are gated, supply nets, power switches, isolation cells, level shifters, retention registers related information. Also power state tables are defined in UPF which shows different combinations of ON/OFF situations of different blocks.

After making RTL and UPF files, Low Power Rules are checked by low power checking tool. This is first step to check Low power content. Here, tool checks for isolation strategy, level shifter strategy, retention register strategy, power switches. It checks for any redundant isolation or retention register or level shifter. Also it will check, strategy is defined properly or not. It will identify if at some place any strategy is required but it is not added. Also it checks for syntax of UPF.In simple, It reads RTL and UPF file. Based on power state table defined in UPF, tool will check different low power rules. If any problems occur at this stage then we need to change UPF file. [13] Once UPF is clean, low power simulation is next step. In this, power aware simulation is carried out. Testbench is written by verification engineer to check low power functionality. Here, Verilog or System Verilog or System C is used as language. In testbench, different scenarios are created like turn off some power domains for some time and again turn on. Also, here one more check is done i.e. all power states are covered or not. If this step gives proper result, next step is synthesis.

After simulation, next step is synthesis. Process of transforming HDL designs to technology

specific gate level netlist is known as synthesis. In this, As first step, tool reads hdl files and check for syntax. Also as low power intent we have to give upf file. After that tool convert all these to technology independent cells representation. After that as second step, tool will try to optimize design at three levels area, timing and power. Here, repeated logic and unused logic will be removed. Also some extra constraints are applied by designers to achieve proper performance. As third step this optimized logic is mapped with technology cells and create netlist. Also after this, different reports are generated to check power, area and timing. In simple words, during synthesis, tool reads in the RTL files and original UPF which contains power intent information, constraints files, libraries and based on their contents, it will convert or synthesizes logical description to a gate-level netlist and an updated UPF file, which is called as UPF' (UPF prime). The UPF' file contains the original UPF information plus explicit supply net connections for special cells created during synthesis. [4]

After this, again low power rules are checked between synthesized netlist and UPF' and checks for low power intent. Same checks are done at this step like all strategies checks, redundant isolation cell or level shifter or retention register. But only thing is here, as input synthesized netlist and UPF' is given. Here one more input is required i.e. library files. Because here tool is checking low power rules for synthesized netlist. [13] After this,formal equivalence is next step. In simple words, Formal equivalence means verification between two representation. Here one input is RTL and UPF. Another input is Synthesized netlist and UPF'. It will check both these representations are equal or not. Normally in this, first all inputs and outputs are checked by name whether they are equal or not. After that tool starts to find different comparison points. Normally RTL means Register Transfer Logic means we are transferring data to one register to another register by doing some process. And as a registers, flip flops are used. Tool will identify each flop's input is one comparison point and it will break whole logic in smaller pieces and then it compares golden and revised inputs. [15] If both golden input and revised input matches then next step is to do floorplan,placement, CTS, routing.

Physical implementation is next step. Here, tool reads in the synthesized gate-level netlist and UPF' power description files, constraints, libraries and based on the file contents, performs physical implementation (floorplan to routing), It will produce another gate-level netlist, which contains all connection with power and ground information. We can called it as PG netlist.Also it will generate an updated UPF file, UPF" (UPF double-prime). The UPF" file contains the UPF' information plus any modifications to low-power circuit structures resulting from physical implementation, such as power switches. In physical implementation part first step is floorplanning. Floorplanning is process of deciding place of blocks on core area. It is an important process of creating and developing a physical model

of the design in the form of an initial optimized layout. Floorplanning directly impacts area, performance and timing. So it is more better to make floorplan good even thoug it will take some more iteration in design cycle.

Based on area of design and hierarchy, a suitable floorplan is decided upon. In floorplan, different memories, hard IPs, macros are considered. Also proper place of these is decided in floorplan. Also some blockage areas are defined in floorplan. Also I/O location are defined during floorplan. In more details, during floorplan, first step is to defined die area, core area, aspect ratio. Here core area means space where cells are going to be placed. So in floorplan, first step is to define layout. Define core area, die area, horizontal or vertical rows to place cells in these rows. Also define shape of layout either L shape layout or rectilinear layout etc. Next is partitioning. Here whole chip is divided into smaller blocks. Partitioning is mainly done due to decrease complexity, making routing easier and also tools can handle it easily. Here, two approaches are used bottom up or top down. In top down approach first whole chip layout is made and then it is divided in smaller blocks till leaf level. And in bottom up approach, it will start with small blocks and finish to full chip level. Now during defining rows multiple approaches are used. When more than three metal layers are available for routing, flip every other cell row and don't leave gap between cell rows. Another configuration is flip every other cell row and also leave gap between every other cell row. Benefit of this to get routing area. And last approach is not flip all rows and leave gap between rows. This is used when two or three metal layers are available only for routing. After that during I/O pads definition, pins are defined in that way so they can easily connect with inputs of different blocks. Also some space is left between pads. They are filled up with filler cells and corner cells. Also it needs to be considered about macro placement. Macros are placed in such a way that it will not create routing congestion, provide compact layout and not create timing issues. So floorplan is an iterative process. [14] Next step is power

planning. Power planning means connect connect all power and ground pins to power and ground rails. Normally power routes are not modified during detail routing. Main things to be considered during power planning are: [14]

1. IR drops must be in proper range
2. Electro migration requirement must meet
3. It must not lead to routing congestion
4. Layout should be compact

There are several types of power structures in power planning.Steps to make power structure contains:

1. Power rings are routed around core first
2. Power pads at core are connected to the core power rings
3. Also around macros, power rings are created if required.
4. Next step is to make vertical and horizontal stripes. It is useful to reduce the IR-drop at the power rails of the standard cells and the macros
5. Power pins of macros are connected to the surrounded power rings or at horizontal/vertical power stripes.
6. Unconnected ports of hard macros or top level ports are connected to power or ground rails.
7. Power pins of standard cells are connected to power stripes so that standard cells can tap power from power stripes.

After this step, next step is placement. In this step, all standard cells and macros are placed on their location. Normally here left to right or top to bottom approach is used. It means data will come from left side inputs and go towards right side outputs or data will come from bottom inputs and go towards top side outputs. Placement has three steps:

1. Global Placement: It will decide approximate location of partitions or different modules on core area
2. Detail Placement: It will refine location of all standard cells inside partition or module.
3. Legalization - If any overlapping is happening or routing to cells are not possible, legalization will do change in location of cells.

If placement is not done properly, it will lead to multiple issues. Major issue here is routing space. It may lead to very low space for routing.In that case, another time placement is necessary. Good routing and circuit performance heavily depend on a good placement algorithm. This is due to the fact that once the position of each block is fixed, very little can be done to improve the routing and the overall circuit performance. At the end of placement two analysis is done. One is timing analysis and another is congestion analysis. If timing is not matching, multiple approaches are used like set don't touch on different nets, high fanout net synthesis, add buffers, limit fanout or cell moving. And if congestion is more, different approaches are used to solve this like congestion options, decrease placement density, downsize cells or make placement blockage. [14]

After placement next step is clock tree synthesis (CTS). Clock is the most important in design. Because it is continuosly changing from 1 to 0 and 0 to 1. So it will dissipate dynamic power more. Also another import thing is all timing measurement are based on clock. Also, in design there may be lot of flops which has clock as input so clock has a high fan-out. Due to all these reasons, Clock is routed first in the design. After Clock tree building, all other routings are done. Major target of Clock Tree Synthesis is minimizing clock skew, minimizing power dissipation through clock, Minimizing insertion delay. Clock tree is synthesized by inserting buffers or inverters or both on clock path. Clock tree synthesis is done after placement because to propogate clock and get the exact skew between clock points, exact location of cells are must requried.If location of cells are changed after clock tree sythesis, it will change in length of routing wire and it will impact timing. Also to avoid crosstalk, clock shielding is done. [14]

There are multiple algorithms available for clock routing like H tree, X tree, zero skew algorithm. EDA tool chooses the optimized algorithm automatically and tried to make skew minimum. Also with this Static timing analysis is done in which hold and setup violations are checked. When paths are not meeting timing requirements i.e. slack is coimng as negative value, those paths must be resolved for proper functionality. One way of doing that is to increase or decrease clock period but it will impact speed of design. Another approaches, inserting buffer, up or down size cells or use timing borrow using latches. [12]

After this next step is routing. Routing is the process of creating physical connections based on logical connectivity. Signal pins are connected by routing metal interconnects. Routing is performed mainly in three major steps: [14]

1. Global routing: It will decide path to route from one pin to another ping.
2. Track Assignment - In this step, each path is given particular metal. Actual metal trace are laid down. Here, tool will try to make path straight without much zig-zag routing.
3. Detailed Routing - Detail routing will complete the routing means it will connect pin to pin and make routing.
   At the end of this step, again same DRC rule checking and timing analysis is doen. After this, completely routed design netlist is prepared. Figure 3.1 shows difference of global routing and detail routing.

After this step, same process with low power rule checking and equivalence checking. Only inputs are different. For low power rule checking, input is modified gate level netlist + UPF". For equivalence checking input is RTL+UPF and modified gate level netlist + UPF". Also,

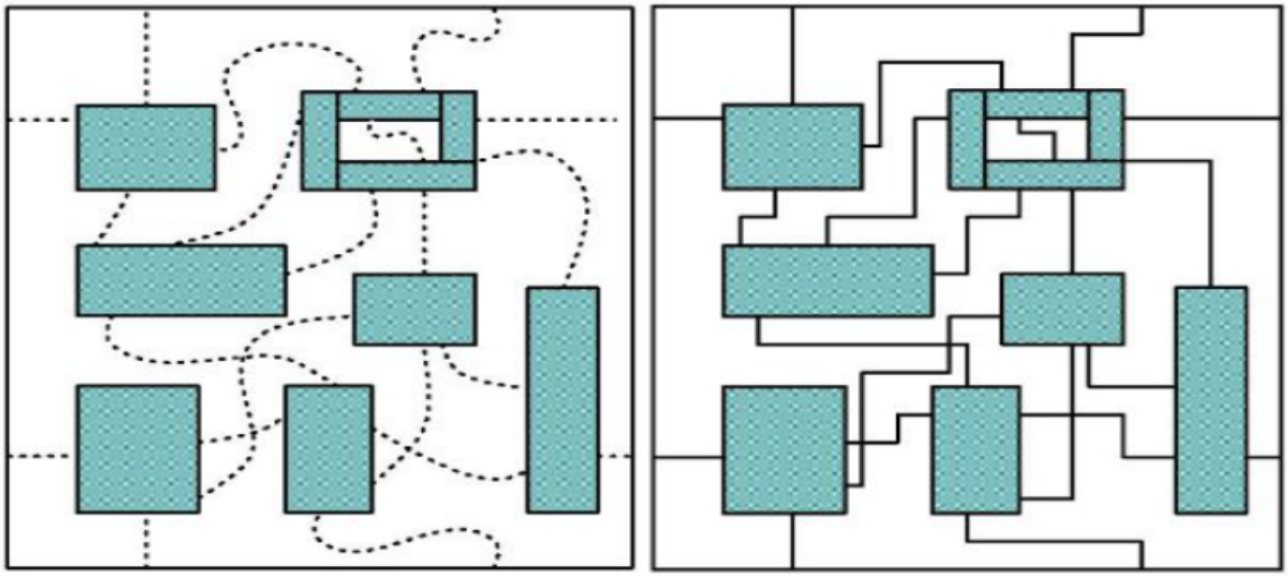


Figure 3.1: Global/Detail Routing [11]

after this equivalence checking is done between synthesized Netlist + UPF' and modified gate level Netlist + UPF".

At the end, Power calculation is done with tool. This is complete flow for low power design. Also low power design flow diagram is shown in figure 3.2.

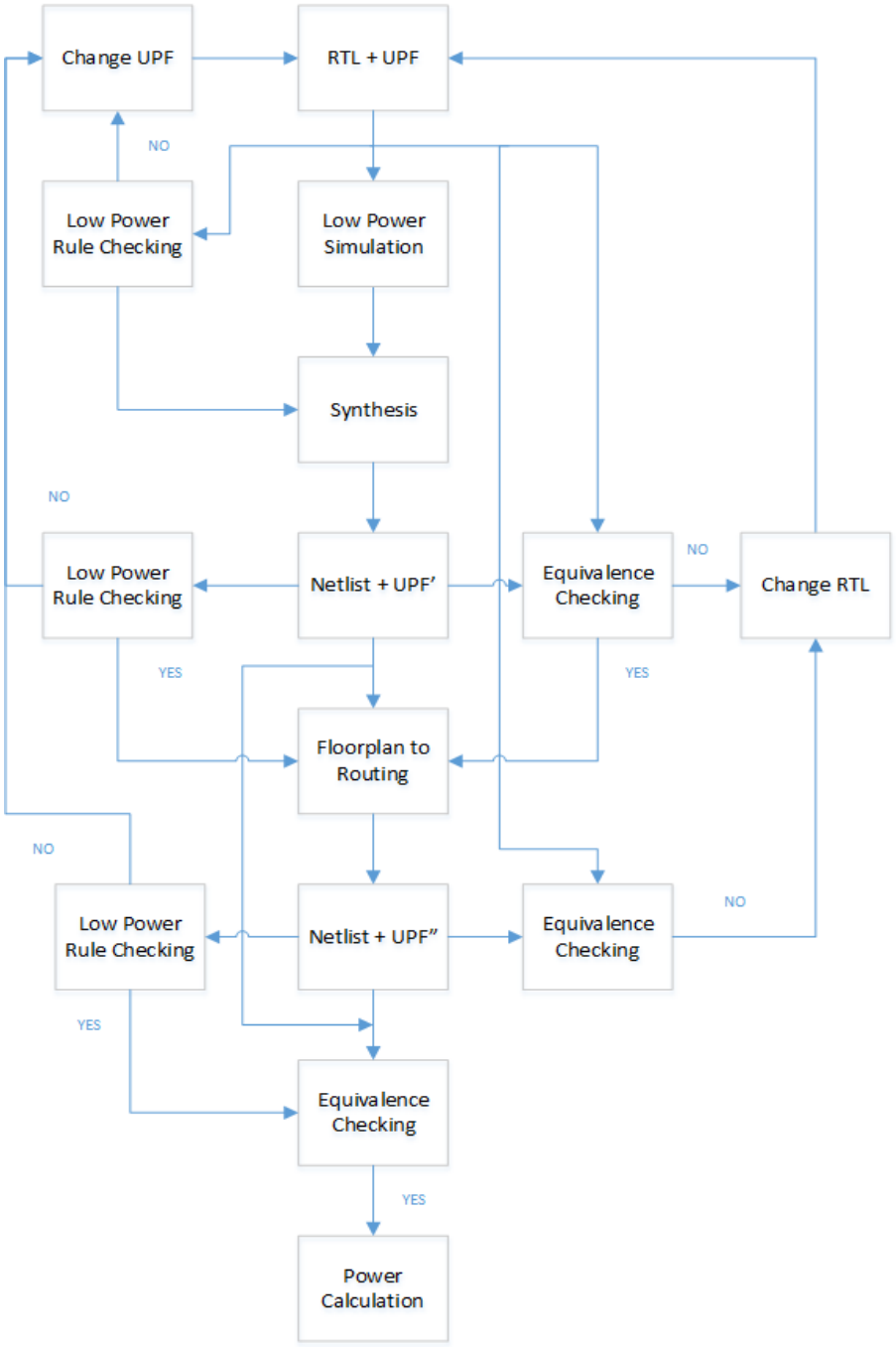


Figure 3.2: Low Power Design Flow

3.2 Architecture of UPFs for all partition

As part of reference design creation,UPFs are made for all partitions and passed these from synthesis, low power rule checking, formal equivalence checking, Place and Route and power calculation. Architectures of UPFs for all partitions are as below:

1. Partition 1 (sochi_mem) :

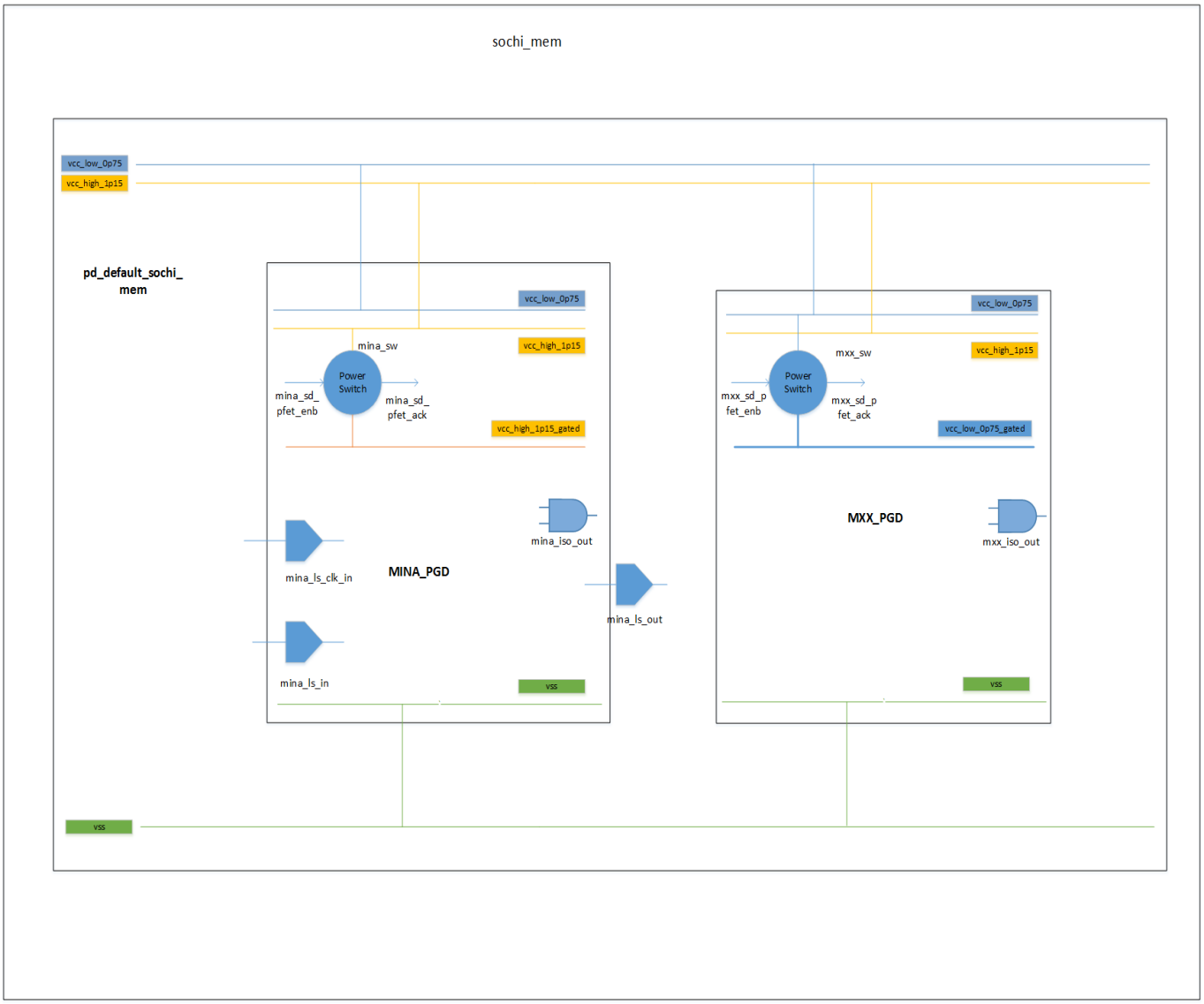


Figure 3.3: UPF diagram for sochi_mem partition

Architecture of UPF:
Power Domains:
1.pd_default_sochi_mem
2.MXX_PGD
3.MINA_PGD
Supply Nets:
1.vcc_1p15
2.vcc_0p75
3.vcc_1p15_gated
4.vcc_0p75_gated
5.vss
Power Switches:

1.mina_sw

2.mxx_sw

Isolation Strategy:

1.mina_iso: outputs

2.mxx_iso: outputs

Level Shifter Strategy:

1.mina_ls: inputs and outputs

2. Partition 2 (Sochi_cdr) :

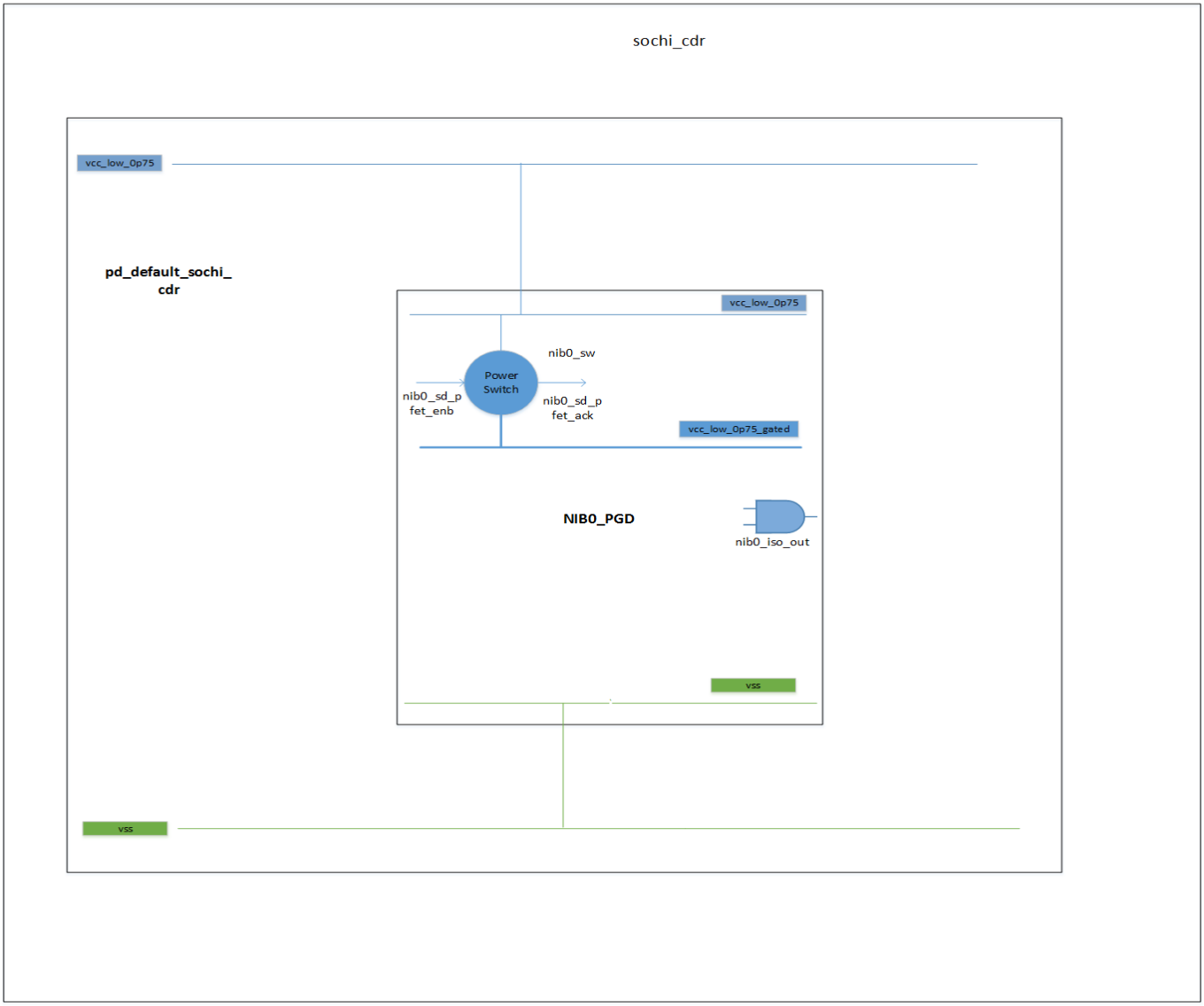


Figure 3.4: UPF diagram for sochi_cdr partition

Architecture of UPF:

Power Domains:

1.pd_default_sochi_cdr

2.NIB0_PGD
Supply Nets:
1.vcc_0p75
2.vcc_0p75_gated
3.vss
Power Switches:
1.nib0_sw
Isolation Strategy:
1.nib0_iso: outputs

3. Partition 3 (Sochi_canal) :

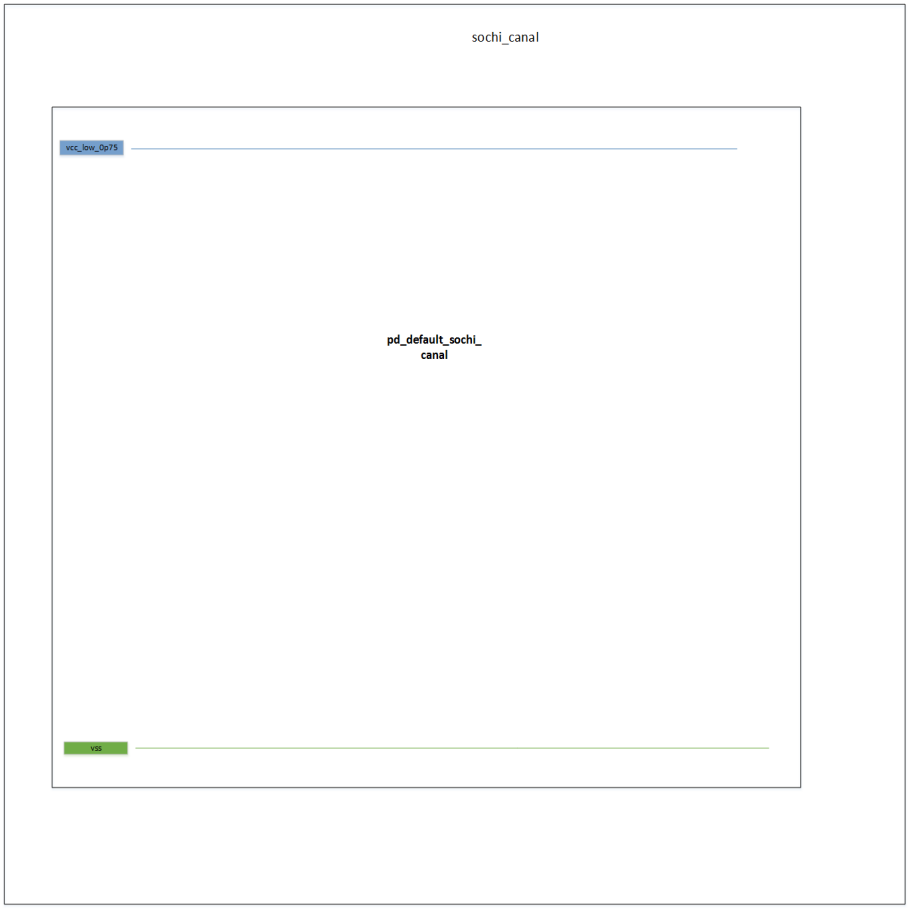


Figure 3.5: UPF diagram for sochi_canal partition

Architecture of UPF:
Power Domains:
1.pd_default_sochi_canal
Supply Nets:
1.vcc_0p75
2.vss

4. Partition 4 (Sochi_copa) :

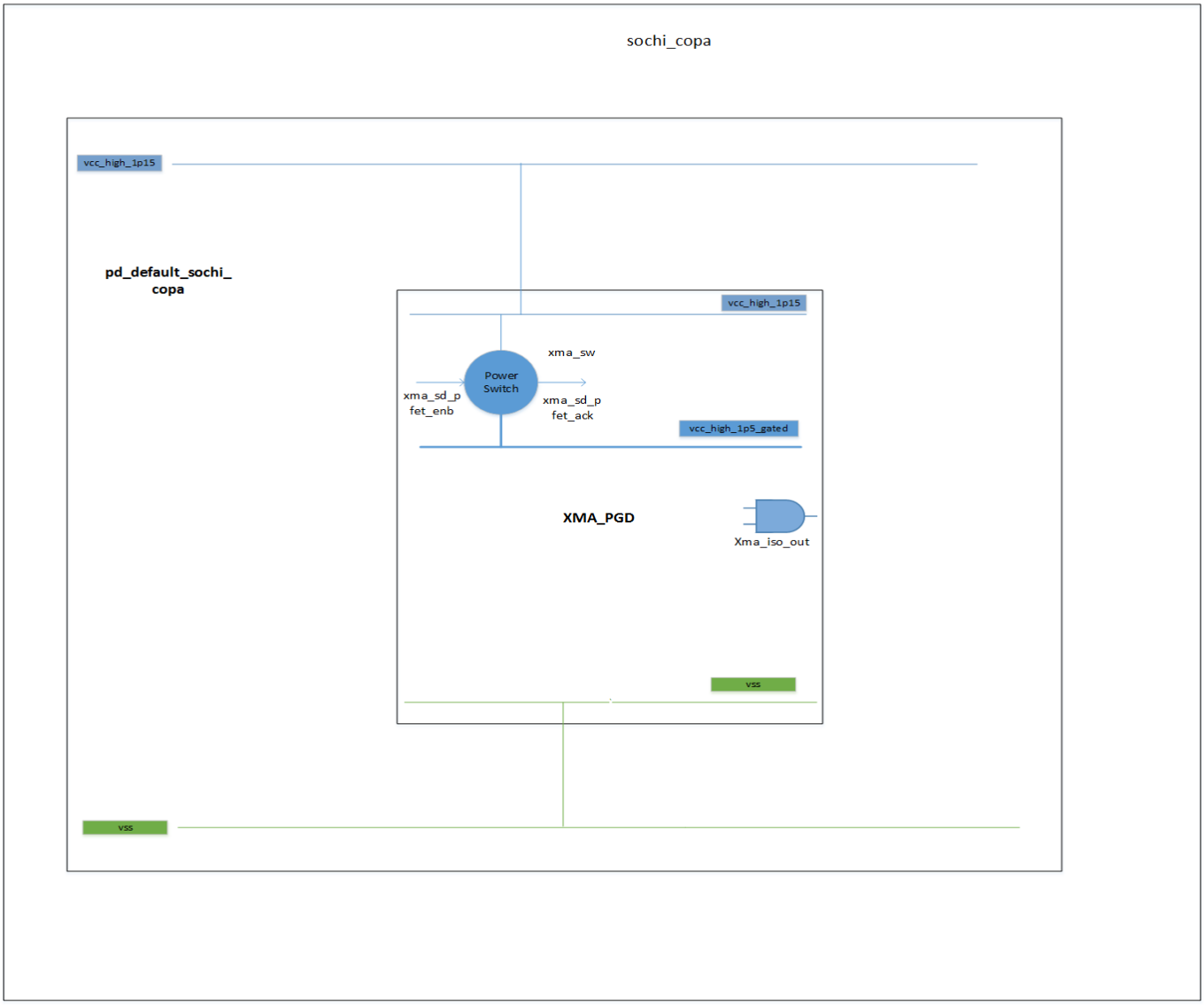


Figure 3.6: UPF diagram for sochi_copa partition

Architecture of UPF:
Power Domains:
1.pd_default_sochi_copa
2.XMA_PGD
Supply Nets:
1.vcc_0p75
2.vcc_0p75_gated
3.vss
Power Switches:
1.xma_sw

Isolation Strategy:
1.xma_iso: outputs

3.2.1 Issues Captured in Flow/Tools during Reference Design Creation

Issues captured during Low Power Lint Checking:
1. Run Type: P&R Netlist + UPF"
Error: Delay Buffer/Always On Buffer is not available in switch enable signal path. As this Low Power rule, it will check ther must be always on buffer or delay cell in path of switch enable signal or switch acknowledgement signal. So that when all switches are on, it will not take current suddenly from power source. Though Always on buffer is available in path, still tool is showing that buffer is not available. As shown in figure, power switch cell input a is connected with always on buffer output still tool is showing error message.

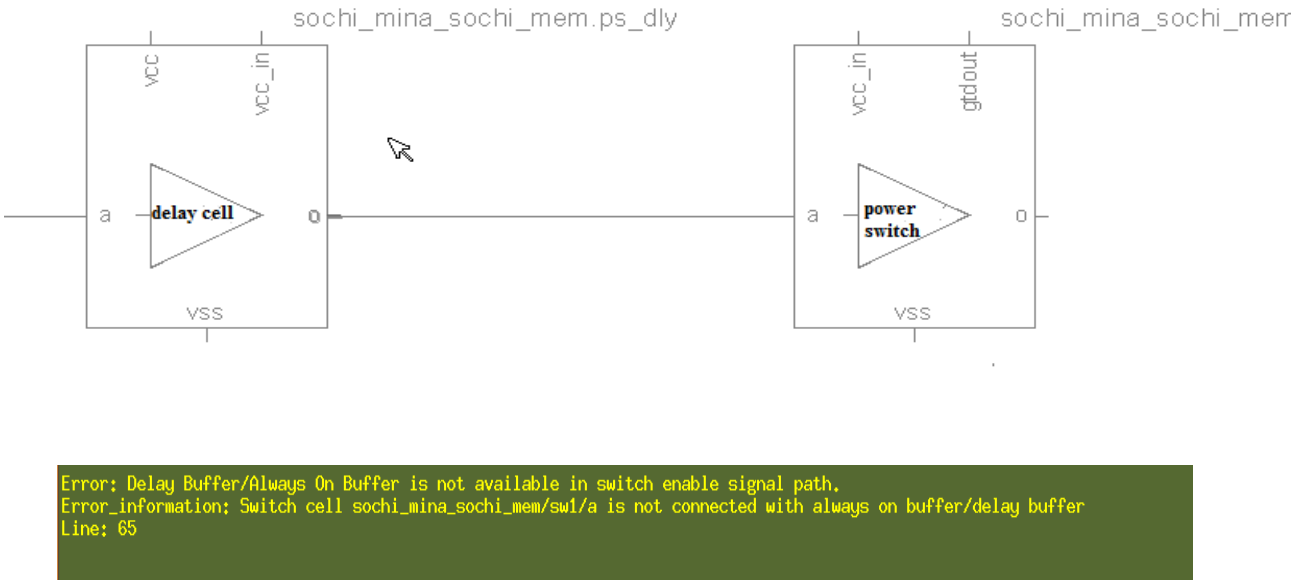


Figure 3.7: Low Power Lint Checking issue 1

Issues captured during Place and Route:
1. During P&R, tool will insert some buffers on path of isolation signal or on path of switch enable signals. They are connected with always on rails. So they are always on buffers. But if isolation strategy is defined as applies to inputs, tool will insert isolation cells for those ports which is not required. So for this tool will add no_isolation strategy in UPF. Now, each strategy should have different name. But tool is giving same name for each isolation

strategy. As it is shown in fiugre, same name no_iso_1 and no_iso_2 are used two times for different elements.

```
set_isolation no_iso_1 -domain MXX_PGD -elements sochi_mxx_sochi_mem/mxx_sd_pfet_enb -no_isolation
set_isolation no_iso_2 -domain MINA_PGD -elements sochi_mina_sochi_mem/mina_sd_pfet_enb -no_isolation
set_isolation no_iso_1 -domain MXX_PGD -elements sochi_mxx_sochi_mem/iso -no_isolation
set_isolation no_iso_2 -domain MINA_PGD -elements sochi_mina_sochi_mem/iso -no_isolation
```

Figure 3.8: Place and Route Tool issue 1

Chapter 4

UPF 2.0 Validation

On March 27, 2009, the "Standard for Design & Verification of Low Power Integrated Circuits" was published as IEEE Std 1801-2009. It is called as UPF 2.0. This UPF standard comes with many new enhancements from Previous UPF 1.0 Standard. Below are those enhancememnts.

1. Supply Sets: Major concept in UPF 2.0 is supply sets.
   Supply set is nothing but collection of different power rails which makes power source. Previously, In UPF 1.0 standard supply nets and supply ports are domain dependents. For each domain we need to declare supply rails and supply ports. Here, concept of supply sets is introduced. In this you have to define all supply rails at top level default power domain. Now you can make supply sets which will associate to these supply rails and then these supply sets are used in any power domain.[17]
   Syntax:
   create_supply_set supply_set_name
   -function function_name net_name
   e.g.
   create_supply_set SS1
   create_supply_set SS2

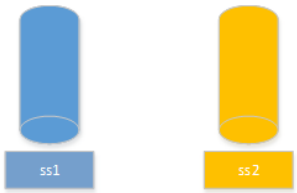


Figure 4.1: Supply Set

In above figure, Supply Set SS1 and SS2 are shown.
Connection of Supply set with supply rails:

By default, each supply set has six different functions:
power,ground,pwell,nwell,deeppwell,deepnwell.
We can connect supply rail with supply set using this function.
Example:
create_supply_set ss1 -function {power vcc1} -function {ground vss} -update
create_supply_set ss2 -function {power vcc2} -function {ground vss} -update
Here, in last –update switch is used because supply sets are already declared and we are just updating these supply sets and connecting supply rails with these supply sets.
If we want to declare supply set and connect supply set with supply rails at same time we can do it by
create_supply_set ss1 –function {power vcc1} –function {ground vss}
Below figure shows how supply sets are connected with supply rails:

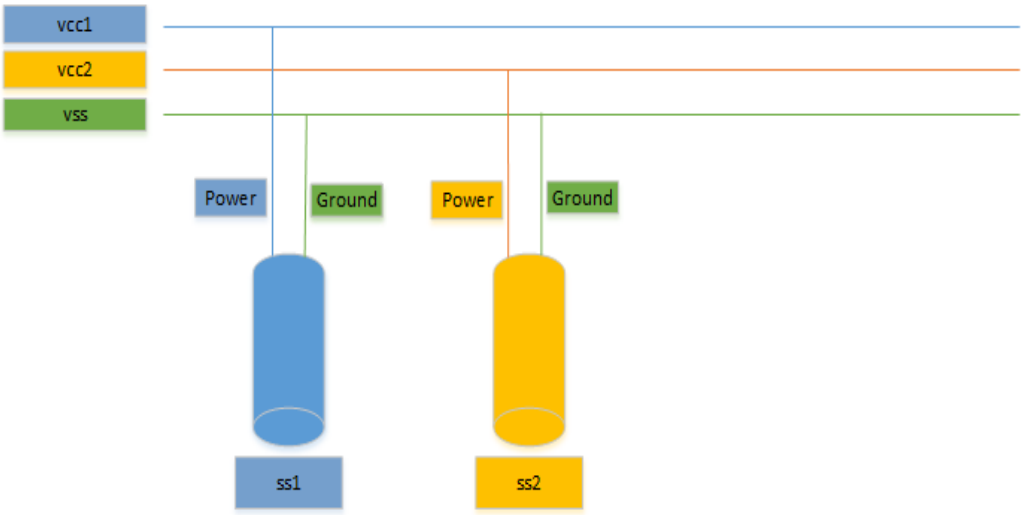


Figure 4.2: Supply Set Association with Supply Rail

2. Supply Set Handle:
   Supply Set Handles are defined for power domain through which we can associate supply sets with power domain. There are two types of supply set handles:

   (a) Predefined Supply Set handles : When we are defining power domain, supply sets handle are defined by default which we can also called as implicit supply set handles.
   e.g. create_power_domain PD1
   When we are defining power domain PD1, multiple predefined supply sets handled are defined. e.g. PD1.primary, PD1.default_retention, PD1.default_isolation etc..

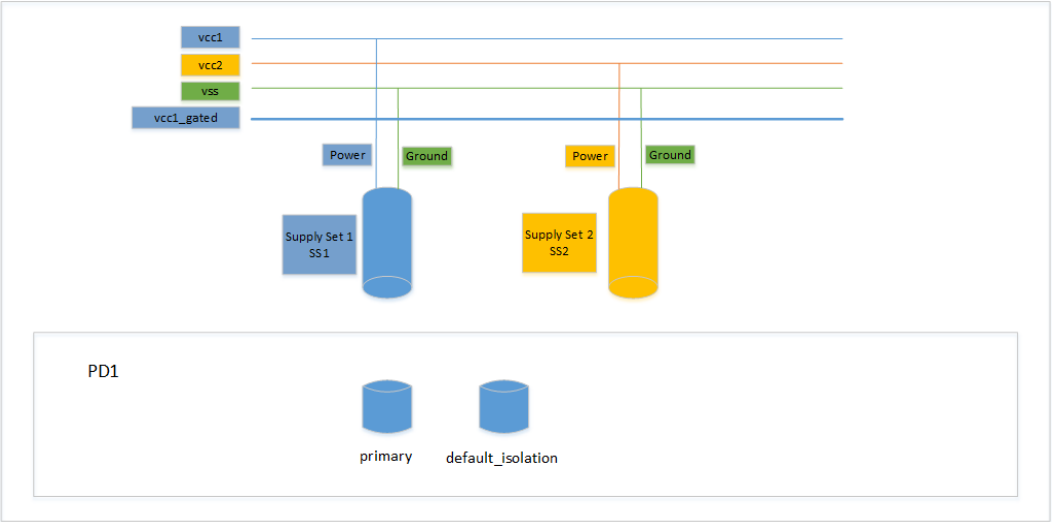


Figure 4.3: Implicit Supply Set Handle

There are two ways to associate implicit supply set handle with supply set:

i. During Create Power domain:
   Syntax:
   create_power_domain domain_name
   –supply primary supply_set_name
   –supply default_isolation supply_set_name
   –supply default_retentions supply_set_name
   –supply extra_supplies_0 supply_set_name
   e.g.
   create_power_domain PD1 –supply {primary ss1} –supply {default_isolation ss1}

ii. Using associate supply set: (This command is only used for implicit supply handle association)
   Syntax:
   associate_supply_set supply_set_name
   –handle power_domain.primary
   e.g.
   associate_supply_set ss1 –handle PD1.primary
   associate_supply_set ss2 –handle PD1.default_isolation

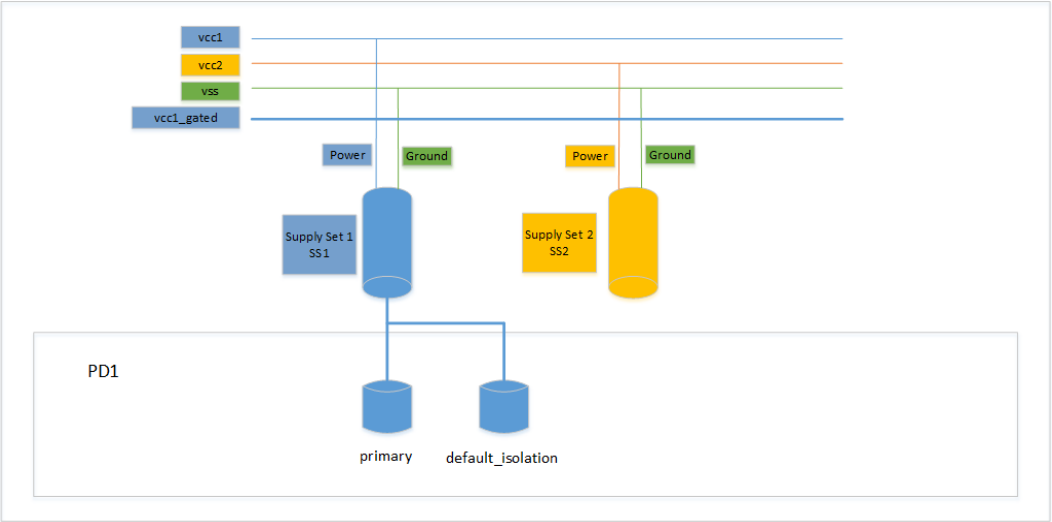


Figure 4.4: Implicit Supply Set Handle

(b) User Defined Supply Set handles :

What if we need more than one supply set in a power domain? Which supply set handle should we use?

For this user defined supply set handle is used which is also called as explicit supply set handle.

During create_power_domain:

create_power_domain PD1 –supply primary ss1 –supply default_isolation ss1 –supply extra_supplies_0 ss2

If we are using only explicit supply set handles than How to declare primary supply net for domain because we are not using power_domain.primary?

Here, we can use UPF 1.0 construct:

set_domain_supply_net PD1 –primary_power_net ss1.power –primary_ground_net ss1.ground

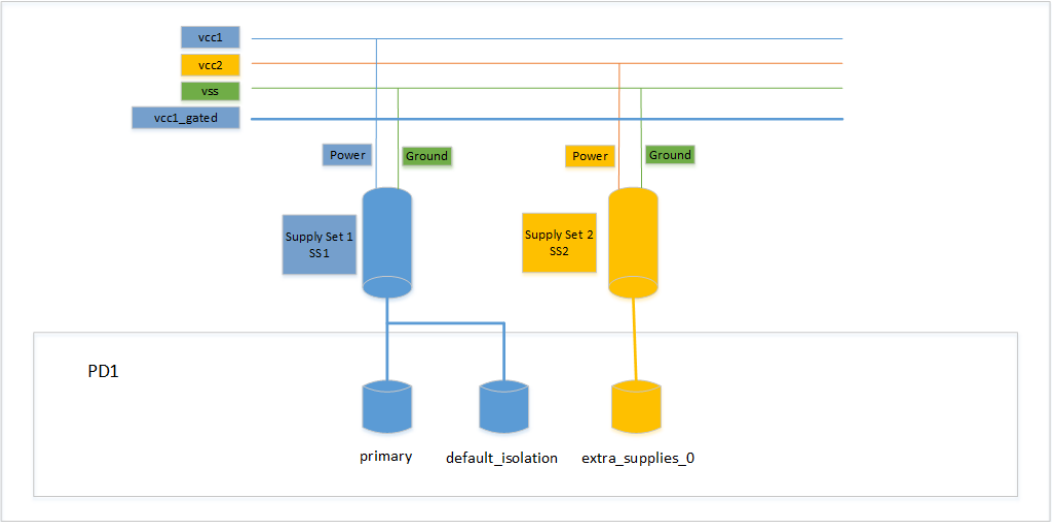


Figure 4.5: Explicit Supply Set Handle

3. Isolation Strategy Declaration: Here, there are two concepts introduced to declare isolation strategy.

   (a) diff_supply: Whenever, outputs of one power domain is going to another power domain and if both power domain's supply sets have different name, diff_supply to true will insert isolation cells on those ports. Here, only supply set name should be different. It is not important if both supply sets power and ground rails are same. Only on name basis, isolation cells are inserted.
   Consider below example:

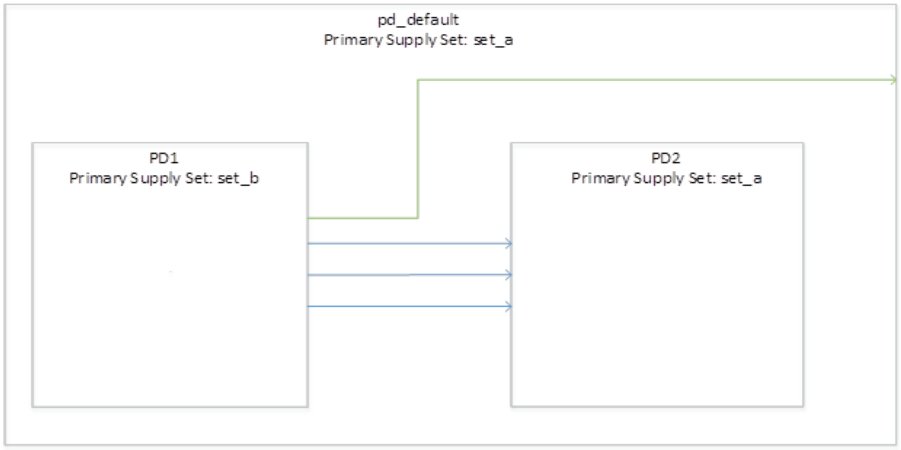


Figure 4.6: diff_supply

In above case, default domain has primary supply set set_a,PD1 has primary supply set set_b and PD2 has primary supply set set_a. Now pd_default is highest on module, PD2 is less than pd_default and PD1 is less than PD2.

pd_default ¿ PD2 ¿ PD1

In this case, isolation is required at output of PD1. And From PD1, output ports are going to PD2 and pd_default which has different supply set name than PD1. So in this case we can use diff_supply option.

set_isolation iso1 –domain PD1 –applies_to outputs –isolation_supply_set set_c –diff_supply_only TRUE –clamp_value 0

(b) Source/sink: When some ports from one domain which is going to another two domains and isolation cells are required for only one domain. And both of these domains are working on different supply sets. We can use sink option to insert isolation cells on first power domain output ports or we can use source option for that power domain for which we want isolation cells. Generally, these two options are used in conjunction with find_objects command to filter out proper ports.[17]

Consider same above example:

In above example, output ports of PD1 are going to pd_default and PD2 which are working on set_a. In this case, we can make strategy with for PD1 domain with sink option.

e.g.

set_isolation iso1 –domain PD1 –isolation_supply_set set_c –sink set_a –clamp_value 0

Or in another way we can also write isolation strategy for PD2 domain with source option.

e.g.

set_isolation iso1 –domain PD2 –isolation_supply_set set_c –source set_b –clamp_value 0

But in this case one port will left in PD1 which is going to pd_default domain so for that we can make isolation strategy with element value.

set_isolation iso1 –domain PD1 –isolation_supply_set set_c –elements port_name –clamp_value 0

Syntax:

set_isolation iso_name

–domain domain_name

–clamp_value 0/1

–source supply_set _name/ -sink supply_set_name / -diff_supply_only true

4. Power state Declaration: In UPF 1.0 standard, generally using add_port_state power states are declared for ports. In UPF 2.0 standard, power states are declared for power rails, power domains. So because of this, there is no need of power state table.[17]
   Syntax:

add_power_state supply_set_name/domain_name
–state state_name {-supply_exp {expr}
–simstate simstate_name }

4.1 UPF 2.0 Example

Same example which is given in chapter 2 is converted to UPF 2.0 and given here. Below table is showing difference between UPF 1.0 and UPF 2.0.
UPF 2.0
###UPF version Declaration
upf_version 2.0

###Supply Set Declaration
create_supply_set set_vcc_1p2
create_supply_set set_vcc_0p7
create_supply_set set_vcc_1p2_gated
create_supply_set set_vcc_0p7_gated

Power Domain Declaration
create_power_domaim PD_DEFAULT
–supply {extra_supplies_1 set_vcc_0p7}
–supply {extra_supplies_2 set_vcc_0p7_gated}
–supply {extra_supplies_3 set_vcc_1p2_gated} create_power_domain PD_GRAPHICS
–elements { graphics_module }
–supply {extra_supplies_1 set_vcc_1p2}
create_power_domain PD_INTERNET –elements { internet_module }
–supply {extra_supplies_1 set_vcc_1p2}
create_power_domain PD_DISPLAY –elements { display_module }
–supply {extra_supplies_1 set_vcc_1p2}
create_power_domain PD_TX_RX –elements { tx_rx_module }
–supply {extra_supplies_1 set_vcc_0p7}

#Supply Net & Port (1.2 v)
create_supply_net vcc_1p2
create_supply_port vcc_1p2
connect_supply_net vcc_1p2 –ports vcc_1p2
#Supply Net & Port (0.7 v)
create_supply_net vcc_0p7
create_supply_port vcc_0p7

```
connect_supply_net vcc_0p7 –ports vcc_0p7
#supply net (1.2 gated)
create_supply_net vcc_1p2_gated
#supply net (0.7 gated)
create_supply_net vcc_0p7_gated

### Supply Set Association
create_supply_set set_vcc_1p2
–function{power vcc_1p2} –function {ground vss}
create_supply_set set_vcc_0p7
–function {power vcc_0p7} –function {ground vss}
create_supply_set set_ vcc_1p2_gated
–function {power vcc_1p2_gated} –function {ground vss}
create_supply_set set_ vcc_0p7_gated
–function {power vcc_0p7_gated} –function {ground vss}

#Primary PowerGround Net Declaration
create_power_domaim PD_DEFAULT
–supply {primary set_vcc_1p2} -update
create_power_domain PD_GRAPHICS
–supply {primary set_vcc_1p2_gated}
-supply {default_isolation set_vcc_1p2}-update
create_power_domain PD_INTERNET
-supply {primary set_vcc_1p2_gated}
-supply {default_isolation set_vcc_1p2} -update
create_power_domain PD_DISPLAY
–supply {primary set_vcc_1p2_gated}
-supply {default_isolation set_vcc_1p2} -update
create_power_domain PD_TX_RX
–supply {primary set_vcc_0p7_gated}
–supply {default_isolation set_vcc_0p7} -update

#Power Switch
create_power_switch graphics_sw –domain PD_GRAPHICS
-output_supply_port {gtdout set_vcc_1p2_gated.power }
-input_supply_port {vcc_in set_vcc_1p2.power }
-control_port {a graphics_sw_en }
-on_state { SW_ON vcc_in {!a} } -off_state {SW_OFF {a} }
create_power_switch internet_sw –domain PD_INTERNET
```

-output_supply_port {gtdout set_vcc_1p2_gated.power }
-input_supply_port {vcc_in set_vcc_1p2.power }
-control_port {a internet_sw_en }
-on_state { SW_ON vcc_in {!a} } -off_state {SW_OFF {a} }
create_power_switch display_sw –domain PD_DISPLAY
-output_supply_port {gtdout set_vcc_1p2_gated.power }
-input_supply_port {vcc_in set_vcc_1p2.power }
-control_port {a display_sw_en }
-on_state { SW_ON vcc_in {!a} } -off_state {SW_OFF {a} }
create_power_switch tx_rx_sw –domain PD_TX_RX
-output_supply_port {gtdout set_vcc_0p7_gated.power }
-input_supply_port {vcc_in set_vcc_0p7.power }
-control_port {a tx_rx_sw_en }
-on_state { SW_ON vcc_in {!a} } -off_state {SW_OFF a} }

#Isolation Strategy
set_isolation graphics_iso –domain PD_GRAPHICS
–applies_to outputs –clamp_value 0
–diff_supply_only true
set_isolation_control graphics_iso –domain PD_GRAPHICS
–isolation_signal iso_graphics_enb
–isolation_sense low –location self
set_isolation internet_iso –domain PD_INTERNET
–clamp_value 0 –sink set_vcc_1p2_gated
set_isolation_control internet_iso –domain PD_INTERNET
–isolation_signal iso_internet_enb
–isolation_sense low –location self
set_isolation display_iso –domain PD_DISPlAY
–applies_to outputs –clamp_value 0
set_isolation_control display_iso –domain PD_DISPLAY
–isolation_signal iso_display_enb
–isolation_sense low –location self
set_isolation tx_rx_iso –domain PD_TX_RX
–applies_to outputs –clamp_value 0
set_isolation_control tx_rx_iso –domain PD_TX_RX
–isolation_signal iso_tx_rx_enb
–isolation_sense low –location self
#Level shifter strategy
set_level_shifter tx_rx_ls_in –domain PD_TX_RX

–applies_to inputs –location self –rule high_to_low
set_level_shifter tx_rx_ls_out –domain PD_TX_RX
–applies_to outputs –location self –rule low_to_high
#Power State
add_power_state set_vcc_1p2 –state Highvoltage_p
{-supply_expr {power == ‘{FULL_ON,1.2} –simstate NORMAL}
add_power_state set_vcc_1p2 –state Highvoltage_g
{-supply_expr {ground == ‘{FULL_ON,0.0}} –simstate NORMAL}
add_power_state set_vcc_0p7 –state Highvoltage_p
{-supply_expr {power == ‘{FULL_ON,0.7}} –simstate NORMAL}
add_power_state set_vcc_1p2_gated –state Highvoltage_p
{-supply_expr {power == ‘{FULL_ON,1.2}} –simstate NORMAL}
add_power_state set_vcc_1p2_gated –state Highvoltage_p
{-supply_expr {power == ‘{OFF}} –simstate CORRUPT}
add_power_state set_vcc_0p7_gated –state Highvoltage_p
{-supply_expr {power == ‘{FULL_ON,0.7}} –simstate NORMAL}
add_power_state set_vcc_0p7_gated –state Highvoltage_p
{-supply_expr {power == ‘{OFF}} –simstate CORRUPT}

4.2 Issues Captured in Flow/Tools during UPF 2.0 Validation

From reference created, For validation sochi_mem partition is taken and converted it's upf 1.0 to different flavors of upf 2.0. These upfs are passed to different tools and captured bugs in tools. There are 3 different flavors of upf 2.0.

1.Mix of implicit and explicit supply sets & isolation strategy with diff supply:

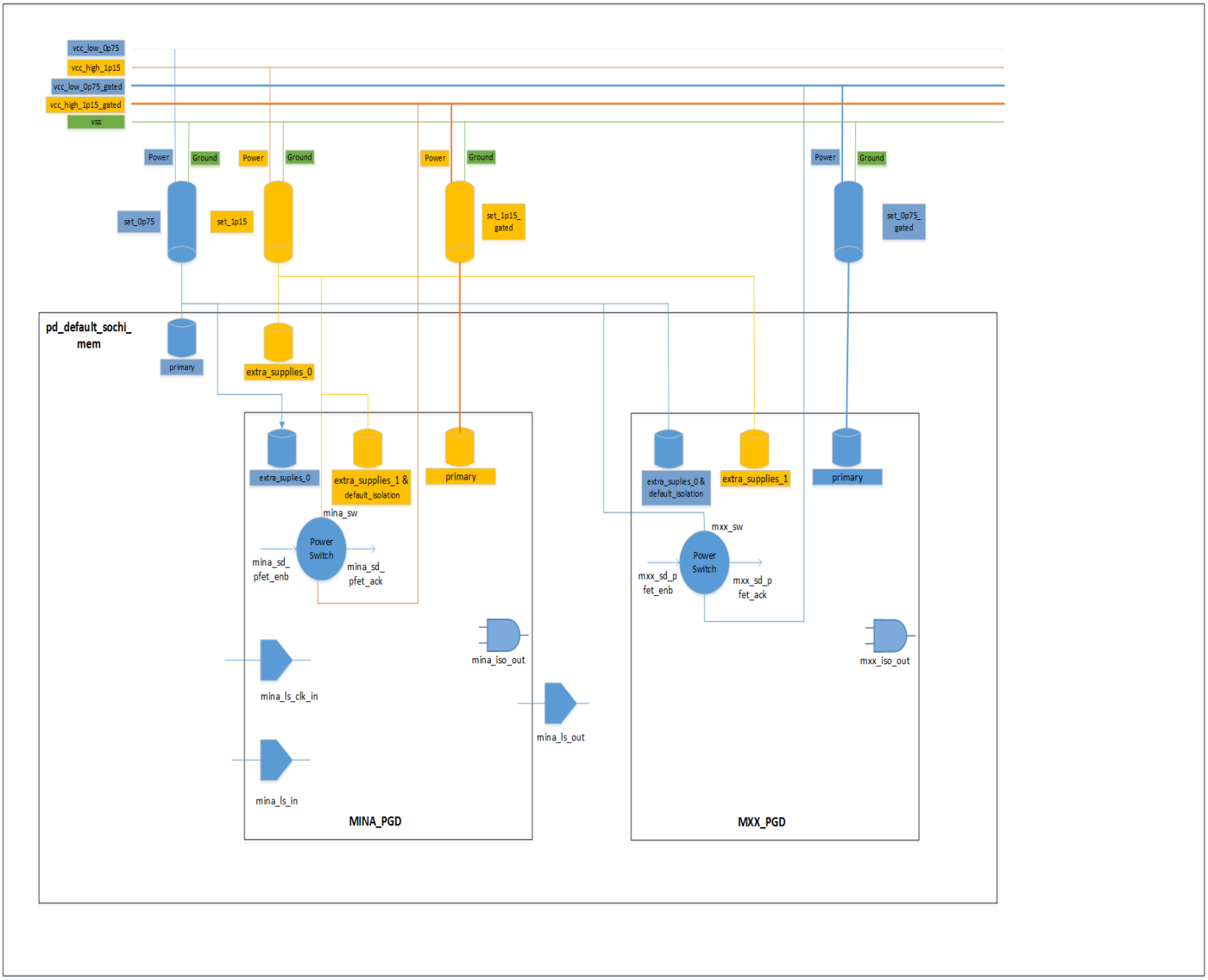


Figure 4.7: UPF diagram with mix supply_sets & diff_supply

2.Mix of implicit and explicit supply sets & isolation strategy with source and sink

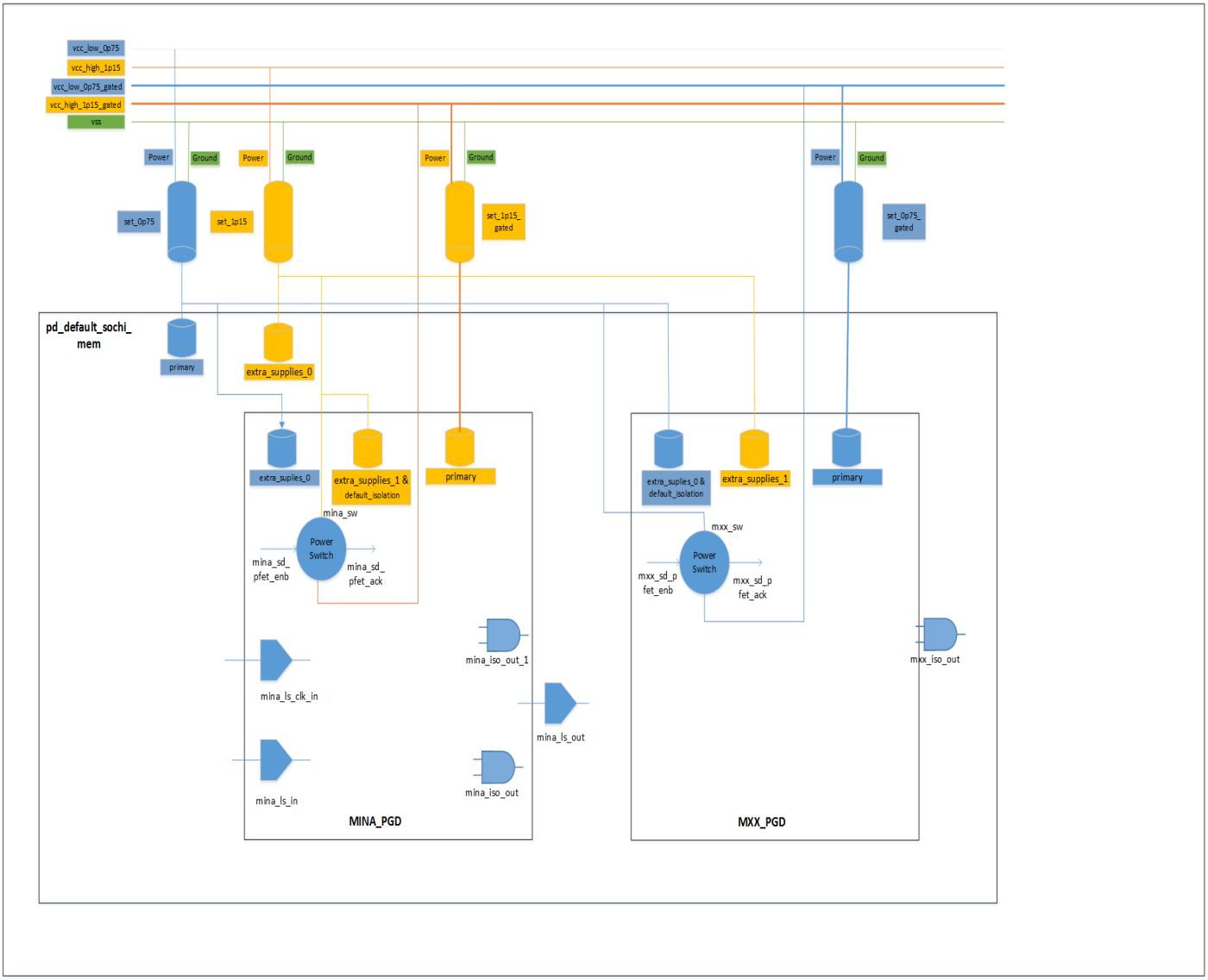


Figure 4.8: UPF diagram with mix supply_sets & source/sink

3.Explicit supply sets & isolation strategy with diff supply:

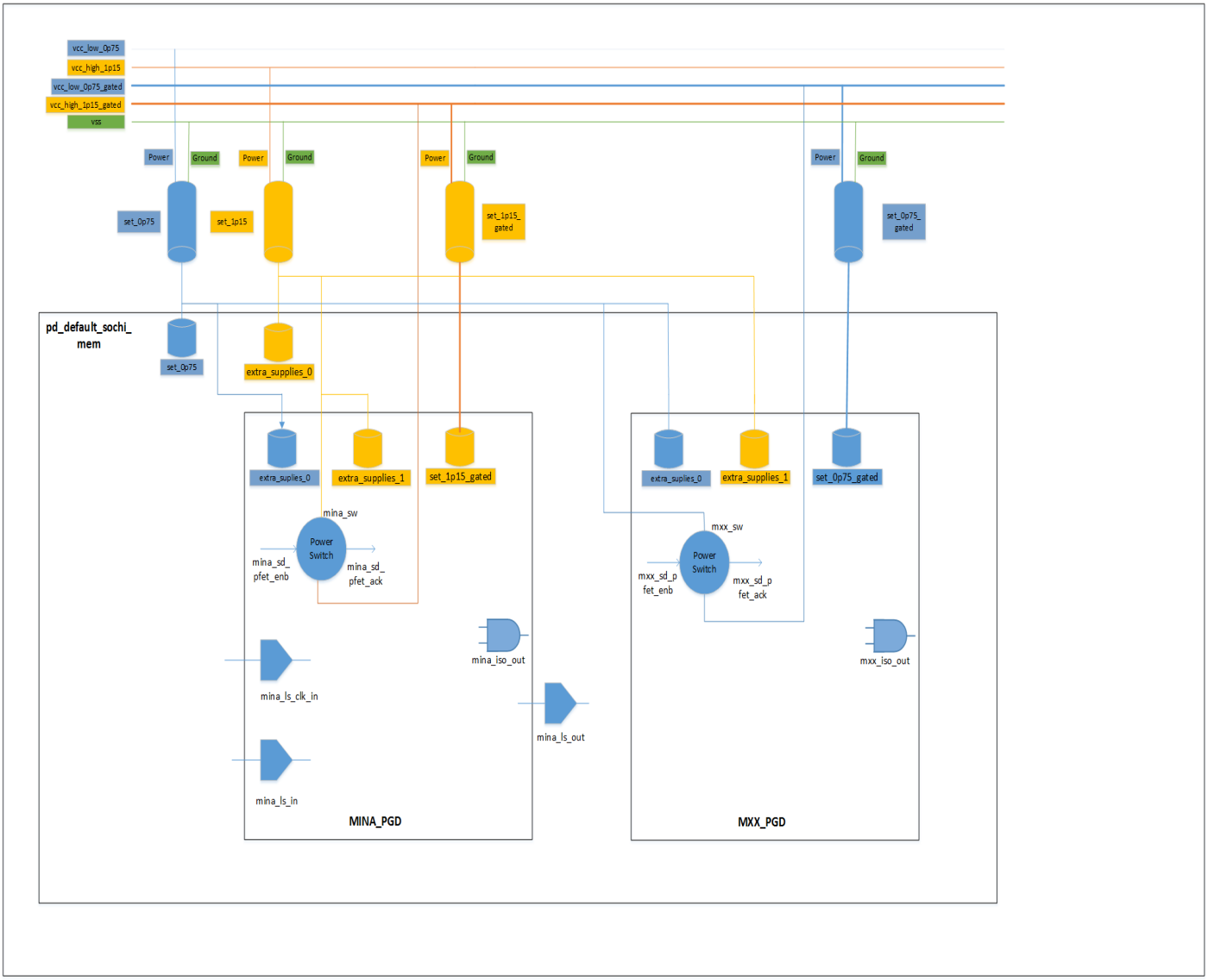


Figure 4.9: UPF diagram with explicit supply set & diff supply

1. Issues captured during Lint Checking:

   (a) Run Type: RTL + UPF Error: All states are not used in Power State Table
   Reason: In UPF 2.0 standard, add_port_state is not used instead of that add_power_state is used to define power states of power rails. But during run, tool is adding extra add_port_state commands and then states which are used in add_port_state command is not used in power state table so tool is giving error that state is not used in power state table.

```
Error: All states are not used in Power State Table
Error_information: State FULL_ON of supply port vcc_low_0p75 is not used in power state table sochi_mem_pst
Line: 20
Error_information: State FULL_ON of supply port vcc_high_1p15 is not used in power state table sochi_mem_pst
Line: 24
Error_information: State FULL_ON of supply port vss is not used in power state table sochi_mem_pst
Line: 28
Error_information: State FULL_ON of supply port gtdout of power switch mina_sw is not used in power state table sochi_mem_pst
Line: 51
Error_information: State FULL_ON of supply port gtdout of power switch mxx_sw is not used in power state table sochi_mem_pst
Line: 60
```

Figure 4.10: Lint checking issue 1

   (b) Run Type: RTL + UPF
   Generally, in UPF 2.0, we can use find_objects command to get different port names instead of giving each port name as part of –elements switch in different strategy command. We can also use wildcards *,? with patterns to get port name. But tool is not returning proper port names during find_objects usage.
   e.g. We are getting some signals which are unwanted. We have given pfet_current_fuse [*] it should report only pfet_current-
   _fuse [0-7] but it is reporting pfet_current_- _fuse[0-7] and pfet-
   _current_fuse_muxd[0-7]. Also some of signals are not able to find.

```
set_port_attributes -ports [find_objects . -pattern {data0_bus_addr*} -object_type port -direction in]
-driver_supply set_0p75
#set_port_attributes -ports [find_objects . -pattern {data0_b?s_data???} -object_type port -direction i
n] -driver_supply set_0p75
set_port_attributes -ports [find_objects . -pattern {data2_bus_addr\[0\]} -object_type port -direction
in] -driver_supply set_0p75
set_port_attributes -ports [find_objects . -pattern {data2_bu?_data\[*\]} -object_type port -direction
in] -driver_supply set_0p75
```

```
set_port_attributes -ports { /data0_bus_addr }  -driver_supply set_0p75
set_port_attributes -driver_supply set_0p75
set_port_attributes -driver_supply set_0p75
```

Figure 4.11: Lint checking issue 2

(c) Run Type: P&R Netlist + UPF"
Error: add_port_state command is missing in UPF
Reason: In our input UPF we don't have add-port-state commands. The tool infers internally add_port_state command from add_power_state command and issues error messages.

```
Error: add_port_state command is missing in UPF
Error_information: add_port_state command for port vcc_low_0p75 is not available in UPF
Line: 15
Error_information: add_port_state command for port vcc_high_1p15 is not available in UPF
Line: 17
Error_information: add_port_state command for port vss is not available in UPF
Line: 19
```

Figure 4.12: Lint checking issue 3

(d) Run Type: P&R Netlist + UPF"
Error: Undefined state in power state table
Reason: Here, tool is not able to get supply value of ground rail 0.0 and declaring undefined state for ground rail.

```
Error: Undefined state in power state table
Error_information: State of supply port vss is undefiend for ALL_ON in power state table
Line: 57
Error_information: State of supply port vss is undefiend for MXX_ON in power state table
Line: 58
Error_information: State of supply port vss is undefiend for MINA_ON in power state table
Line: 59
Error_information: State of supply port vss is undefiend for PRE_BOOT in power state table
Line: 60
```

Figure 4.13: Lint checking issue 4

(e) Supply_connection_report:
This report contains all ports which have related supply net set on those. This

report is not giving any port names which are one dimensional. Also it is giving only top level module ports which has related supply net. If any lower level module pins have related supply net it will not give any value.

(f) Giving warnings on redundant isolation cells
Here, in upf isolation cells are inserted and these isolation cells are required isolation cells. But tool is giving isolation cells are redundant.As shown in diagram, MINA_PGD has isolation cell. Level shifter is placed in default power domain. And default_power_domain is more always on than MINA_PGD. so it requires isolation cell but tool is giving redundant isolation cell.

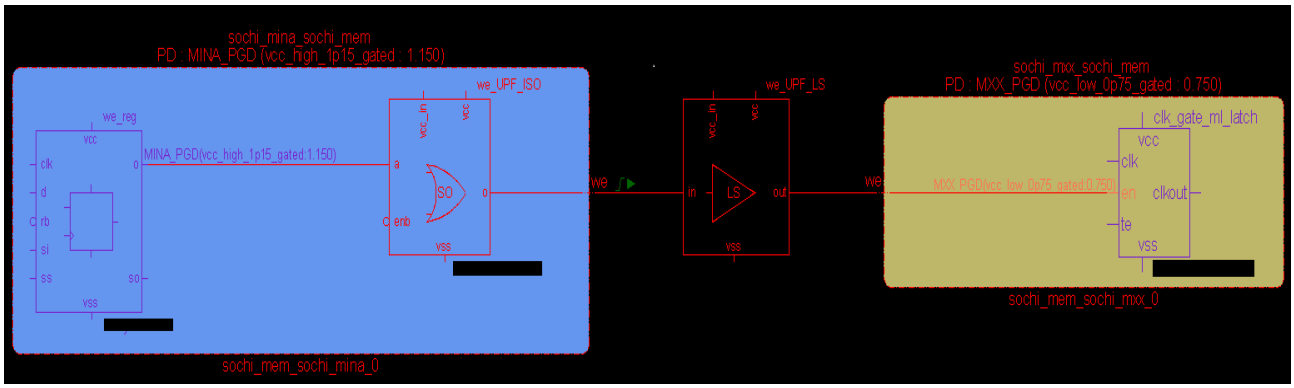


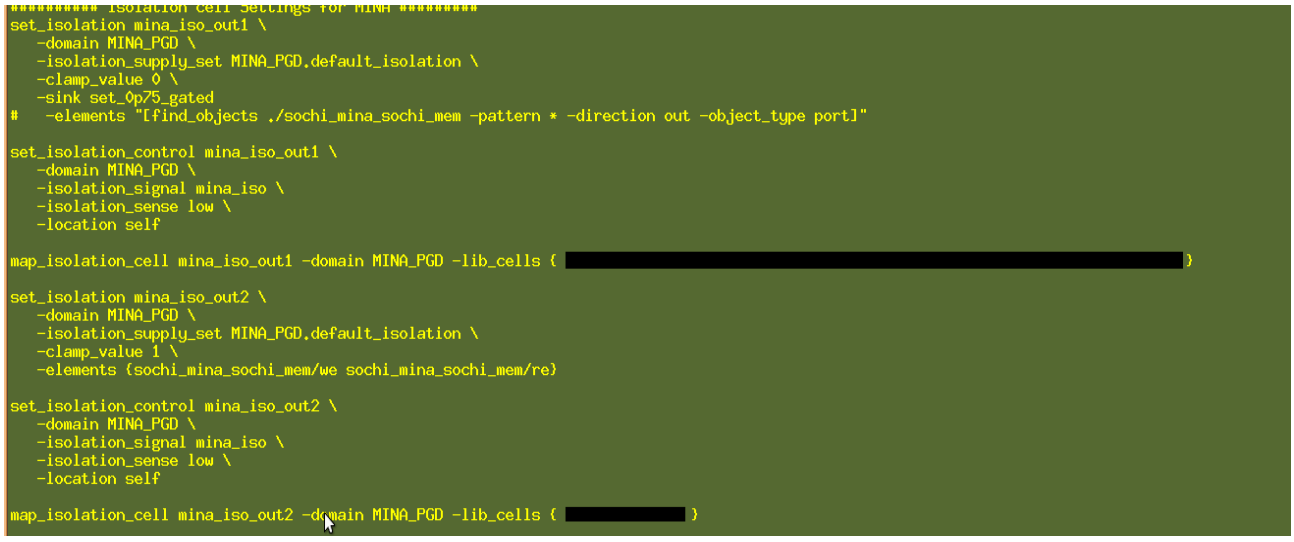
Figure 4.14: Lint checking issue 6

2. Issues captured during Synthesis:

   (a) Synthesis tool is not able to pick up proper isolation cell. If we place isolation cell as self in power domain, it needs dual rail isolation cell and if we place as parent, it needs single rail isolation cell. Without map isolation cell command tool is not able to pick up proper isolation cell. Every time it is picking up single rail isolation cell.

   (b) check_mv_design Report: This report checks low power information. This report is not giving any error regarding wrong isolation cell picked up.

   (c) Isolation strategy is not working properly with find_objects.
   As shown in figure 4.15, There are two isolation strategies, one with sink option and AND isolation cells and another with elements list and OR gate isolation cells. In this case,tool is inserting proper types of isolation cells.
   As shown in figure 4.16, situation is first isolation strategy is defined with find_objects and sink option with AND gate isolation cells. Now, second isolation strategy is defined with elements which has two elements from first strategy with OR gate isolation cell. Tool is not inserting OR gate isolation cells and inserting AND gate isolation cells.
   As shown in figure 4.17, order of isoaltion strategy definition is changed. But still tool is inserting AND gate isolation cells.
   As shown in figure 4.18,another situation is first isolation strategy is defined with find_objects with AND gate isolation cells. And second isolation strategy is defined with elements list which has again two ports from first isolation strategy. And here tool is not considering second isolation strategy and giving error of ports already used in another isolation strategy.



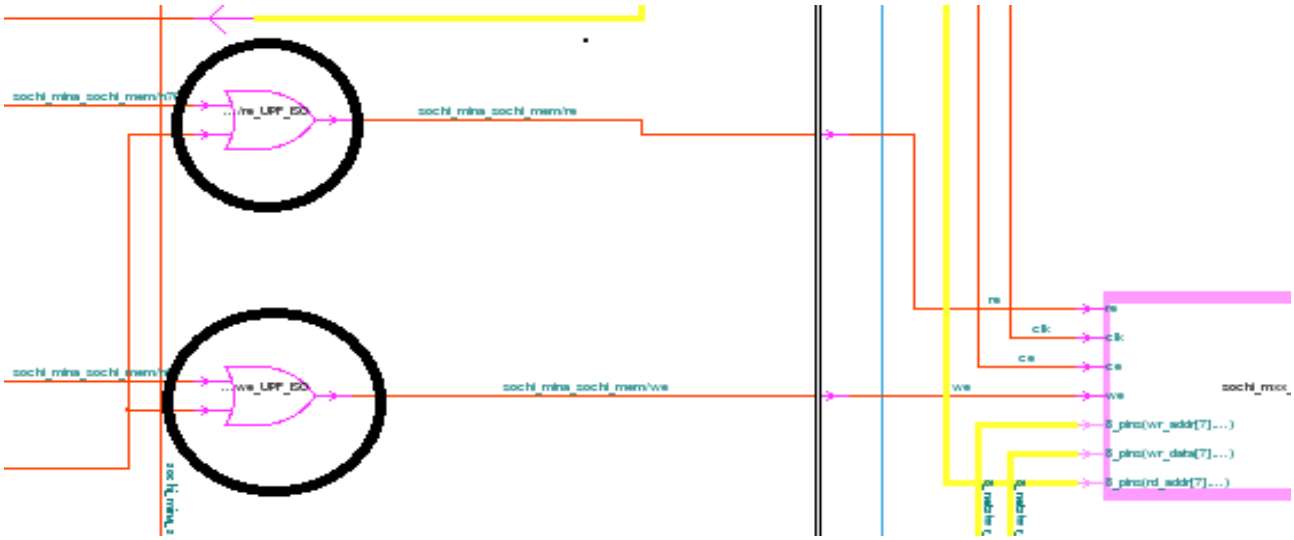


Figure 4.15: Synthesis tool issue 3

```
########## Isolation cell settings for MINA ##########
set_isolation mina_iso_out1 \
   -domain MINA_PGD \
   -isolation_supply_set MINA_PGD.default_isolation \
   -clamp_value 0 \
   -sink set_0p75_gated \
   -elements "[find_objects ./sochi_mina_sochi_mem -pattern * -direction out -object_type port]"

set_isolation_control mina_iso_out1 \
   -domain MINA_PGD \
   -isolation_signal mina_iso \
   -isolation_sense low \
   -location self

map_isolation_cell mina_iso_out1 -domain MINA_PGD -lib_cells { [redacted] }

set_isolation mina_iso_out2 \
   -domain MINA_PGD \
   -isolation_supply_set MINA_PGD.default_isolation \
   -clamp_value 1 \
   -elements {sochi_mina_sochi_mem/we sochi_mina_sochi_mem/re}

set_isolation_control mina_iso_out2 \
   -domain MINA_PGD \
   -isolation_signal mina_iso \
   -isolation_sense low \
   -location self

map_isolation_cell mina_iso_out2 -domain MINA_PGD -lib_cells { [redacted] }
```

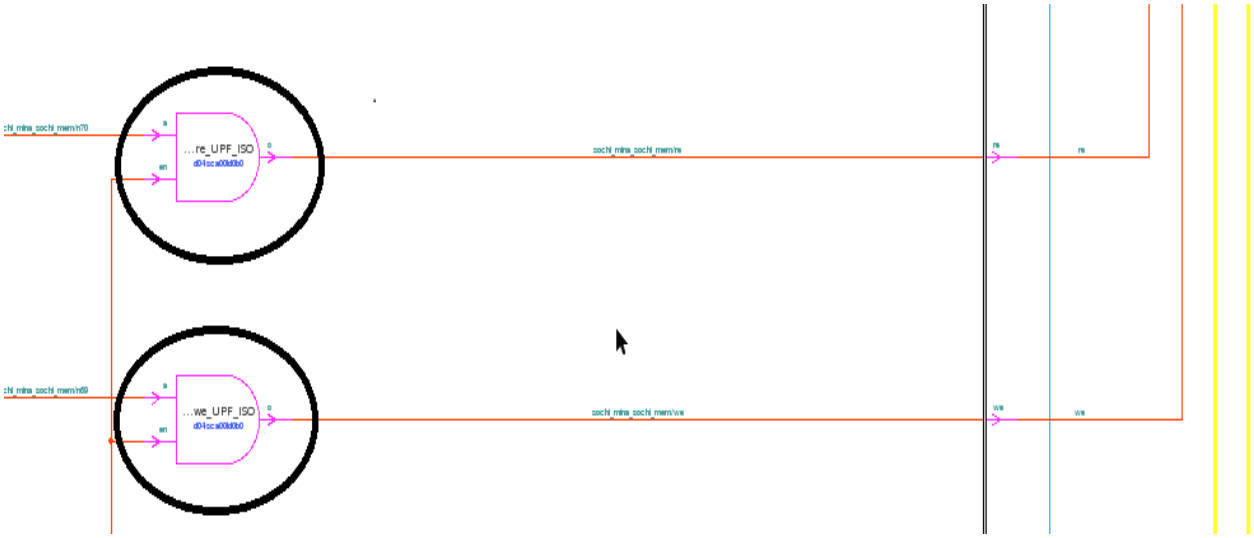


Figure 4.16: Synthesis tool issue 3

```
########## Isolation cell settings for MINA ##########
set_isolation mina_iso_out2 \
   -domain MINA_PGD \
   -isolation_supply_set MINA_PGD.default_isolation \
   -clamp_value 1 \
   -elements {sochi_mina_sochi_mem/we sochi_mina_sochi_mem/re}

set_isolation_control mina_iso_out2 \
   -domain MINA_PGD \
   -isolation_signal mina_iso \
   -isolation_sense low \
   -location self

map_isolation_cell mina_iso_out2 -domain MINA_PGD -lib_cells { [redacted] }

set_isolation mina_iso_out1 \
   -domain MINA_PGD \
   -isolation_supply_set MINA_PGD.default_isolation \
   -clamp_value 0 \
   -sink set_0p75_gated \
   -elements "[find_objects ./sochi_mina_sochi_mem -pattern * -direction out -object_type port]"

set_isolation_control mina_iso_out1 \
   -domain MINA_PGD \
   -isolation_signal mina_iso \
   -isolation_sense low \
   -location self

map_isolation_cell mina_iso_out1 -domain MINA_PGD -lib_cells { [redacted] }
```

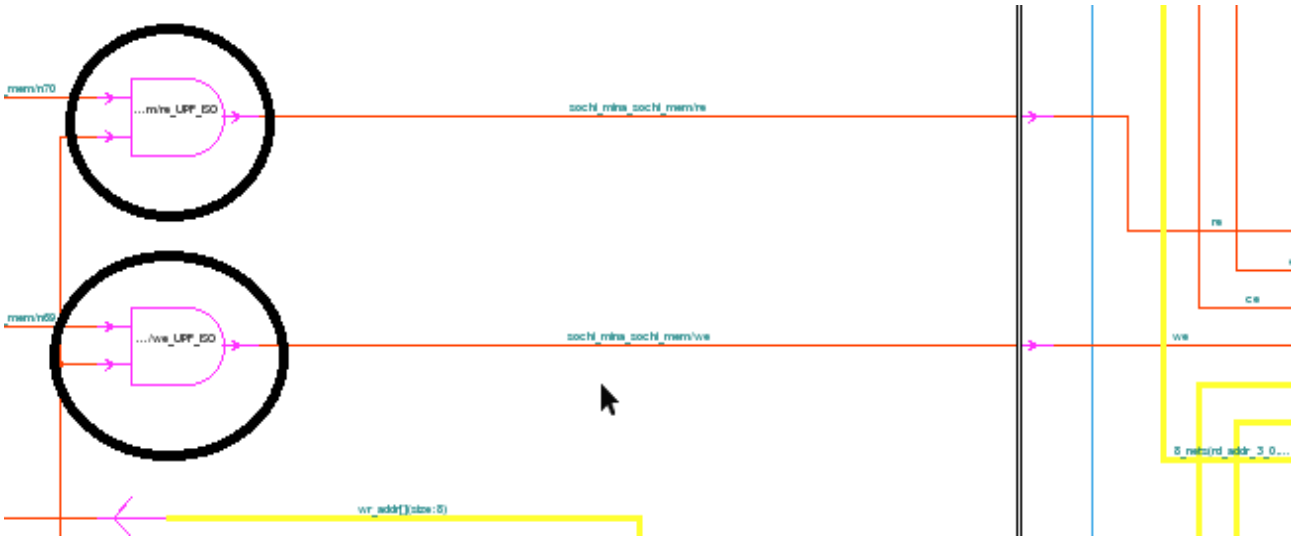


Figure 4.17: Synthesis tool issue 3

```
########## Isolation cell Settings for MINA ##########
set_isolation mina_iso_out1 \
   -domain MINA_PGD \
   -isolation_supply_set MINA_PGD.default_isolation \
   -clamp_value 0 \
   -elements "[find_objects ./sochi_mina_sochi_mem -pattern * -direction out -object_type port]"
#   -sink set_0p75_gated \
#   -elements "[find_objects ./sochi_mina_sochi_mem -pattern * -direction out -object_type port]"

set_isolation_control mina_iso_out1 \
   -domain MINA_PGD \
   -isolation_signal mina_iso \
   -isolation_sense low \
   -location self

map_isolation_cell mina_iso_out1 -domain MINA_PGD -lib_cells { [redacted] }

set_isolation mina_iso_out2 \
   -domain MINA_PGD \
   -isolation_supply_set MINA_PGD.default_isolation \
   -clamp_value 1 \
   -elements {sochi_mina_sochi_mem/we sochi_mina_sochi_mem/re}

set_isolation_control mina_iso_out2 \
   -domain MINA_PGD \
   -isolation_signal mina_iso \
   -isolation_sense low \
   -location self

map_isolation_cell mina_iso_out2 -domain MINA_PGD -lib_cells { [redacted] }
```

```
set_isolation mina_iso_out2 -domain MINA_PGD -isolation_supply_set MINA_PGD.default_isolation -clamp_value 1 -elements {sochi_mina_sochi_mem/we sochi_min
a_sochi_mem/re}
Error: Port sochi_mina_sochi_mem/we is already defined in another isolation strategy
Error: Port sochi_mina_sochi_mem/re is already defined in another isolation strategy
```

Figure 4.18: Synthesis tool issue 3

3. Issues captured during P&R:

   (a) Without using map_isolation command in upf, synthesis tool is not picking up proper isolation cell. And P&R tool is connecting wrong power rail connection for this isolation cell. In this, isolation cell is placed as self in power domain. So, dual rail isolation cell is required. So it will be connected with gated rail and always on power rail. But as shown in figure 4.19, isolation cell is connected with gated rail only.

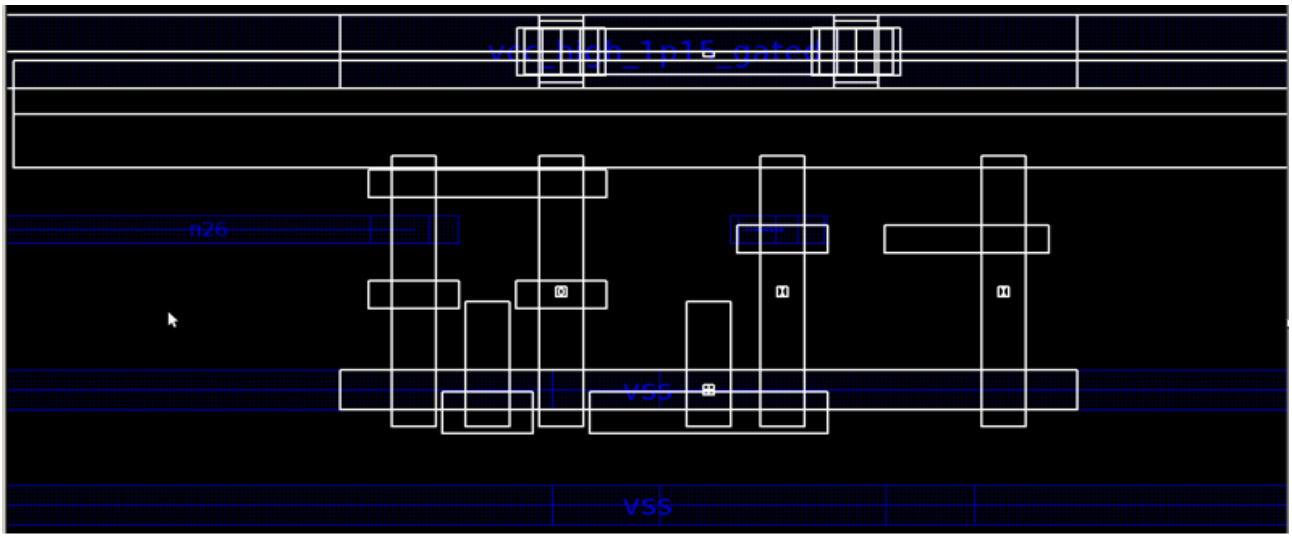


Figure 4.19: P&R tool issue 1

   (b) Verilog dump out by P&R tool is giving wrong power rail name for isolation cell. And it is connected with another power rail. As sown in figure 4.19, isolation cell is connected to gated rail from schematic but in verilog file it is connected to always on rail.

```
        rd_addr_5_UPF_ISO (.a ( n80 ) , .b ( n122 ) , .o ( rd_addr[5] )
, .vcc ( vcc_high_1p15 ) , .vss ( vss ) ) ;
        rd_addr_4_UPF_ISO (.a ( n81 ) , .b ( n122 ) , .o ( rd_addr[4] )
, .vcc ( vcc_high_1p15 ) , .vss ( vss ) ) ;
        rd_addr_3_UPF_ISO (.a ( n8200 ) , .b ( n100 ) , .o ( rd_addr[3] )
, .vcc ( vcc_high_1p15 ) , .vss ( vss ) ) ;
        rd_addr_2_UPF_ISO (.a ( n83 ) , .b ( n100 ) , .o ( rd_addr[2] )
, .vcc ( vcc_high_1p15 ) , .vss ( vss ) ) ;
        rd_addr_1_UPF_ISO (.a ( n84 ) , .b ( n100 ) , .o ( rd_addr[1] )
, .vcc ( vcc_high_1p15 ) , .vss ( vss ) ) ;
        rd_addr_0_UPF_ISO (.a ( n85 ) , .b ( n100 ) , .o ( rd_addr[0] )
, .vcc ( vcc_high_1p15 ) , .vss ( vss ) ) ;
```

Figure 4.20: P&R tool issue 2

   (c) P&R tool is changing upf and adding associate_supply_set commands also replacing supply sets with actual power rail names. If we have used -supply switch in

create_power_domain command to associate supply sets with implicit sypply handles. Tool is replacing that -supply switch with associate_supply_set command.

```
create_power_domain pd_default_sochi_mem -include_scope -supply {extra_supplies_0 set_1p15}
create_power_domain MXX_PGD -elements {sochi_xma_sochi_mem/FT__U1 sochi_mina_sochi_mem/DHC_MXX_PGD DHC_
MXX_PGD sochi_mxx_sochi_mem FT__U1 } -supply {extra_supplies_0 set_0p75} -supply {extra_supplies_1 set_
1p15}
create_power_domain MINA_PGD -elements sochi_mina_sochi_mem -supply {extra_supplies_0 set_0p75} -supply
 {extra_supplies_1 set_1p15}
associate_supply_set set_0p75 -handle pd_default_sochi_mem.primary
associate_supply_set set_0p75_gated -handle MXX_PGD.primary
associate_supply_set set_0p75 -handle MXX_PGD.default_isolation
associate_supply_set set_1p15_gated -handle MINA_PGD.primary
associate_supply_set set_1p15 -handle MINA_PGD.default_isolation
```

```
create_power_domain pd_default_sochi_mem -supply { primary set_0p75 } -update
create_power_domain MXX_PGD -supply {primary set_0p75_gated} -supply {default_isolation set_0p75} -upda
te
create_power_domain MINA_PGD -supply {primary set_1p15_gated} -supply {default_isolation set_1p15} -upd
ate
```

Figure 4.21: P&R tool issue 3

4. Issues captured during Formal Equivalence Checking:

   (a) Wrong isolation cells are inferred during RTL vs Syntehsis run.
   Isolation strategy is defined with source and sink option. Design has total 43 isolation cells. Tool is inferring 63 isolation cells. It is inferring back to back 2 isolation cell for each port.

   (b) Tool is not able to identify power rail associated with supply set and giving error related supply set. As shown in figure 4.22, here in UPF actual supply rails are used as set_domain_supply_net and in UPF' it is replaced with equivalent supply sets. But tool is not able to identify these and giving non equivalent points related to supply set.

```
Non-equivalent: Supply set handle MXX_PGD.primary is not available golden side
Golden Value: set_0p75_gated
Revised Value: MXX_PGD.primary
```

```
set_domain_supply_net pd_default_sochi_mem
   -primary_power_net set_0p75.power
   -primary_ground_net set_0p75.ground
set_domain_supply_net MXX_PGD
   -primary_power_net set_0p75_gated.power
   -primary_ground_net set_0p75_gated.ground
set_domain_supply_net MINA_PGD
   -primary_power_net set_1p15_gated.power
   -primary_ground_net set_1p15_gated.ground
```

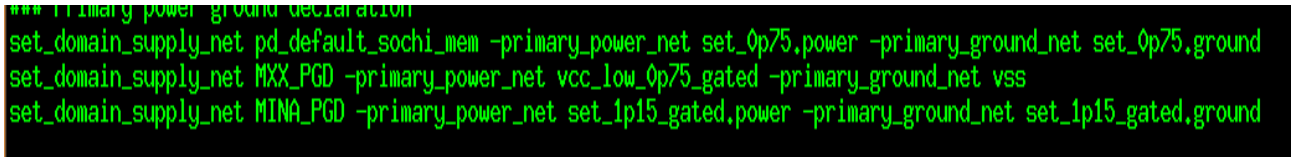


Figure 4.22: fev tool issue 2

Chapter 5

Conclusion

5.1 Conclusion

EDA vendor tools constantly strive to add new features, support new standards keeping in pace with VLSI company requirements. Complexity of handling these tools in the design industry is a constant battle. Design communities spend a lot of time qualifying these tools & features before introducing in the work-flow. With UPF usage gaining momentum, qualifying tools for 1.0 support and extending it to 2.0 readiness was a good challenge. UPF impacting all phases of design was first flow-flushed with a reference design before taking it on actual. While results seen with tools, using 1.0, was comparable with expectations, with 2.0, the tools still seem to be not fully mature. Moving to latest standard meant results matching the older standard. However the observation was difference in performance and interpretation of these standards. On-going work with EDA companies to understand tool behavior and match our expectations.

Bibliography

[1] 1801-2009 – IEEE Standard for Design and Verification of Low Power Integrated Circuits, March 2009, DOI=10.1109/IEEESTD.2009.4809845, http://ieeexplore.ieee.org/servlet/opac?punumber=4809843

[2] http://www.scribd.com/doc/230569103/UPF-Fundamentals-Handouts

[3] Synthesis ABCs

[4] Synthesis tool user guide

[5] UPF exploration tool user guide

[6] solvnet.synopsys.com

[7] Leakage Power Minimization Techniques for Embedded Processors by Jatin Nawnit Mistry

[8] A Clock-Gated, Double Edge Triggered Flip Flop Implemented With Transmission Gates By Xiaowen Wang

[9] An ASIC Low Power Primer Analysis, Techniques and Specifications by Rakesh Chadha and J. bhasker

[10] Low Power Methodology Manual For System on Chip Design by Michael Keating, David Flynn, Robert Altken, Alan Gibbons, Kaijian Shi

[11] www.asic-soc.blogpost.in

[12] http : //usebackend:wordpress:com/2012/12/18/clock-tree-synthesis/

[13] Low power rule checking tool user guide

[14] Physical Implementation tool user guide

[15] Formal Equivalence Checking tool user guide

[16] Development of Algorithm for integration of clock gating and power gating techniques by Gaurang Upasani

[17] 1801-2013 – IEEE Standard for Design and Verification of Low Power Integrated Circuits, May 2013, 10.1109/IEEESTD.2013.6521327, http://ieeexplore.ieee.org/servlet/opac?punumber=6521325