Low Power RTL Design and RTL quality Check

By

Govil M. Badghare

Roll No: 10MECV22

Department of Electronics and Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382481

May 2012

Low Power RTL Design and RTL quality Check

Major Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology

In

Electronics and Communication Engineering

(VLSI Design)

By

Govil M. Badghare

(10MECV22)

Under the Guidance of

Devineni Narasimharao and P Vishnu M

Intel Technology India Pvt. Ltd.

Department of Electronics and Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382481

May 2012

Declaration

This is to certify that

i) 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.

ii) Due acknowledgement has been made in the text to all other material used.

Govil M. Badghare

Certificate

This is to certify that the Major Project entitled **" Low Power RTL Design and RTL quality Check"** submitted by **Govil M. Badghare(10MECV22)**, towards the partial fulfillment of the requirements for the degree of **Master of Technology in VLSI Design** of **Nirma University of Science and Technology, Ahmedabad** is the record of work carried out by him under my supervision and guidance. In my 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 my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: ____________________	**Place: Ahmedabad**
External Guide	**HOD**
____________________	____________________
Devineni Narasimharao and P Vishnu M	**Prof. A. S. Ranade**
Intel Technology India Pvt. Ltd.,	**Professor, EC**
Director	**Internal Guide**
____________________	____________________
Dr.K.R.Kotecha	**Prof. Amisha Naik**
Director, IT, NU,	**Professor (VLSI Design)**
	Nirma University

Acknowledgements

Apart from the efforts of me, the success of this project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to those people who have been instrumental in the successful completion of this project.

I would like to thank my manager **Mr. Amanna,Santosh Kumar** , **Platform Architect,**,Intel Technology India Pvt. Ltd., Bangalore, for providing me an opportunity to perform an Internship which helped me to complete this project work and an opportunity to work in such an exciting and upcoming field of SOC design.

I would like to thank my **mentors Devineni Narasimharao and P Vishnu M**, Intel Technology India Pvt. Ltd, for their valuable guidance and useful suggestions throughout this project work.For their timely guidance, support and also allowing me to have faithful discussions throughout the project work.

I would like to thank My All The Team members at Intel Technology India Pvt. Ltd.,for their excellent direction and guidance for my project work. In addition, a special thanks to Management of Intel Technology India Pvt. Ltd, Bangalore for providing me with all the required facilities.

I would like to express my sincere thanks to **Dr. Ketan Kotecha**, Director, Nirma Institute of Technology for his kind patronage.

I would like to thank my internal guide **Mrs. Amisha Naik** for guiding, constant encouragement, support and suggestions for improvement during the course of my project. His encouraging words brought out the best in me.

Finally we thank all the members of the M.Tech.(VLSI Design) Engineering Department, Nirma Institute of Technology, who has been a source of great help in our venture.

- Govil M. Badghare

10MECV22

Abstract

- The project combines the various sensor together.For this purpose we require communication between GPIO,IP's and CHIP core so we have to build a block which will allow to communicate these three blocks with each other.So I have designed the block called DATAPATH SELECTION MODULE .It will allow to operate the GPIO in I/O mode ,normal mode and functional mode.The RTL design is in verilog and it is successfully verified and synthesized. Also the project contains RTL quality check which checks the quality of RTL code that follows the certain rules according to LANGUAGE REFERENCE MANUAL.In this project I performed the LINTRA on each of the partition and on top RTL.I report all the error and warnings to a designer and fixed error and warning which affect the design quality.By doing the LINTRA on RTL design it is easy to synthesize the design.

- UPF is designed to reflect the power intent of a design at a relatively high level. UPF scripts describe which power rails should be routed to individual blocks, when blocks are expected to be powered up or shut down, how voltage levels should be shifted as signals cross from one power domain to another and whether measures should be taken to retain register and memory-cell contents if the primary power supply to a domain is removed.When the blocks are power down how to apply the isolation ?All these low power related issue can be solved using UPF.

Contents

List of Figures

Chapter 1

What is SoC?

A system on a chip or system on chip (SoC or SOC) is an Integrated circuit(IC) that integrates all components of a computer or other electronic system into a single chip. It may contain digital,analog or mixed - signal and often radio-frequency functions-all on a single chip substrate.

The VLSI manufacturing technology advances has made possible to put millions of transistors on a single die. This enables designers to put systems on a single chip thus moving everything from board to single chip resulting to growth of system on chip (SOC) technology.SOC design includes efforts to integrate heterogeneous or different types of silicon IPs (intellectual properties) on to the same chip, like memory, uP, random logics, and analog circuitry. SOC often incorporates analog components, and can also include opto/microelectronic mechanical system components in the future. An SOC design provide significant advantages in terms of speed , area ,reliability ,security and power however suffers from high system complexity , fabrication costs and increase verification requirements. However increasing reusability of IP and highly sophisticated tools (hardware/software) are making the SOC designs an extremely favorable and exciting option for modern applications.

Chapter 2

INTRODUCTION TO RTL

In integrated circuit design, register transfer level (RTL) is a level of abstraction used in describing the operation of a synchronous digital circuit. In RTL design, a circuit's behavior is defined in terms of the flow of signals (or transfer of data) between hardware registers, and the logical operations performed on those signals. Register transfer level abstraction is used in hardware description languages (HDLs) like Verilog and VHDL to create high-level representations of a circuit, from which lower-level representations and ultimately actual wiring can be derived.

2.1 REGISTER TRANSFER LOGIC

When designing digital integrated circuits with a hardware description language, the designs are usually engineered at a higher level of abstraction than transistor level (logic families) or logic gate level. In HDLs the designer declares the registers (which roughly correspond to variables in computer programming languages), and describes the combination logic by using constructs that are familiar from programming languages such as if-then-else and arithmetic operations. This level is called register transfer level. The term refers to the fact that RTL focuses on describing the flow of signals between registers. A synchronous circuit consists of two kinds of elements: registers and combinational logic. Registers (usually implemented as D flip-flops) synchronize the circuit's operation to the edges of the clock signal, and are the only elements in the circuit that have memory properties. Combinational logic performs all the logical functions in the circuit and it typically consists of logic gates. Using an EDA tool for synthesis, this description can usually be directly translated to an equivalent hardware implementation file for an ASIC or an FPGA. The synthesis tool also performs logic optimization. At the register transfer level, some types of circuits can be recognized. If there is a cyclic path of logic from a register's output to its input (or from a set of registers outputs to its inputs), the circuit is called a state machine or can be said to be sequential logic. If there are logic paths from a register to another without a cycle, it is called a pipeline.

2.2 REGISTER TRANSFER LOGIC MODEL

In the register transfer level model we split the complete system state up into registers and consider the flow of information 'in bulk' from one register to the next on each clock tick.

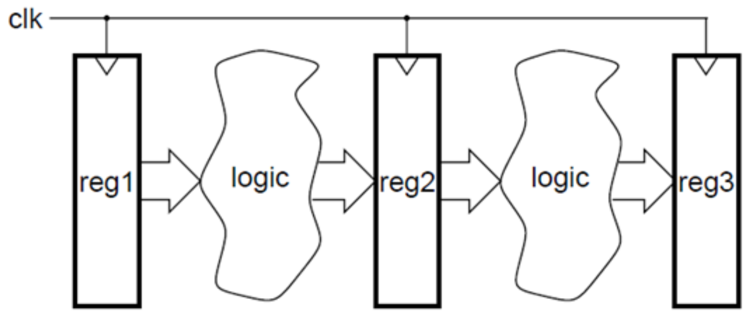


Figure 2.1: RTL Model

2.2.1 TIMING REQUIREMENTS IN RTL DESIGN

- **Setup time** : Time for which the register inputs must be constant before the corresponding clock edge.
- **Hold time** : Time for which the register inputs must be constant after the corresponding clock edge.

Chapter 3

TOP LEVEL DIAGRAM TEST CHIP

3.1 Block diagram

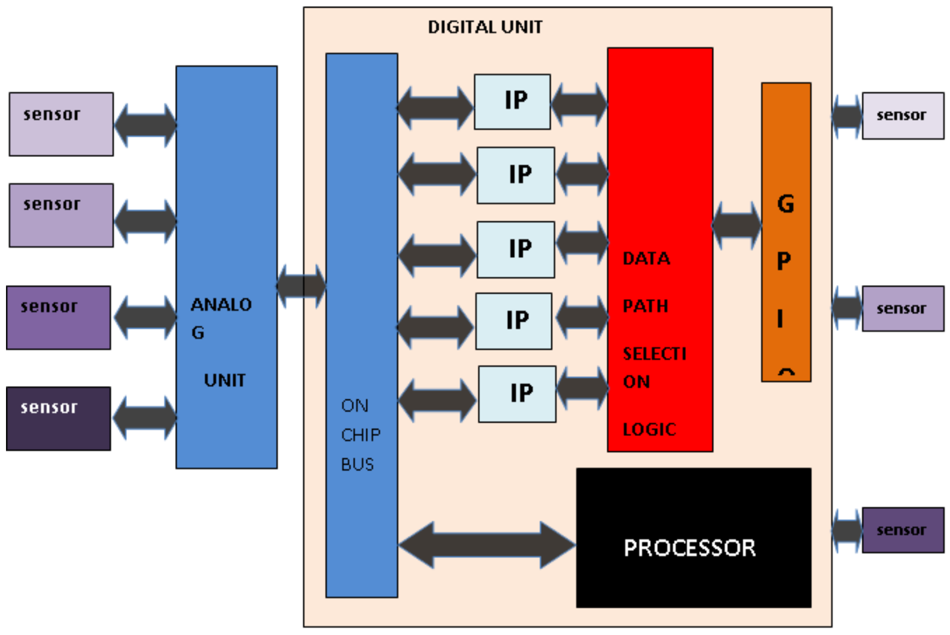


Fig 2 – BLOCK DIAGRAM OF SENSORS PROJECT

Figure 3.1: Block Level Diagram Of Test Chip

3.2 DESCRIPTION

The above schematic shows the generalized block diagram for the project. It consists of 2 basic units

- Analog Front Unit
- Digital Control Unit

Sensors are attached both to the analog as well as digital side of the design and these sensor data is sampled and given to the different IP's to processing. The IP's talk to each other through an on chip bus which can be easily seen in the rtl code. Transactions take place using various protocols and interfaces present in the chip level design. The SoC architecture contains various devices like DMA, Memory, microcontroller and diff types of IPs with the connectivity bridges. Mainly these bridges are just connectivity between the various IPs for compatibility purpose. One IPs information may be not suit to communicate with the another IPs, so bridge can acts as mediator to accommodate the service.

3.3 DATAPATH SELECTION LOGIC

3.3.1 DESCRIPTION

This is the block diagram for datapath selection logic. GPIO's are placed at the top and bottom side of the chip.there are total 9 GPIO's with 160 I/O pads.chip contains GPIO's ,IP and chip core logic,to have the communication between these GPIO ,IP and chip core logic ,we are designing the one of the partition for our chip called DATAPATH SELECTION LOGIC. In other language we can say it is acting as a communication bridge between GPIO and CHIP CORE LOGIC.As we are using GPIO at top and bottom side we designed top and bottom datapath selection logic.

3.4 BLOCK DIAGRAM

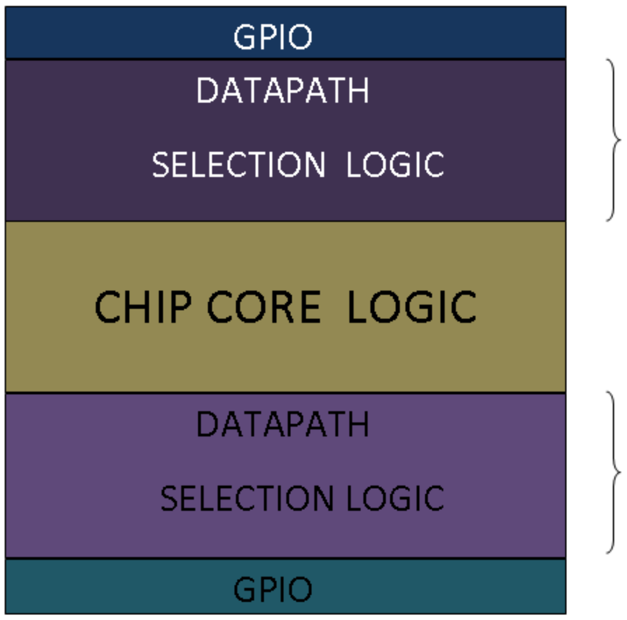


Fig 3. – INTERCONNECTION OF DATATPATH SELECTION LOGIC

Figure 3.2: Block Level Diagram Of Datapath Module

3.5 MUX BASED DATAPATH SELECTION LOGIC

The above fig shows that how the communication between GPIO pads and chip core logic takes place.GPIO has input and output direction.If it is outpu,t the path is chip core mux gpio pads.If it is input demux will take care of this path.MUX and DEMUX both share the same selection register line.The direction of GPIO as input and output is taken care by direction mux.It consist of input and output buffer with a particular IP enable.

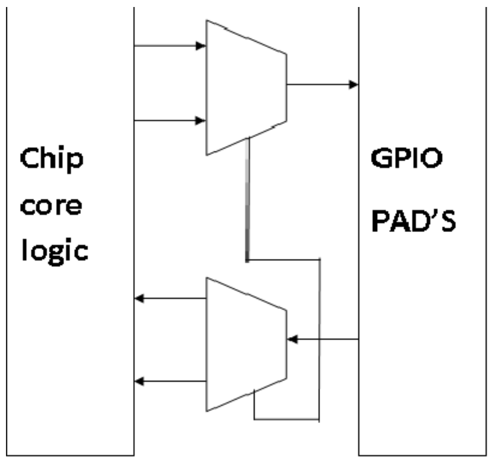


Figure 3.3: Mux Based Datapath Selection Logic

Chapter 4

FINITE STATE MACHINE

Finite state machines are mathematical abstraction used to design digital logic. It is a behavior model composed of finite number of states, transitions between those states thus indicating the logic flow. One hot encoding is used to to encode the states which ensures low switching activity (thus low power consumption and less prone to glitches) and also adding or deleting the states become easier. This technique also proves to be faster than other encoding logics. By using this technique critical paths of the design can be found easier.

4.1 SPECIFICATION

- Input : CLOCK,reset,acknowledge,ack

- Output: battery full,battery remain.

- Clock cycles : From the cycle, ack bit is high, module takes 8 clock cycles to complete one full traversal.

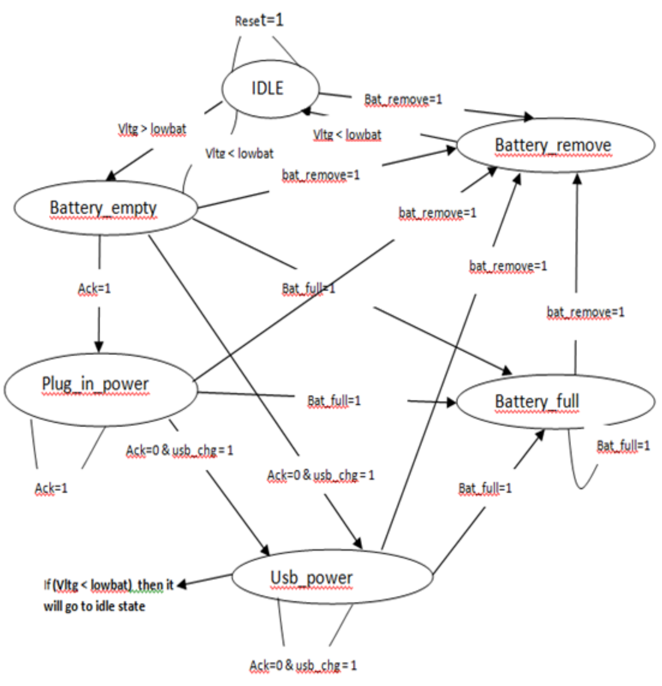


Fig 5 – STATE MACHINE FOR BATTERY CHARGING

Figure 4.1: STATE MACHINE FOR BATTERY CHARGER

4.2 STATE DIAGRAM FOR CHARGING THE BATTERY

4.3 WORKING MECHANISM

Number of STATE

- idle
- battery empty
- plug in power
- usb power
- battery complete
- battery remove

- **idle** : in this state machine we are using active low reset.When reset is low and system voltage is greater than low battery voltage then it will go to the next state called battery empty state.When battery removal signal = '1' then state moves to the battery removal state.

- **battery empty**:when system voltage is greater than low battery voltage it will remain in battery empty state.but when system voltage is less than low battery voltage it will again goes to the idle state.when battery removal and battery complete are high then it will goes to the respective battery removal and battery complete state. when ack is high it goes to the next state as plug in power state.

- **plug in power**:state will remain in plug in power till ack=1. when battery removal and battery complete are high then it will goes to the respective battery removal and battery complete state.when ack=0 and usb chg =1 then next state is usb power state.

- **usb power**:state will remain in usb power state till ack=0 and usb chg =1. when battery removal and battery complete are high then it will goes to the respective battery removal and battery complete state. When system voltage is less than low battery voltage it will again goes to the idle state.when bat full = 1 state will go the battery full state.

- **battery complete**:state will in same state till bat full =1.when bat removal =1 then it will go to the battery removal state.When system voltage is less than low battery voltage it will again goes to the idle state.

- **battery remove**:when we know that battery is charged fully we can remove the battery as battery is fully charged. When system voltage is less than low battery voltage it will again goes to the idle state.

4.4 CONCLUSION

This chapter deals with design of Finite State machine. The specification was translated into a state diagram and the module was designed to accomplish its required functionality.

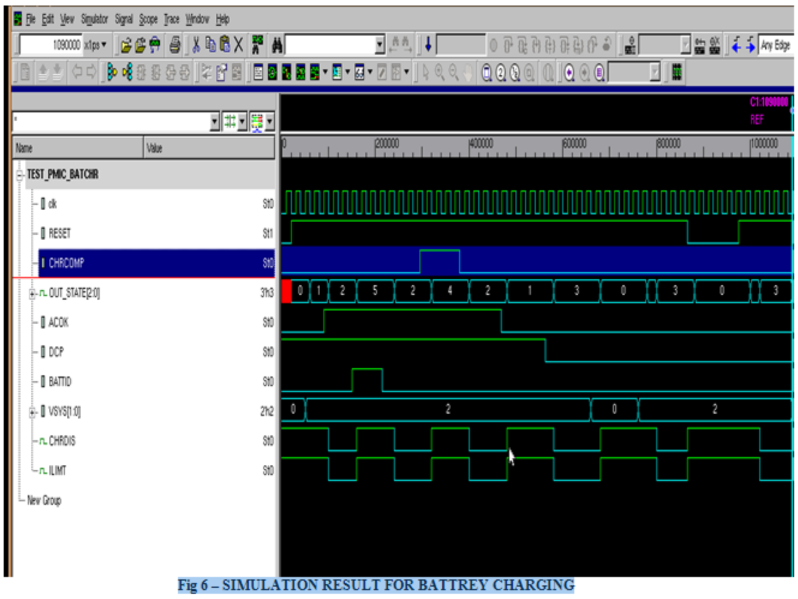


Fig 6 – SIMULATION RESULT FOR BATTREY CHARGING

Figure 4.2: Simulation Results

Chapter 5

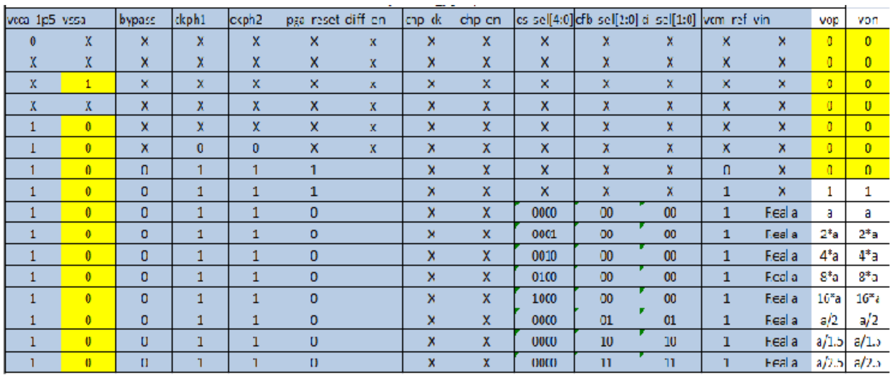
Behavioral model coding

5.1 Behavioral model coding

As shown earlier in the overall project block diagram the SOC also consists of an analog front end interface for the analog sensors. For the veri

cation of these analog blocks in the design behavioral models of the different circuits used were coded (in Verilog). The models were coded based on truth tables provided by the analog design team with real value inputs and outputs. Thus, BMODs were basically used to mimic the functionality of the analog circuits in the digital world. A total of 13 BMODs were coded for the analog design used in the project. As shown in the below screenshot the ouputs are analog in nature and thus the different values of vo from 0.6 -24 according to the appropriate stimuli applied.

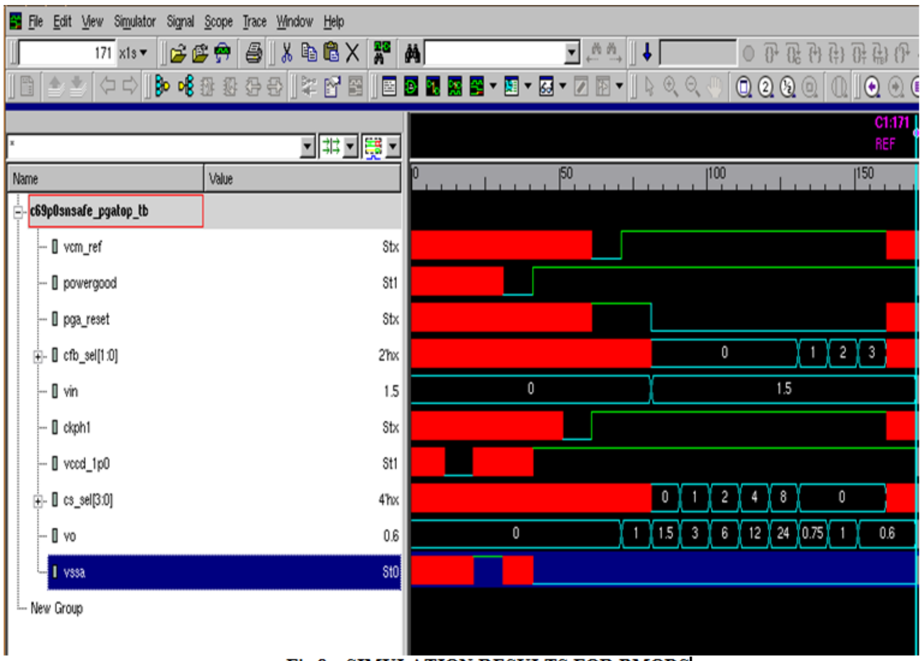
5.2 TRUTH TABLE



vcca 1p5	vssa	bypass	ckph1	ckph2	pga reset	diff en	chp clk	chp en	cs sel[4:0]	cfb sel[1:0]	ci sel[1:0]	vcm ref	vin	vop	von
0	X	X	X	X	X	x	X	X	X	X	X	X	X	0	0
X	X	X	X	X	X	x	X	X	X	X	X	X	X	0	0
X	1	X	X	X	X	x	X	X	X	X	X	X	X	0	0
X	X	X	X	X	X	x	X	X	X	X	X	X	X	0	0
1	0	X	X	X	X	x	X	X	X	X	X	X	X	0	0
1	0	X	0	0	X	x	X	X	X	X	X	X	X	0	0
1	0	0	1	1	1		X	X	X	X	X	0	X	0	0
1	0	0	1	1	1		X	X	X	X	X	1	X	1	1
1	0	0	1	1	0		X	X	0000	00	00	1	Real a	a	a
1	0	0	1	1	0		X	X	0001	00	00	1	Real a	2*a	2*a
1	0	0	1	1	0		X	X	0010	00	00	1	Real a	4*a	4*a
1	0	0	1	1	0		X	X	0100	00	00	1	Real a	8*a	8*a
1	0	0	1	1	0		X	X	1000	00	00	1	Real a	16*a	16*a
1	0	0	1	1	0		X	X	0000	01	01	1	Real a	a/2	a/2
1	0	0	1	1	0		X	X	0000	10	10	1	Real a	a/1.5	a/1.5
1	0	0	1	1	0		X	X	0000	11	11	1	Real a	a/2.5	a/2.5

Figure 5.1: Truth Table For BMMODS

5.3 SIMULATION RESULTS



Name	Value
c69p0snsafe_pgatop_tb	
vcm_ref	StX
powergood	St1
pga_reset	StX
cfb_sel[1:0]	2'hx
vin	1.5
ckph1	StX
vccd_1p0	St1
cs_sel[3:0]	4'hx
vo	0.6
vssa	St0
New Group	

Fig 8 – SIMULATION RESULTS FOR BMODS

Figure 5.2: Simulation Result

Chapter 6

RTL QUALITY CHECK(LINTRA TOOL)

Lintra is a tool for linting RTL code - that is, analyzing RTL code to verify that it conforms to design rules and coding guidelines. Lintra is built for RTL languages,providing an RTL Data Model with a C++/Perl application program interface (API), and a built-in light synthesis capability. Lira supports iHDL, Verilog and System Verilog.The warnings and errors issued during Lira's compilation flow form Lintra's 'built in rules'.

6.1 RUNNING LINTRA

To enable Lintra to be installed and run as a 'stand alone' tool, all libraries of required infrastructure tools (including Gandalf and Lira) are provided with Lintra release.Users that wish to run Dare rules (disabled by default) should also install Dare.To run Lintra, set up an environment variable LINTRA to point to your Lintra installation area. setenv LINTRA (Lintra installation area) By default, Lintra will load the UDRs from LINTRA/data. This behavior can be overridden by de

ning an additional environment variable, specifying an alternative directory for the UDRs: setenv LINTRARULES Lintra UDR area.

Chapter 7

LINTRA INPUTS AND OUTPUTS

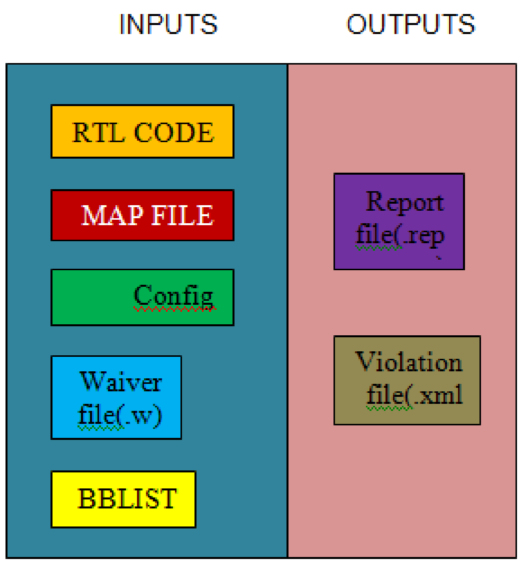


Fig 9 LINTRA INPUTS AND OUTPUTS

Figure 7.1: LINTRA I/O s

- **Lintra inputs** : RTL code configuration file(.f), waiver file(.w), map file(map) , bblist
- **Lintra outputs**: Report file(.rep) Violation file(.xml)

7.1 LINTRA INPUT AND OUTPUT DIAGRAM

7.2 LINTRA INPUT

- **RTL code** : Lintra can be run on any .v and .sv file.This the input for lintra as we perform linta on RTL code.for this file we will check the errors and warnings. Eg: (filename1).v, (filename2).v Below Input files have to be created by user in Lintra run directory.

- **Map file** : It contains commands for setting environmental variables which are utilized by the lintra script. For the SBL lintra script, the top module name and Lintra run directory are required. Eg: For running lintra on any module, following settings are required in Map file. setenv TOP MODULE module name setenv LINT DIR i.e.: path for lint directory.

- **BB list** : BB list is a black box list. Sometimes, we don't want to perform the lint on sub-modules with in a module. In such conditions, those sub module names should be kept in the bblist file. Eg: Since SPI is an external IP, we might not want to run Lintra on this module. Following are the few contents of a BB list used for Fabric plus. SPI I2C

- **Configuration file** :It should contain the paths of directories where Lintra can search for the modules. The mode of usage of BB list and the path for BB list also need to be mentioned here. There are different modes in which Lintra process the BB list. Full mode implies Lintra won't analyze the entire module and modules mode implies it will analyze the given module but wont analyze the sub-modules within the given module. In SBL2, full mode has been used. Lintra run script has the option '-f' to specify the config file. Eg: Below are few contents of config file used for fulladder- fulladder-cfg.f -bb full:bblist. -I ../../fulladder/rtl -I ../../halfadder/rtl -I ../../xor/rtl

- **Waiver file** :It can be used to mask the errors or warnings which are of low priority or which are invalid. Lintra supports two types of waiver mechanisms.

- waivers that are embedded in the RTL as comments.

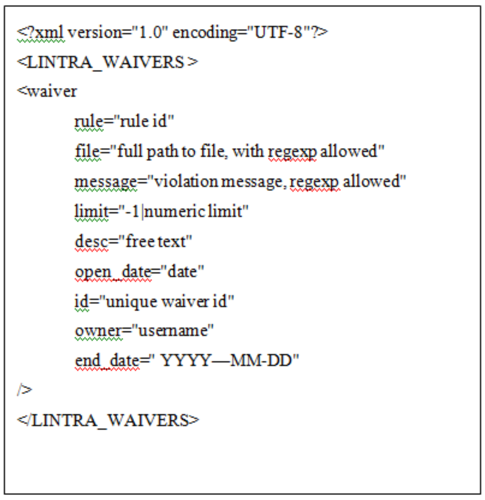
- waivers provided in a standalone waiver file.In SBL2 , stand alone waiver file has been used. Lintra run script has the script has the option '-wf' to specify the waiver file.A stand-alone waiver file can be specified during invocation using the command line option -waiver files, -wf file nam¿. The file name provided may contain wildcards to capture multiple files e.g. lintra.wave, but should not contain a path.A path can be specified with the switch waiver directory, wd.If the switch is not specified, waiver files will be loaded from current directory.The switch can be provided multiple times to load from multiple directories. To search all directories in Lira search path for waiver files, specify 'LIRA' instead of a valid path. Note in case of wavier files with identical names in different directories only the first file will be loaded, the rest will be ignored. The waivers can be created and/or edited from Arena GUI, or through manual editing of the waiver file, which is in XML format. We strongly recommend using Arena GUI for waiver editing, as it provides error checks to guard from waivers that do not capture the intended violation. Each waiver contains the following filter fields: rule id, message regexp, fileregexp, and limit. The waiver will filter out violations that match the given rule id, whose message matches the given message (whether as an exact string match or as a regular expression match) and whose location matches the given file expression. If any of the filters are left unspecified, messages will not be filtered by this attribute (e.g. if only the message is speci

  ed, the rule id and file will be ignored). In addition, if a number greater than zero is specified for the limit, the number of violations caught by the waiver will be thus limited. This is particularly useful when the violation message is the same for all violations of the rule, and offers a way to waive a particular violation, yet be warned when additional violations of this rule occur (in the same file). The regular expression semantics used by Lintra is QT regular expression, which is similar to PERL, where QT's quantifiers are the same as Perl's greedy quantifiers. See the appendix for details. Note - when using regular expressions,it may be necessary to escape characters in the original message text (such as brackets) by adding / before them. In addition to the filter, the waiver contains a description field, a unique id, an owner, opening date and expiration date. The description field is optional, and contains free text. The unique id, owner, and opening date fields are mandatory,and are automatically set to the userid , system date, and combination of user plus timestamp when the waiver is created through Lintra GUI. The expiration date can be used to create temporary waivers.

The format of the Waiver file is: Eg: A sample waiver file
xml version
LINTRA WAIVERS
! Waiving Low priority warnings
waiver rule="80028"
le=".*/*.v" message= ".*" desc="Low priority for now" limit="-1"
open date="" id="sbl2-lint-80028" owner="govil"/
! END
/LINTRA WAIVERS

7.3 LINTRA OUTPUT

- **Report file** It reports all the errors and warnings.It contains all the violation which are occurred in (.v) and (.sv) file.it gives all the details of the file like path of the file ,on which line error occurred ,severity of the violation ,which lintra rule it follows and a message of the error. As shown in example below with lintra rule(0531) the warning(severity) for line no. 418 is bits 'carry 0' of



```
<?xml version="1.0" encoding="UTF-8"?>
<LINTRA_WAIVERS >
<waiver
        rule="rule id"
        file="full path to file, with regexp allowed"
        message="violation message, regexp allowed"
        limit="-1|numeric limit"
        desc="free text"
        open_date="date"
        id="unique waiver id"
        owner="username"
        end_date=" YYYY—MM-DD"
/>
</LINTRA_WAIVERS>
```

Fig 10 MODEL WAIVER FILE FOR LINTRA

Figure 7.2: waivers

an internal signal are never assigned. Eg: Contents of report file - full adder.v. Line Sever. Rule Message

==

418 Warning 0531 bits 'carry 0' of an internal signal are never assigned. 696 Error 60044 Continuous assignment in declaration of net 'half add op' is not allowed. Line Sever. Rule Message

==

193 Warning 0527 bits 'ortcosc32s is i' of an input signal are never used. 194 Warning 2058 bits 'ortcosc32s is o' of output signal are never assigned. Line Sever. Rule Message

==

172 Warning 0527 bits 'apbresetl' of an input signal are never used. 250 Warning 2058 bits 'gpio ctrl[18], gpio ctrl[20], gpio ctrl[22], gpio ctrl[24], gpio ctrl[26], gpio ctrl[38], gpio ctrl[42], gpio ctrl[58]' of output signal are never assigned. 425 warning v2 0529 bits 'bit0 buf7' of an internal signal are used but unas- signed.

- **violation file(.xml)**-The violation file uses con

gurations file that allows customizing Lintra runs to the project requirements. It provides control over which severity levels will cause Lintra to terminate with a 'failed' exit status, and provides capabilities for determining which rules are active and tailoring the rules by changing the value of con

gurable parameters that are coded into the rules. The configuration file is in XML format, as specified below. Configuring the severity levels : The configuration file can be used to define severity levels, specifying two attributes:

- Specify priorities for the severity levels. These will be used when sorting violation reports by the violation severity.

- Specify if violations of a severity level are allowed. This is used to determine if Lintra should exit with a non-zero status - i.e. if during lint violations with a non-allowed level are found, the exit status at the end of the run will be non-zero.

7.4 LINTRA EXIT STATUS

Lintra will terminate with one of the following -
One - If errors detected in the command line or when loading the configuration file or the UDRs prevented the lint run, Lintra will terminate with an exit status of 1
Two - If during lint violations of a severity declared as 'cause failure' in the configuration file were reported, Lintra will report 'lint Failed' and terminate with an exist status of 2.

- **Types of Violation in Lintra :**

- **FATAL**:If fatal violation comes then the lintra will get terminiated.so,to remove the fatal error is the first priority of Lintra.

- **ERROR**:depending on the type of error it can be fixed or waived.Waiver file is explain below.

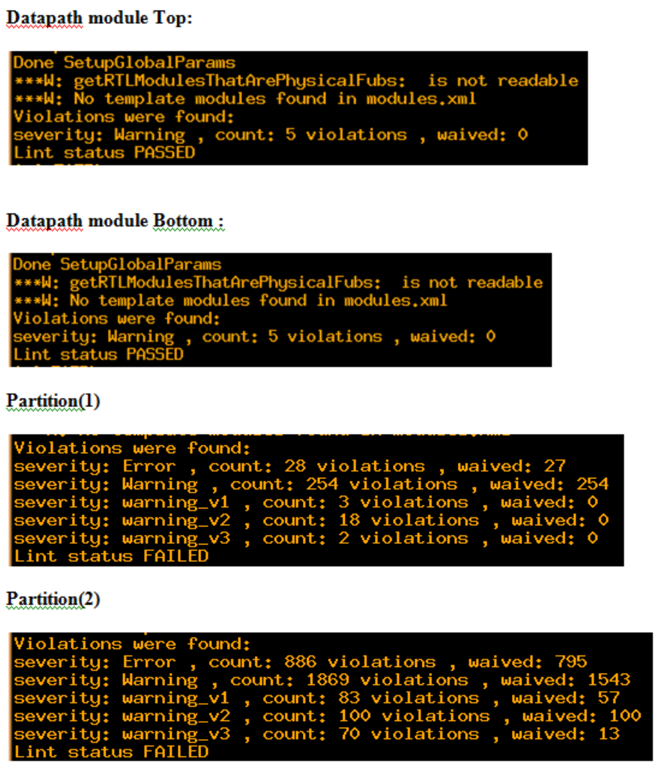
- **WARNING**:It is decided by the designer whether he wants to fix the warning or to waive it.

- **WARNING V1,V2,V3**:these are low priority warning it will not affect a lot to rtl design.

Chapter 8

LINTRA RESULTS

Following are the lintra results for the corresponding module.As per the following results many of the errors and the warning are waived as they does not affect the quality of RTL design and these errors will not affect in synthesis of the design.



Datapath module Top:

```
Done SetupGlobalParams
***W: getRTLModulesThatArePhysicalFubs:  is not readable
***W: No template modules found in modules.xml
Violations were found:
severity: Warning , count: 5 violations , waived: 0
Lint status PASSED
```

Datapath module Bottom :

```
Done SetupGlobalParams
***W: getRTLModulesThatArePhysicalFubs:  is not readable
***W: No template modules found in modules.xml
Violations were found:
severity: Warning , count: 5 violations , waived: 0
Lint status PASSED
```

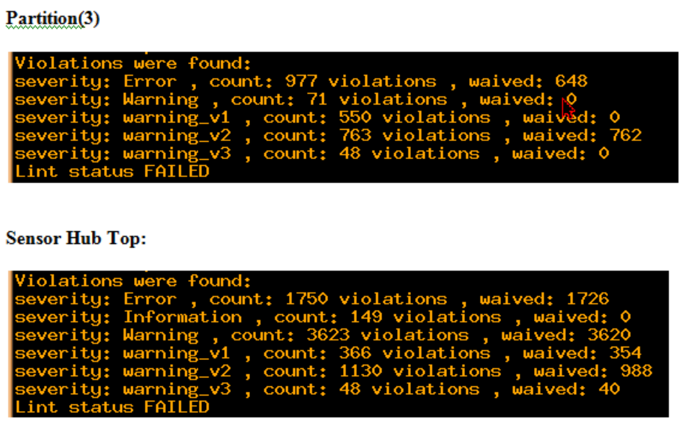
Partition(1)

```
Violations were found:
severity: Error , count: 28 violations , waived: 27
severity: Warning , count: 254 violations , waived: 254
severity: warning_v1 , count: 3 violations , waived: 0
severity: warning_v2 , count: 18 violations , waived: 0
severity: warning_v3 , count: 2 violations , waived: 0
Lint status FAILED
```

Partition(2)

```
Violations were found:
severity: Error , count: 886 violations , waived: 795
severity: Warning , count: 1869 violations , waived: 1543
severity: warning_v1 , count: 83 violations , waived: 57
severity: warning_v2 , count: 100 violations , waived: 100
severity: warning_v3 , count: 70 violations , waived: 13
Lint status FAILED
```

Figure 8.1: Lintra Results



Partition(3)

```
Violations were found:
severity: Error , count: 977 violations , waived: 648
severity: Warning , count: 71 violations , waived: 0
severity: warning_v1 , count: 550 violations , waived: 0
severity: warning_v2 , count: 763 violations , waived: 762
severity: warning_v3 , count: 48 violations , waived: 0
Lint status FAILED
```

Sensor Hub Top:

```
Violations were found:
severity: Error , count: 1750 violations , waived: 1726
severity: Information , count: 149 violations , waived: 0
severity: Warning , count: 3623 violations , waived: 3620
severity: warning_v1 , count: 366 violations , waived: 354
severity: warning_v2 , count: 1130 violations , waived: 988
severity: warning_v3 , count: 48 violations , waived: 40
Lint status FAILED
```

Figure 8.2: Lintra Results

Chapter 9

Increasing Challenges Of Power And Low Power Design Strategies

9.1 Increasing Challenges Of Power

In earlier generations of IC design technologies, the main parameters of concern were timing and area. EDA tools were designed to maximize the speed while minimizing area. Power consumption was a lesser concern. CMOS was considered a low-power technology, with fairly low power consumption at the relatively low clock frequencies used at the time, and with negligible leakage current. In recent years, however, device densities and clock frequencies have increased dramatically in CMOS devices, thereby increasing the power consumption dramatically. At the same time, supply voltages and transistor threshold voltages have been lowered, causing leakage current to become a significant problem. As a result, power consumption levels have reached their acceptable limits, and power has become as important as timing or area. High power consumption can result in excessively high temperatures during operation. This means that complex and expensive heat sinks and cooling systems are often required for product operation. Laptop computers and hand-held electronic devices can become uncomfortably hot to the touch. Higher operating temperatures also reduce reliability because of electromigration and other heat-related failure mechanisms. High power consumption also reduces battery life in portable devices such as laptop computers, cell phones, and personal electronics. As more features are added to a product, power consumption increases and the battery life is reduced, requiring a larger, heavier battery or shorter life between charges. Battery technology has lagged behind the increased demands for power.

9.2 Static And Dynamic Power

Designers must consider two types of power consumption, dynamic and static. Dynamic power is consumed during the switching of transistors, so it depends on the clock frequency and switching activity. Static power is the transistor leakage current that flows whenever power is applied to the device, so it is not related to the clock frequency or switching activity.

9.2.1 Dynamic Power

Dynamic power is the energy consumed during logic transitions on nets, consisting of two components, switching power, short circuit power and glitching power. Switching power results from the charging and discharging of the external capacitive load on the output of a cell.
P=cV2f

P - power dissipated, c - capacitance, V- Applied voltage, F - frequency. Short Circuit Power takes place when input changes, power Dissipation takes place even when ,no load or parasitic capacitance.Glitching Power is mismatch or Imbalance in path length.

9.2.2 Static Power or Leakage Power

Leakage current was negligible in earlier CMOS technologies. However, with shrinking device geometries and reduced threshold voltages, leakage power is becoming increasingly significant, sometimes approaching the levels of dynamic power dissipation. Current drawn from the supply due to input voltage.

9.3 Low Power Design Techniques

There are several different RTL and gate-level design strategies for reducing power. Some methods, such as clock gating, have been used widely and successfully for many years. Others, such as dynamic voltage and frequency scaling,power switching and multithreshold voltage are the other power reduction techniques.As power becomes increasingly important in advanced technologies, more methods are being exploited to achieve the design requirements.

9.3.1 Scaling

Scaling is an important factor that affects power consumption.As technology is scaled,supply voltage also be scaled so power consumption also reduced. The most basic way to reduce power is to reduce the supply voltage. Power usage is proportional to the square of the supply voltage. If supply voltage is scaled by 1/k the power consumption is scaled by a factor of 1/k.Reduction in supply voltage results in increase in circuit delay. Td = CL *Vdd / u Cox(W/L)(Vdd -Vt) Td - circuit delay, CL - load capacitance, Vdd - supply voltage, Cox - oxide capacitance, Vt - threshold voltage.

9.3.2 Clock Gating

Clock gating is a dynamic power reduction method in which the clock signals are stopped for selected register banks during times when the stored logic values are not changing. Most of the power is dissipated during transition.Gating clock is the most effective way to reduce the transition. Place gated clock at high level.Clock gating is a dynamic power reduction method in which the clock signals are stopped for selected register banks during times when the stored logic values are not changing.One possible implementation of clock gating is shown in Figure Clock gating is particularly useful

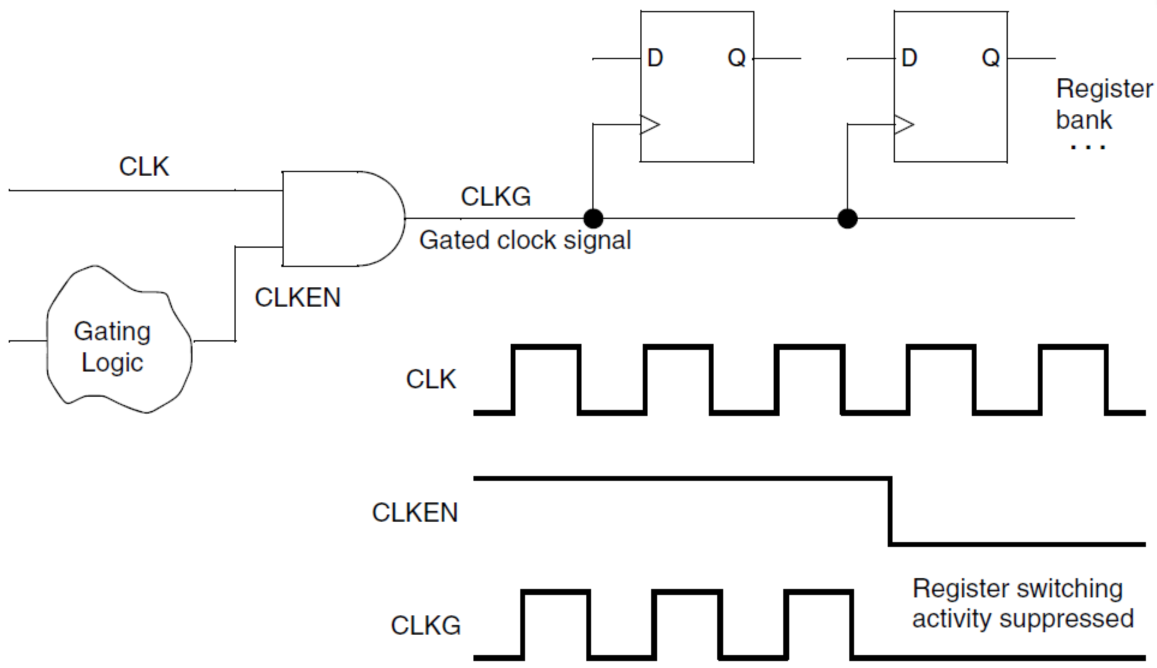


Figure 9.1: clock gating

for registers that need to maintain the same logic values over many clock cycles. Shutting off the clocks eliminates unnecessary switching activity that would otherwise occur to reload the registers on each clock cycle. The main challenges of clock gating are finding the best places to use it and creating the logic to shut off and turn on the clock at the proper times.

9.3.3 Multi-Voltage Design

Different parts of a chip might have different speed requirements.In multi voltage designs providing power to the different power domains is challenging. Every power domain requires independent local power supply and grid structure. For example, block 1and block 2 might need to be faster than a peripheral block. As mentioned earlier, a lower supply voltage reduces power consumption but also reduces speed. To get maximum speed and lower power at the same time, block 1and block 2 can operate with a higher supply voltage while the peripheral block operates with a lower voltage, as shown in Figure below Providing two or more supply voltages on a single

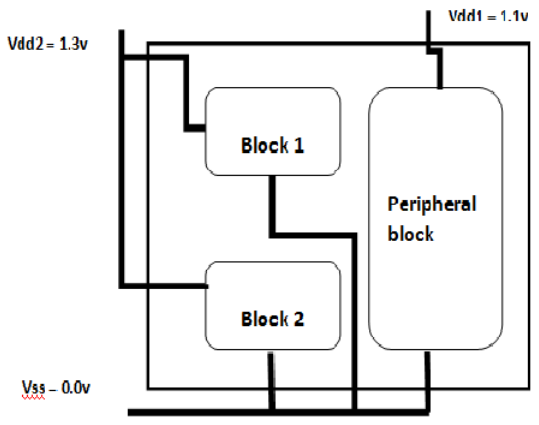


Figure 9.2: Multi-Voltage Design

chip introduces some complexities and costs. Additional device pins must be available to supply the chip voltages, and the power grid must distribute each of the voltage supplies separately to the appropriate blocks. Where a logic signal leaves one power domain and enters another, if the voltages are significantly different, a level-shifter cell is necessary to generate a signal with the proper voltage swing. In the example shown in Figure , a level shifter converts a signal with 1.3-volt swing to a signal with a 1.1-volt swing. A level shifter cell itself requires two power supplies that match the input and output supply voltages.

9.3.4 Power Switching

Power switching is a power-saving technique in which portions of the chip are shut down completely during periods of inactivity. For example, in a cell phone chip, the block that performs voice processing can be shut down when the phone is in standby

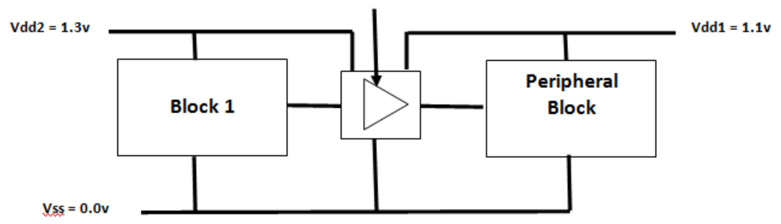


Figure 9.3: Level Shifter In Multi-Voltage Design

mode. When the user places a call or receives an outside call, the voice processing block must "wake up"from its powered-down state. Power switching has the potential to reduce overall power consumption substantially because it lowers leakage power as well as switching power. It also introduces some additional challenges, including the need for a power controller, a power-switching network, isolation cells, and retention registers. A power controller is a logic block that determines when to power down and power up a specific block.There is a certain amount of time and power cost for powering down and powering up a block, so the controller should determine the appropriate power-down times with a high degree of certainty. A block that can be powered down must receive its power through a power-switching network, consisting of a larger number of transistors with source-to-drain connections between the always-on power supply rail and the power pins of the cells. The power switches are distributed physically around or within the block. The network, when switched on, connects the power to the logic gates in the block. When switched off, the power supply is effectively disconnected from the logic gates in the block. Any use of power switching requires isolation cells where signals leave a powered-down block and enter a block that is always on (or currently powered up). An isolation cell provides a known, constant logic value to an always-on block when the power-down block has no power, thereby preventing unknown or intermediate values that could cause crowbar currents. One simple implementation of an isolation cell is shown in Figure , When the block on the left is powered up, the signal P UP is high and the output signal passes through the isolation cell unchanged (except for a gate delay). When the block on the left is powered down, P UP is low, holding the signal constant at logic 0 going into the always-on block. Other types of isolation cells can hold a logic 1 rather than 0, or can hold the signal value latched at the time of the power-down event. Isolation cells must themselves have power during block power-down periods. The power switching

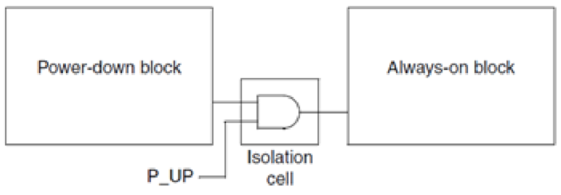


Figure 9.4: Isolation Cell

can be combined with multivoltage operation. Different blocks can be designed to operate at different voltages and also to be separately powered down when they are not needed. In that case, the interface cells between different blocks must perform both level shifting and isolation functions, depending on whether the two blocks are operating at different voltages or one is shut down. A cell that performs both functions is called an enable level shifter. This cell must have two separate power supplies, just like any other level shifter.

Chapter 10

Unified Power Format For Low Power Design

10.1 Introduction

UPF (Unified power format) is a standard format from Synopsys which is commonly used to implement low power designs.The IEEE 1801 Standard for Design and Verification of Low Power Integrated Circuits, also known as the Unified Power Format (UPF), provides a consistent way to specify power implementation intent throughout the design process, including synthesis, physical implementation, and verification. This consistency makes it easier to perform simulation, logical equivalence checking, and design verification in the presence of specific low-power features in a given design.Using UPF commands, you can specify the supply network, switches, isolation, retention, and other aspects relevant to power management of a chip design.

10.2 Power Intent Concepts

The UPF language provides a way to specify the power requirements of a design, but without specifying explicitly how those requirements are implemented. The language specifies how to create a power supply network to each design element, the behavior of supply nets with respect to each other, and how the logic functionality is extended to support dynamic power switching to design elements. It does not contain any placement or routing information. The UPF specification is separate from the RTL description of the design. In the UPF language, a power domain is a group of elements in the design that share a common set of power supply needs. By default, all logic elements in a power domain use the same primary supply and primary ground. Other power supplies may optionally be defined for a power domain as well. A power domain is typically implemented as a contiguous voltage area in the physical chip layout, although this is not a requirement of the language. Each power domain has a scope. The scope is the level of logic hierarchy designated as the root of the domain.The scope is the hierarchical level at which the domain is defined and is an ancestor of the elements belonging to the power domain. Each scope in the design has supply nets and supply ports at the defined hierarchical level of the scope. A supply net is a conductor that carries a supply voltage or ground throughout a given power domain. A supply net that spans more than one power domain is said to be "reused" in multiple domains. A supply port is a power supply connection point between two adjacent levels of the design hierarchy, between the parent and child blocks of the hierarchy. A supply net that crosses from one level of the design hierarchy to the next passes through a supply port. A power switch (or simply switch) is a device that turns on and turns off power for a supply net. A switch has an input supply net, an output supply net that can be switched on or off, and at least one input signal to control switching. The switch can optionally have multiple input control signals and one or more output acknowledge signals. A power state table lists the allowed combinations of voltage values and states of the power switches for all power domains in the design. A level shifter must be present where a logic signal leaves one power domain and enters another at a substantially different supply voltage. The level shifter converts a signal from the voltage swing of the first domain to that of the second domain. An isolation cell must be present where a logic signal leaves a switchable power domain and enters a different power domain. The level shifter generates a known logic value during shutdown of the domain. If the voltage levels of the two domains are substantially different, the interface cell must be able to perform both level shifting (when the domain is powered up) and isolation (when the domain is powered down). A cell that can perform both functions is called an enable level shifter. In a power domain that has power switching, any registers that must retain data during shutdown must be implemented as retention registers. A retention register has a separate, always-on supply net, sometimes called the backup supply, which keeps the data stable in in the retention register while the primary supply of the domain is shut down. The power network example shown in Figure demonstrates some of the power intent concepts. This chip is designed to operate with three power supplies that are always on (although the UPF syntax also supports externally switchable power supplies), at three different voltage levels. The top-level chip occupies the top-level power domain, PD TOP. The domain PD TOP is defined to have four supply ports: VDD1, VDD2, VDD3, and GND. The black squares along the border of the power domain represent the supply ports of that domain. In addition

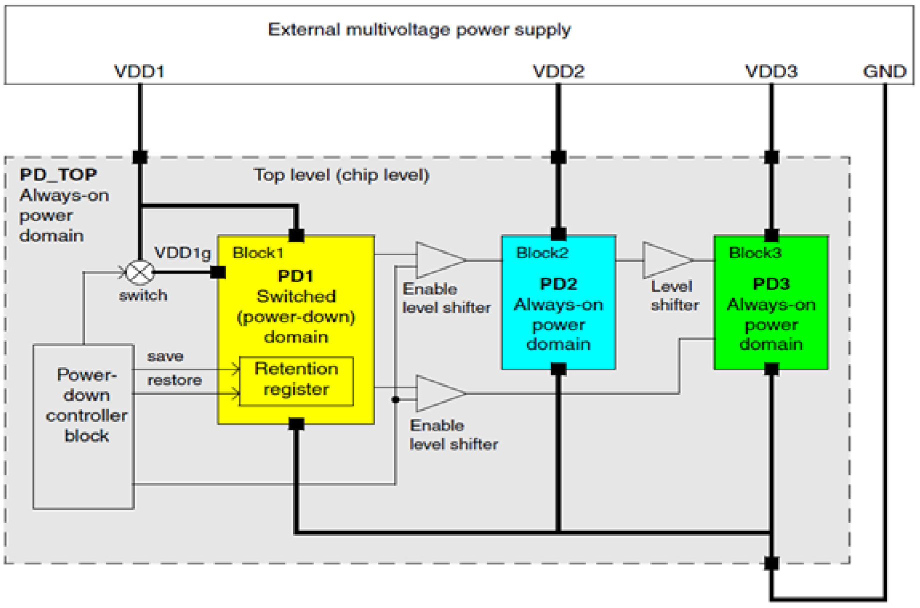


Figure 10.1: Power Intent Diagram

to the top-level power domain, PD TOP, there are three more power domains defined, called PD1, PD2, and PD3, created at the levels of three hierarchical blocks, Block1, Block2, and Block3, respectively. Each block has supply ports (shown as black squares in the diagram) to allow supply nets to cross from the top level down into the block level. In this example, PD TOP, PD2, and PD3 are always-on power domains that operate at different supply voltages, VDD1, VDD2, and VDD3, respectively. PD1 is a power domain that has two supplies: a switchable supply called VDD1g and an always-on supply from VDD1. The always-on power supply maintains the domain's retention registers while VDD1g is powered down. A power switch shuts off and turns on the power net VDD1g, either by connecting or disconnecting VDD1 and VDD1g. A power-down controller logic block at the top level generates the control signal for the switch. It also generates the save and restore signals for the retention registers in domain PD1 and the control signals for the isolation cells between domain PD1 and the always-on domains PD2 and PD3. These isolation cells generate known signals during times that VDD1g is powered down. Because domains PD1, PD2, and PD3 operate at different supply voltages, a level shifter must be present where a signal leaves one of these domains and enters another. In the case of the signals leaving PD1 and entering PD2 or PD3, the interface cells must be able to perform both level shifting and isolation functions, because PD1 can be powered down.

10.3 Synopsys Low Power Flow

The Synopsys low-power synthesis, implementation, and verification flow is shown in Figure. The flow starts with the register-transfer level (RTL) description of the logic of the design, together with a separate UPF description of the power intent of the design. The RTL and UPF descriptions are contained in separate files so that they can be maintained and modified separately. The initial UPF description is designated the UPF0 file in this example. Design Compiler reads in the RTL logic and original UPF power intent descriptions, and based on their contents, synthesizes a gate-level netlist and an updated UPF file, designated UPF' (UPF prime) in this

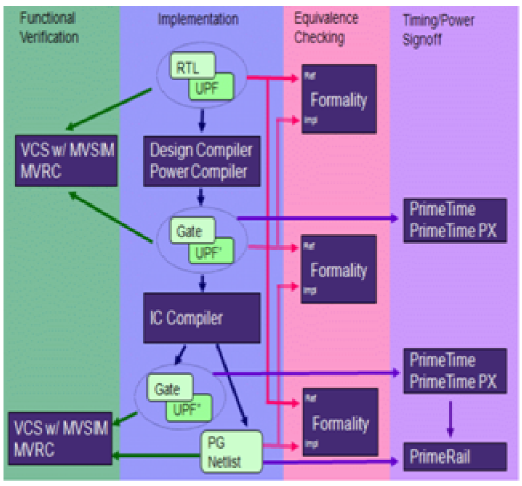


Figure 10.2: SYNOPSYS LOW POWER FLOW

example. The UPF' file contains the original UPF information plus explicit supply net connections for special cells created during synthesis. IC Compiler reads in the gate-level netlist and UPF' power description files, and based on the file contents, performs physical implementation (placement and routing), producing a modified gate-level netlist, a complete power and ground (PG) netlist, and 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 a power switches. The data files used in this flow can be used for functional verification with the VCS simulator, formal equivalence checking with Formality, and timing and power verification with PrimeTime, PrimeTime PX, and PrimeRail. The VCS simulator and MVSIM multivoltage simulation tool can be used for functional verification of the design with multivoltage features at several different stages of the flow: at the RTL level before synthesis, at the gate level after synthesis with power-related cells added, and after placement and routing with the power switches added. At each level, VCS and MVSIM co-simulate the design to verify the impact of voltage changes in a power-managed chip, allowing you to accurately and reliably detect any low-power design issues. MVRC checks for adherence to multivoltage rules and reports any problems related to power connectivity, power architecture, or power intent consistency. PrimeTime reads the gate-level netlist from Design Compiler or IC Compiler and also reads the UPF descriptions generated by those tools. It uses the UPF information to build a virtual model of the power network and to annotate voltage values appropriately on each power pin of each leaf-level gate instance in the design. PrimeTime does not modify the power domain description in any structural or functional way, so it does not write out any UPF commands.

10.4 UPF Requirements

In order to use power-saving strategies such as clock gating, multivoltage, multiple-Vt library cells, or power switching, following are the logic cells that support these strategies. Some of the types of cells that support low-voltage design are described in the following sections:

- Clock-Gating Cells
- Level Shifters
- Isolation Cells
- Power-Switch Cells
- Always-On Logic Cells
- Retention Register Cells

Clock-Gating Cells

10.4.1 Clock-Gating Cells

Synthesis tools such as Power Compiler can determine where clock gating can be used to provide the greatest power-saving benefit, and can automatically insert clock-gating circuits into the design to implement the clock-gating functions. Inserting clock-gating circuitry into anexisting clock network can introduce skew that adversely affects timing. A clock-gating cell can incorporate any kind of logic such as multiple

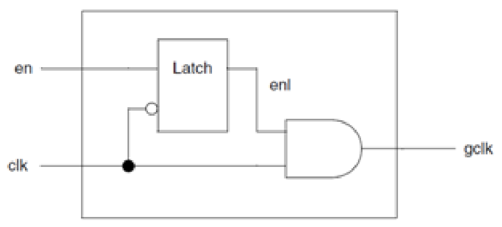


Figure 10.3: Clock-Gating Cells

enable inputs, test clock input, global scan input, asynchronous reset latch, active-low enabling logic, or inverted gated clock output. Power Compiler is free to optimize the enabling logic surrounding a clock-gating cell by absorbing the surrounding logic into the logic functions available inside the cell.

10.4.2 Level Shifters

In a multivoltage design, a level shifter is required where each signal crosses from one power domain to another. The level shifter operates as a buffer with one supply voltage at the input and a different supply voltage at the output. Thus, a level shifter converts a logic signal from one voltage swing to another, with a goal of having the smallest possible delay from input to output. The description of a level-shifter cell

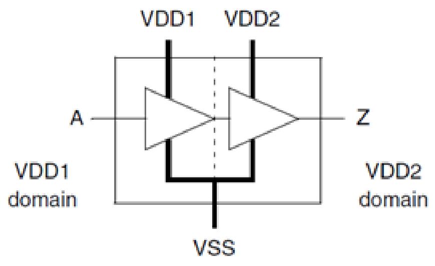


Figure 10.4: Level Shifter

must have information about the type of conversion performed (high-to-low, low-to-high, or both), the supported voltage levels, and the identities of the respective power pins that must be connected to each power supply.

10.4.3 Isolation Cells

In a design with power switching, an isolation cell is required where each logic signal crosses from a power domain that can be powered down to a domain that is not powered down. The cell operates as a buffer when the input and output sides of the cell are both powered up, but provides a constant output signal during times that the input side is powered down. An enable input controls the operating mode of the cell. A cell that can perform both level-shifting and isolation functions is called an enable level-shifter cell. This type of cell is used where a signal crosses from one power domain to another, where the two voltage levels are different and the first domain can be powered down.

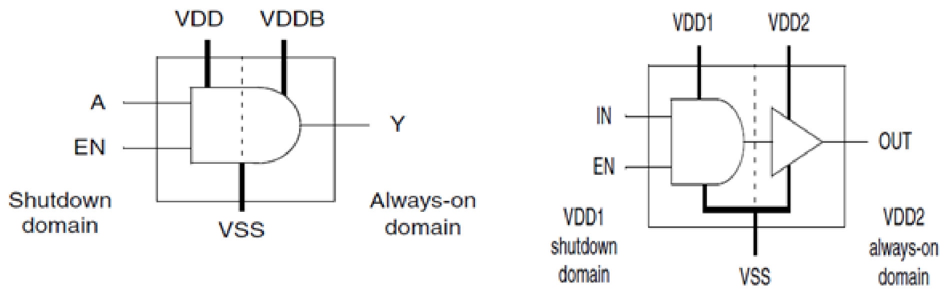


Figure 10.5: Isolation And Enable Shift Register

10.4.4 Power-Switch Cells

In a design with power switching, either header or footer type power switch cells are required to supply power for cells that can be powered down. A header type power switch connects the power rail to the power supply pins of the cells in the power-down domain. A footer type power switch connects the ground rail to the ground supply pins of the cells in the powerdown domain. An input logic signal to the power switch controls the connection or disconnection state of the switch.

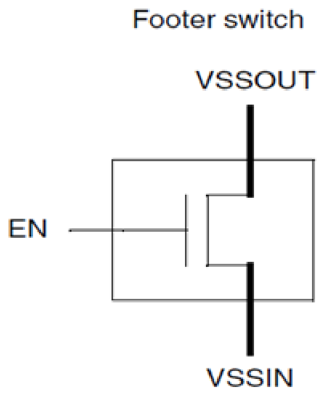


Figure 10.6: Footer Switch

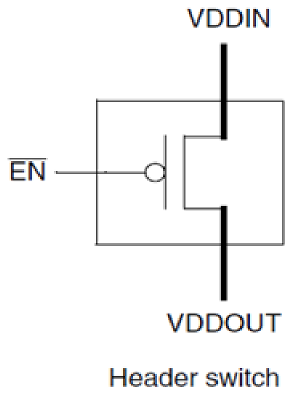


Figure 10.7: Header Switch

10.4.5 Always-On Logic Cells

When dealing with shutdown domains, there can be some situations in which certain cells in the shutdown portion need to continuously stay active, such as for implementing retention registers, isolation cells, retention control paths, and isolation enable paths. For example, if a save signal or restore signal passing through a shutdown voltage area needs buffering, an always-on buffer cell must be used. This type of logic is called always-on logic, which is built with always-on library cells. Compared to an ordinary cell, a functionally equivalent always-on cell has a backup power supply that operates continuously, even during the shutdown mode.

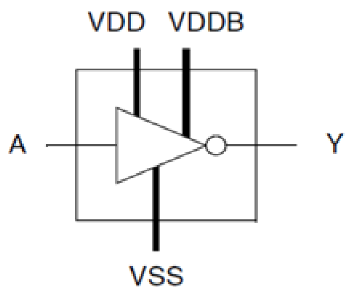


Figure 10.8: Always ON

10.4.6 Retention Register Cells

In a design with power switching, there are several different ways to save register states before power-down and restore them upon power-up in the power-down domain. One method is to use retention registers, which are registers that can maintain their state during power-down by means of a low-leakage register network and an always-on power supply.

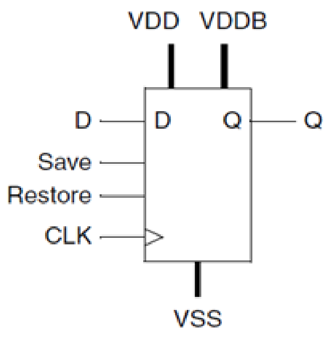


Figure 10.9: Retention Register

Chapter 11

Unified Power Format (UPF) For Real Time Project

UPF is designed to reflect the power intent of a design at a relatively high level. UPF scripts describe which power rails should be routed to individual blocks, when blocks are expected to be powered up or shut down, how voltage levels should be shifted as signals cross from one power domain to another and whether measures should be taken to retain register and memory-cell contents if the primary power supply to a domain is removed.When the blocks are power down how to apply the isolation ?All these low power related issue can be solved using UPF. In this project we are not using retention register and as operating voltage is 1.05v throuout the design so no need of level shifter in this design.The chip has two power domain,subsystem domain and GPIO domain.Both the power domain share the same supply voltage and ground. GPIO is Always ON block while the subsystem block is a power down block.SO we are using the power switch to switch off the subsystem power domain.Similarly when we power down the subsystem block some of the input signal which are coming into the GPIO, we have to isolate those inputs.So whichever input coming from subsystem to GPIO we are using firewall(isolation) for these inputs. Power controller is created in testbench to control the power switching and isolation.Power controller on and off the power switch using pwr_ctrl_sig.When pwr_ctrl_sig is '1',both the block are 'ON'.Subsystem block goes into the power down mode when pwr_ctrl_sig is '0'. Power intent diagram for the chip top is shown below.It contains two power domain vcc_subsys_upf_top and vcc_gpio_upf_top.For vcc_gpio_upf_top we are using the isolation cell. Power controller block is used for power switching and isolation.It control the on and off of vcc_subsys_upf_top and isolation for vcc_gpio_top. There are three supply port in a top chip level vcc_subsys_upf, vcc_gpio_upf and vss_upf. vcc_subsys_upf is the supply port for vcc_subsys_upf_top.vcc_subsys_upf_top also have switchable port as vss1_upf which is the output port for vcc_subsys_upf_top.vcc_gpio_upf is supply port for vcc_gpio_upf_top.vss_upf is the ground which is common for both the power domain. A power switch shuts off and turns on the power net vss1_upf, either by connecting or disconnecting vcc_subsys_upf and vss1_upf. A power-down controller logic block at the top level generates the control signal for the switch. It also generates the isolation control signal for vcc_gpio_upf_top.

11.1 Power Intent Diagram

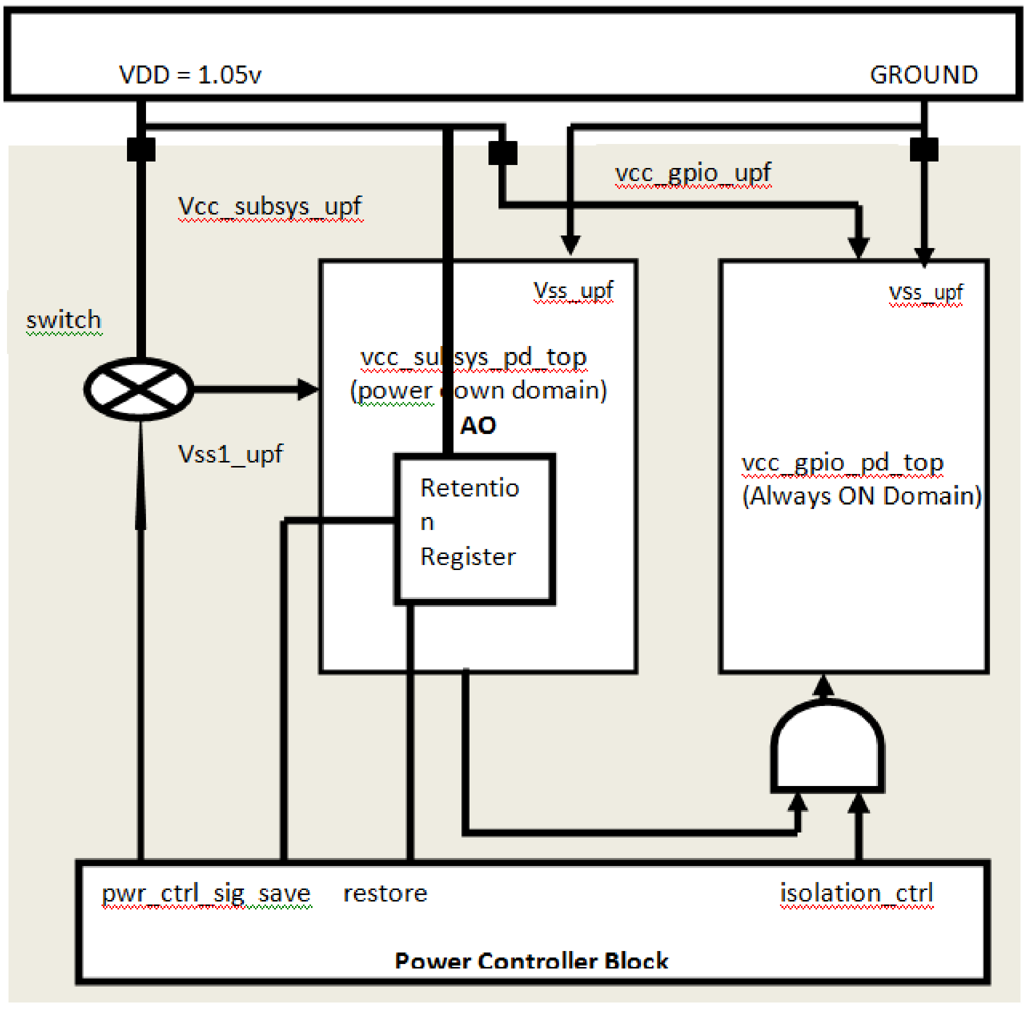
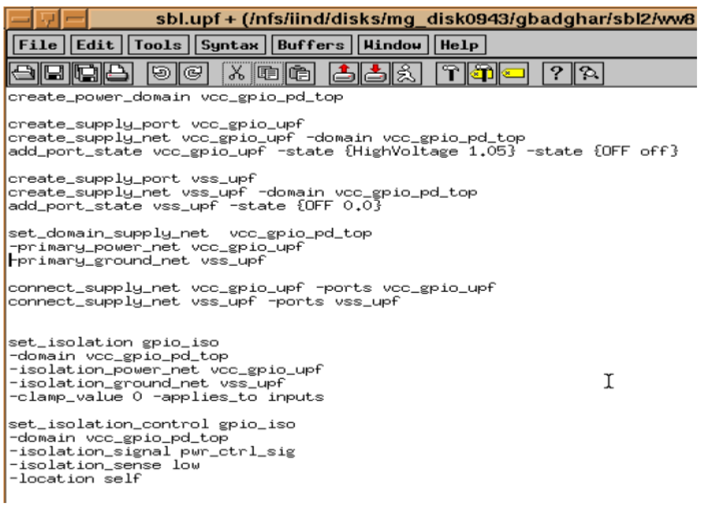


Figure 11.1: Power Intent Diagram

11.2 Example of UPF

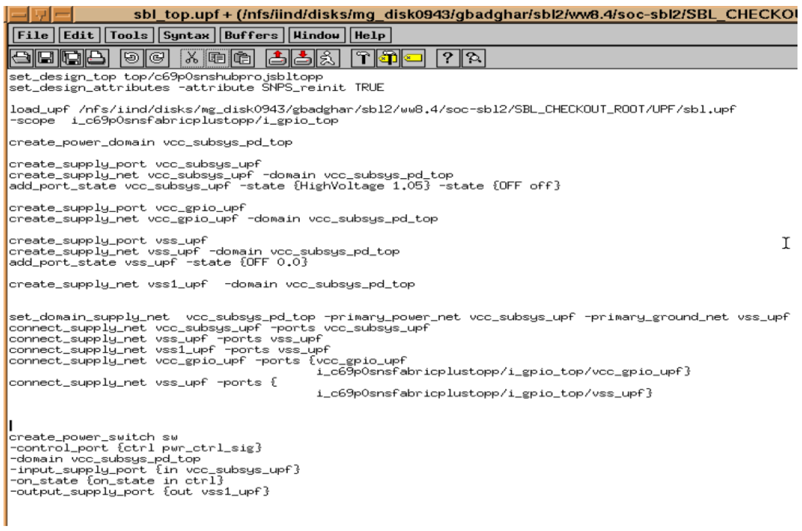
- UPF for GPIO Power Domain


sbl.upf + (/nfs/iind/disks/mg_disk0943/gbadghar/sbl2/ww8
File Edit Tools Syntax Buffers Window Help

```
create_power_domain vcc_gpio_pd_top

create_supply_port vcc_gpio_upf
create_supply_net vcc_gpio_upf -domain vcc_gpio_pd_top
add_port_state vcc_gpio_upf -state {HighVoltage 1.05} -state {OFF off}

create_supply_port vss_upf
create_supply_net vss_upf -domain vcc_gpio_pd_top
add_port_state vss_upf -state {OFF 0.0}

set_domain_supply_net  vcc_gpio_pd_top
-primary_power_net vcc_gpio_upf
-primary_ground_net vss_upf

connect_supply_net vcc_gpio_upf -ports vcc_gpio_upf
connect_supply_net vss_upf -ports vss_upf


set_isolation gpio_iso
-domain vcc_gpio_pd_top
-isolation_power_net vcc_gpio_upf
-isolation_ground_net vss_upf
-clamp_value 0 -applies_to inputs

set_isolation_control gpio_iso
-domain vcc_gpio_pd_top
-isolation_signal pwr_ctrl_sig
-isolation_sense low
-location self
```

Figure 11.2: UPF for GPIO Power Domain

- UPF for subsystem Power Domain

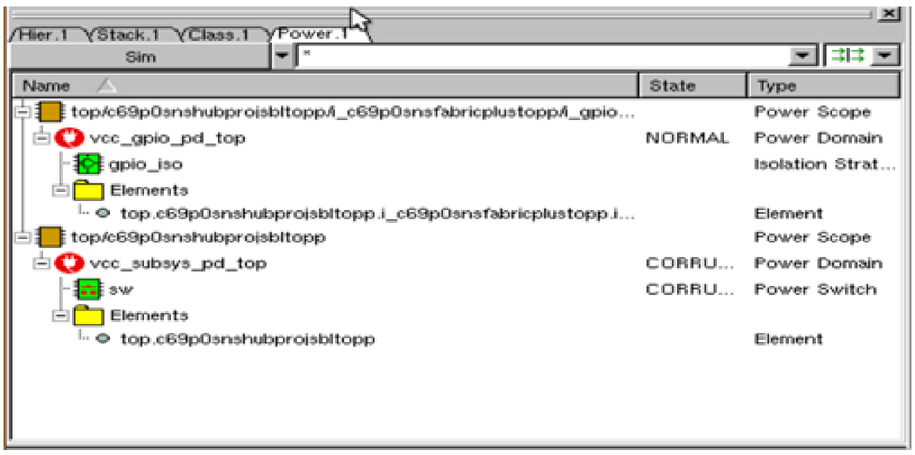

sbl_top.upf + (/nfs/iind/disks/mg_disk0943/gbadghar/sbl2/ww8.4/soc-sbl2/SBL_CHECKO
File Edit Tools Syntax Buffers Window Help

```
set_design_top top/c69p0snshubprojsbltopp
set_design_attributes -attribute SNPS_reinit TRUE

load_upf /nfs/iind/disks/mg_disk0943/gbadghar/sbl2/ww8.4/soc-sbl2/SBL_CHECKOUT_ROOT/UPF/sbl.upf
-scope  i_c69p0snsfabricplustopp/i_gpio_top

create_power_domain vcc_subsys_pd_top

create_supply_port vcc_subsys_upf
create_supply_net vcc_subsys_upf -domain vcc_subsys_pd_top
add_port_state vcc_subsys_upf -state {HighVoltage 1.05} -state {OFF off}

create_supply_port vcc_gpio_upf
create_supply_net vcc_gpio_upf -domain vcc_subsys_pd_top

create_supply_port vss_upf
create_supply_net vss_upf -domain vcc_subsys_pd_top
add_port_state vss_upf -state {OFF 0.0}

create_supply_net vss1_upf  -domain vcc_subsys_pd_top


set_domain_supply_net  vcc_subsys_pd_top -primary_power_net vcc_subsys_upf -primary_ground_net vss_upf
connect_supply_net vcc_subsys_upf -ports vcc_subsys_upf
connect_supply_net vss_upf -ports vss_upf
connect_supply_net vss1_upf -ports vss_upf
connect_supply_net vcc_gpio_upf -ports {vcc_gpio_upf
                                        i_c69p0snsfabricplustopp/i_gpio_top/vcc_gpio_upf}
connect_supply_net vss_upf -ports {
                                        i_c69p0snsfabricplustopp/i_gpio_top/vss_upf}


create_power_switch sw
-control_port {ctrl pwr_ctrl_sig}
-domain vcc_subsys_pd_top
-input_supply_port {in vcc_subsys_upf}
-on_state {on_state in ctrl}
-output_supply_port {out vss1_upf}
```

Figure 11.3: UPF for subsystem Power Domain

11.3 Results Of UPF

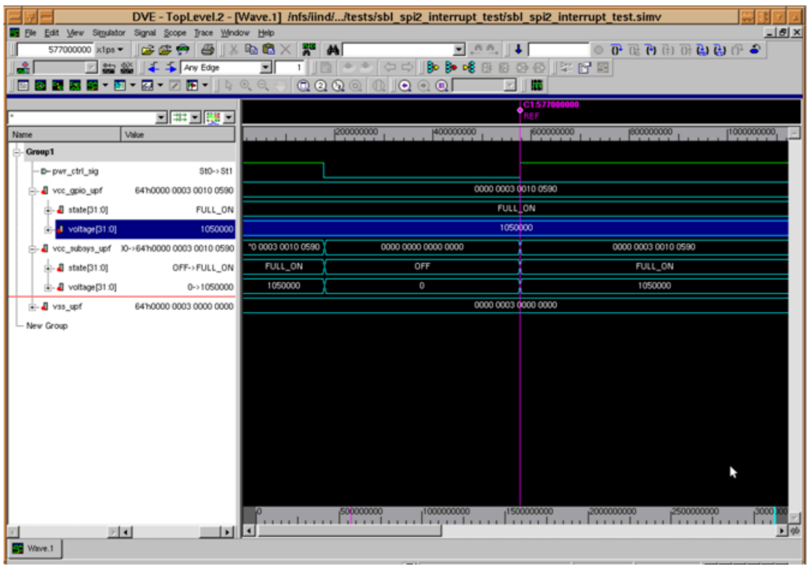
- **Power Domains With Isolation Strategy and Power Switch**



Name	State	Type
top/c69p0snshubprojsbltopp/i_c69p0snsfabricplustopp/i_gpio...		Power Scope
vcc_gpio_pd_top	NORMAL	Power Domain
gpio_iso		Isolation Strat...
Elements		
top.c69p0snshubprojsbltopp.i_c69p0snsfabricplustopp.i...		Element
top/c69p0snshubprojsbltopp		Power Scope
vcc_subsys_pd_top	CORRU...	Power Domain
sw	CORRU...	Power Switch
Elements		
top.c69p0snshubprojsbltopp		Element

Figure 11.4: Power Switch And Isolation Cell

- **pwr_ctrl_sig acting as control switch for vcc_subsys_pd_top**



DVE - TopLevel.2 - [Wave.1] /nfs/iind/.../tests/sbl_spi2_interrupt_test/sbl_spi2_interrupt_test.simv

Name	Value
Group1	
pwr_ctrl_sig	St0->St1
vcc_gpio_upf	64'h0000 0003 0010 0590
state[31:0]	FULL_ON
voltage[31:0]	1050000
vcc_subsys_upf	X0->64'h0000 0003 0010 0590
state[31:0]	OFF->FULL_ON
voltage[31:0]	0->1050000
vss_upf	64'h0000 0003 0000 0000
New Group	

Figure 11.5: Switch sw for vcc_subsys_pd_top

- **Firewall (isolation cells) clamping the value '0' to the inputs (coming from subsystem power domain)**

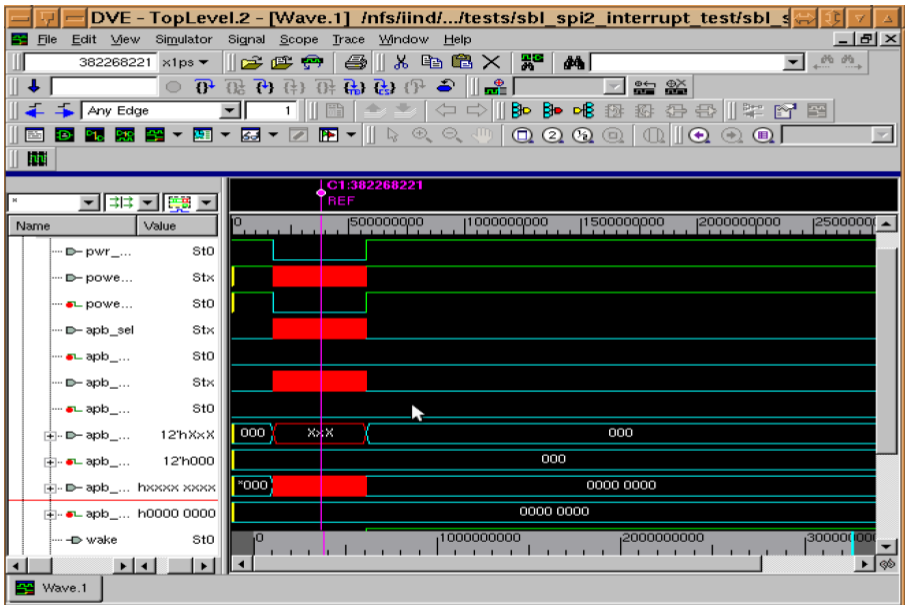


Figure 11.6: Isolated Inputs

- **When subsystem is power down and coming back to the initial state when subsystem is power ON.**

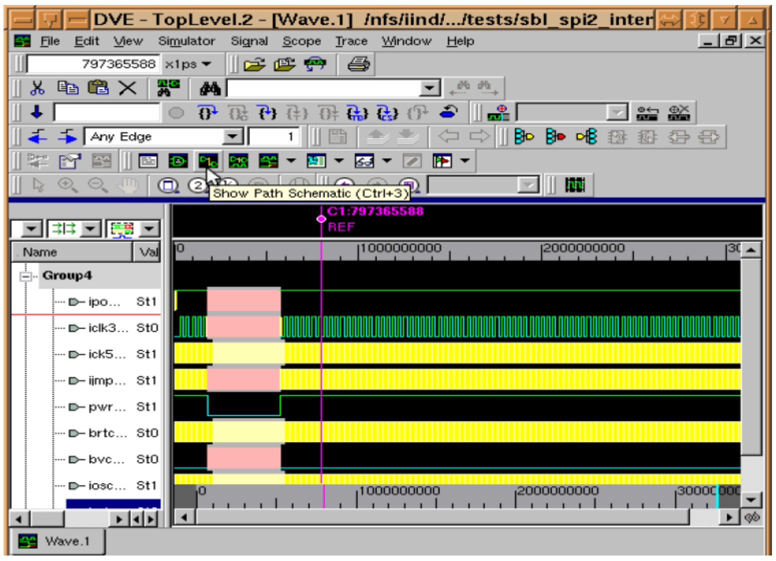


Figure 11.7: Retention Register

Chapter 12

Conclusion

- Datapath module for for test chip project has designed and verified successfully.It is one of the partition for sensor hub project.Also performed the lintra on top and bottom datapath module and fixed all the error and warning in RTL code.
- Lintra is used to check the rtl quality of the design.I have performed the lintra on each partition and on the chip level module.All the fatal errors are fixed from rtl design.Depending upon the design requirement we have waived many of errors and warnings.
- Unified Power Format is inserted in the design.UPF file contains the power switch,isolation cell and retention register.Depending on the value of pwr_ctrl_sig subsytem is powered ON and Off.Also in GPIO domain isolation cells are working.And Retention Register are saving and storing the value.By enabling UPF in a design we are able to save 49 percent power compare to without UPF.For power simulation we are using MVSIM(multivoltage simulator) and for estimation PTpx and Redhawk.

References

[1] Intel ppts for lintra circulating for internal training.

[2] Samir Palnitkar, Verilog HDL: A Guide to Digital Design and Synthesis.

[3] Bhasker J, Verilog HDL Synthesis, A practical Primer, Star Galaxy publishing.

[4] Digital design by Moris Mano

[5] An Engineering Approch To Digital Design by William Fletcher

[6] Low Power Design Flow From Synopsys.

[7] "VLSI Signal Processing" By Keshab K. Parhi

[8] Intel PPT's On Unified Power Format