

# **Power Analysis and Feedback to optimize power of Mixed Signal SoC's**

**M.Tech. Thesis**

By  
**PALASH SOMKUWAR**



**DISCIPLINE OF ELECTRICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY INDORE**

**JUNE 2020**

# **Power Analysis and Feedback to optimize power of Mixed Signal SoC's**

**A THESIS**

*Submitted in partial fulfillment of the  
requirements for the award of the degree  
of*  
**Master of Technology**

*by*  
**PALASH SOMKUWAR**



**DISCIPLINE OF ELECTRICAL ENGINEERING  
INDIAN INSTITUTE OF TECHNOLOGY INDORE  
JUNE 2020**



# INDIAN INSTITUTE OF TECHNOLOGY INDORE

## CANDIDATE'S DECLARATION

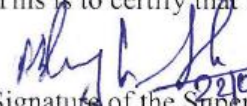
I hereby certify that the work which is being presented in the thesis entitled **Power Analysis and Feedback to optimize power of Mixed Signal SoC's** in the partial fulfillment of the requirements for the award of the degree of **MASTER OF TECHNOLOGY** and submitted in the **DISCIPLINE OF ELECTRICAL ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my work carried out during the time from July 2018 to June 2020 under the supervision of Dr. Abhinoy Kumar Singh, Inspire Faculty, Dr. Saptarshi Ghosh, Assistant Professor, and Seshagiri Guttapalli, External Supervisor.

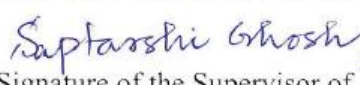
The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

22/06/2020

**Signature of the student with date**  
**PALASH SOMKUWAR**

-----  
This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.


  
Signature of the Supervisor of  
M.Tech. thesis (with date)

  
Signature of the Supervisor of  
M.Tech. thesis (with date)

**DR. ABHINOY KUMAR SINGH**


**DR. SAPTARSHI GHOSH**

-----  
**PALASH SOMKUWAR** has successfully given his M.Tech. Oral Examination held on **22<sup>nd</sup> June 2020**

  
Signature(s) of Supervisor(s) of M.Tech. thesis  
Date: 22/06/2020

  
Convener, DPGC  
Date: 22-06-2020

  
Signature of PSPC Member #1  
Date: 22/06/2020

  
Signature of PSPC Member #1  
Date: 22/6/2020



# Acknowledgment

The thesis would not have been possible without the constant and continuous guidance and the help from many individuals who in one way or another have contributed and extended their valuable assistance in the form of knowledge, efforts, and time for Prepare and complete this study.

First of all, sincere thanks to **Dr. Abhinoy Kumar Singh**, Inspire Faculty, IIT Indore, **Dr. Saptarshi Ghosh**, Assistant Professor, IIT Indore for their continuous support, motivation, and immense knowledge. I would also like to extend my heartfelt gratitude to my PSPC members, **Dr. Vimal Bhatia** and **Dr. Somnath Dey** for their valuable suggestions and feedback. Their guidance has helped me during my time in Thesis research and writing.

**Mr. Prasada Raju SY**, Group Manager, AES, NXP Semiconductors, for accepting me as an intern in his team. I am very grateful for his professional knowledge, friendly care and consideration of my academic requirements

**Mr. Seshagiri Guttapalli**, Lead layout Designer, AES, NXP Semiconductors, Thanks to his patience and firm encouragement, his constructive knowledge sharing at different stages of his work is thought-provoking and they helped me focus on my ideas.

I appreciate **Sai Vijaya Bhaskar Gade and Swaminatha Vijayaraj Anandan**, Design Engineer, NXP Semiconductors, The long discussion helped me understand the technical details of my work. I also thank them for their tireless support and trust in me as a thesis advisor.

Finally, I would like to thank my family and dear friends, Without their love, the most important thing is that the omnipresent God has given me the power to immerse them in all this is impossible.



## ***DEDICATION***

*This work is dedicated to my family, dear friends, and my teachers.....*





## **Abstract**

With technology scaling and an increase of chip complexity, the power consumption of chip has been rising and its power architecture is getting complicated. Many power management techniques like power gating, multi-voltage, multi-threshold are applied to reduce the power dissipation of devices. UPF is an IEEE 1801 standard format to describe the power architecture, also called power intent, including power network connectivity and power reduction methods. It enables verification of power intent in the early phases of the design cycle. The UPF developed should be consistent with the design at all stages of the design cycle and it should be updated according to the modifications made in the design.

In Normal Synthesis Flow we there are no power efficient techniques involved so we go with modified synthesis flow to reduce the power w.r.t to power reduction techniques discussed in the thesis. Normal synthesis flow doesn't give the flexibility with power whereas if we go with the Modified Synthesis flow not only we can effectively manage to reduce power but we can also see the effective utilization of power and wastage. Besides, parallel development of power intent for complex designs, limitations of UPF standard to describe few power intent components effectively and time-consuming conventional UPF flow hinder efficient UPF development and management problems and some of its solutions are discussed.

This methodology also ensures proper restructuring and demotion of UPF along with automation of UPF development, demotion, and verification. This methodology is applied to a design consisting of memory and basic modules and we can say that the flow discussed above has effectively reduced power up to 28% at the RTL stage itself and has been further optimized while doing the physical implementation.



# Table of Contents

<b>Abstract.....</b>	<b>iii</b>
<b>List of Figures.....</b>	<b>viii</b>
<b>List of Tables .....</b>	<b>ix</b>
<b>Acronyms .....</b>	<b>x</b>
<b>1. INTRODUCTION .....</b>	<b>1</b>
1.1 Background .....	1
1.2 Problems.....	1
1.3 Motivation & Objectives.....	2
1.4 Power intent at the Initial stage of ASIC design cycle.....	3
1.5 Power intent Definition .....	4
1.6 UPF flow with the design cycle .....	5
1.7 Challenges in the development of UPF.....	7
1.8 Proposed Idea .....	11
1.9 Thesis Organization.....	12
<b>2. SYNTHESIS.....</b>	<b>13</b>
2.1 Normal Synthesis Flow .....	14
2.2 Elaboration .....	15
2.3 Design Constraints .....	15
2.4 Defining Optimization Settings.....	18
2.4.1 Preserving Instances and Modules.....	18
2.4.2 Grouping and Ungrouping .....	19
2.4.3 Partitioning.....	21
2.4.4 Mapping to Complex Sequential Cells .....	23
2.4.5 Technology Mapping .....	24
2.4.6 Optimizing Total Negative Slack.....	25
2.4.7 Making DRC the very best Priority .....	25
2.5 Performing Synthesis .....	25
2.5.1 RTL Optimization.....	26

2.5.2	Global Focus Mapping.....	26
2.5.3	Global Incremental Optimization .....	26
2.5.4	Incremental Optimization (IOPT).....	26
2.6	Modified Synthesis Flow .....	27
2.7	Concept of Clock Gating.....	28
2.7.1	Integrated Clock Gating cell .....	29
2.8	Power Gating.....	30
2.9	Power Intent .....	31
2.9.1	UPF (Unified Power Format).....	31
2.10	Result and comparison .....	37
2.11	Conclusion.....	41
<b>3.</b>	<b>Place and Route.....</b>	<b>42</b>
3.1	Introduction .....	42
3.2	Floorplanning .....	42
3.2.1	Core Boundary.....	43
3.2.2	Power Planning.....	45
3.3	Placement .....	47
3.2.1	Cell Padding .....	47
3.4	Clock Tree Synthesis.....	48
3.5	Routing.....	49
3.5.1	Global Routing .....	50
3.5.2	Detailed Routing.....	50
3.6	ECO Changes .....	51
3.6.1	Pre-Mask/Unconstrained ECO/All Layer ECO.....	52
3.6.2	Post-Mask/Freeze Silicon ECO/Metal Mask ECO.....	52
3.7	Addition of Spare Cells, Filler Cells & Metal Filling....	54
3.7.1	Spare Cell Addition.....	54
3.7.2	Filler additions .....	55
3.7.3	Metal Filling.....	56
3.8	Conclusion.....	57
<b>4.</b>	<b>CONCLUSION AND FUTURE SCOPE.....</b>	<b>58</b>

4.1 Conclusion.....	58
4.2 Future Scope.....	58
<b>References .....</b>	<b>60</b>

# List of Figures

<u>Figure 1.1 Design Description.....</u>	<u>5</u>
<u>Figure 1.2 UPF Flow.....</u>	<u>6</u>
<u>Figure 1.3 Power architecture of SoC.....</u>	<u>8</u>
<u>Figure 1.4 Restructuring Problem .....</u>	<u>10</u>
<u>Figure 2.1 Normal Synthesis Flow .....</u>	<u>14</u>
<u>Figure 2.2 SDC Block Diagram .....</u>	<u>16</u>
<u>Figure 2.3 Technology Gates.....</u>	<u>24</u>
<u>Figure 2.4 Boolean Logic .....</u>	<u>24</u>
<u>Figure 2.5 Modified Synthesis Flow .....</u>	<u>27</u>
<u>Figure 2.6 Clock Gating .....</u>	<u>28</u>
<u>Figure 2.7 AND Clock Gate and Waveforms.....</u>	<u>29</u>
<u>Figure 2.8 OR Clock Gating with waveforms .....</u>	<u>30</u>
<u>Figure 2.9 Power Gating Techniques .....</u>	<u>30</u>
<u>Figure 2.10 Isolation strategy.....</u>	<u>35</u>
<u>Figure 2.11 Condition for Isolation .....</u>	<u>35</u>
<u>Figure 2.12 Retention Cell.....</u>	<u>36</u>
<u>Figure 2.13 Level Shifter .....</u>	<u>37</u>
<u>Figure 2.14 Area Report without UPF.....</u>	<u>40</u>
<u>Figure 2.15 Area Report with UPF .....</u>	<u>40</u>
<u>Figure 3.1 Cell Locations in a block .....</u>	<u>43</u>
<u>Figure 3.2 Floor plan.....</u>	<u>44</u>
<u>Figure 3.3 Power Planning of Design.....</u>	<u>46</u>
<u>Figure 3.4 Placement of Design .....</u>	<u>48</u>
<u>Figure 3.5 Clock Tree of Design .....</u>	<u>49</u>
<u>Figure 3.6 Routing Techniques.....</u>	<u>50</u>
<u>Figure 3.7 Routing Congestion .....</u>	<u>51</u>

## List of Tables

<i><u>Table 1. Wire load Model.....</u></i>	<i><u>13</u></i>
<i><u>Table 2 Power Domain .....</u></i>	<i><u>32</u></i>
<i><u>Table 3 Supply port and Switches.....</u></i>	<i><u>33</u></i>
<i><u>Table 4 Port states .....</u></i>	<i><u>34</u></i>
<i><u>Table 5 Power State Table .....</u></i>	<i><u>34</u></i>
<i><u>Table 6 Results of Normal Synthesis .....</u></i>	<i><u>38</u></i>
<i><u>Table 7 Result of Modifies Synthesis .....</u></i>	<i><u>39</u></i>
<i><u>Table 8 Gates Report.....</u></i>	<i><u>40</u></i>

# Acronyms

<u>ASIC</u>	<u>Application Specific Integrated circuit</u>
<u>CPF</u>	<u>Common Power Format</u>
<u>CTS</u>	<u>Clock Tree Synthesis</u>
<u>DEF</u>	<u>Design Exchange Format</u>
<u>DRC</u>	<u>Design Rule Check</u>
<u>ECO</u>	<u>Engineering Change Order</u>
<u>EDA</u>	<u>Electronic Design Automation</u>
<u>GDSII</u>	<u>Graphic Database System Information Interchange</u>
<u>HDL</u>	<u>Hardware Description Language</u>
<u>LEC</u>	<u>Logic Equivalence Check</u>
<u>LEF</u>	<u>Library Exchange Format</u>
<u>LIB</u>	<u>Liberty File / Library file</u>
<u>MOS</u>	<u>Metal Oxide Semiconductor (N &amp; P)</u>
<u>PVT</u>	<u>Process, Voltage &amp; Temperature</u>
<u>PnR</u>	<u>Place and Route</u>
<u>PLL</u>	<u>Phase Locked Loop</u>
<u>RTL</u>	<u>Register Transfer Language</u>
<u>SoC</u>	<u>System on Chip</u>
<u>SDC</u>	<u>Synopsys Design Constraints</u>
<u>TCL</u>	<u>Tool Control Language</u>
<u>UPF</u>	<u>Unified Power Format</u>
<u>VLSI</u>	<u>Very Large Scale Integration</u>



# Chapter

## 1. INTRODUCTION

### 1.1 Background

According to the Moore's Law, which states that “in every 18 months the number of transistors doubles itself on a chip” and this trend continues, the law was first proposed in 1965[1]. Till now the Moore's law has not been proven wrong and with the exponential growth of increase in transistors, technology is also reducing its technology nodes which were 50um at first and now it has reached up to 3nm. This shrinking in technology node is increasing challenges and simultaneously the power challenges are also getting tougher,

Due to the continuous expansion of the scale and the growing demand for SoC functions, the complexity in design and Timing analysis becomes very difficult. The expansion speed of the technology is the same as the expansion speed of the interconnection, but the width of the interconnection is reduced each time it is scaled, Hence increase in the resistance of the interconnect. Due to an increase in resistance the analysis of the PVT corner is increasing [2]. The challenge that comes up is that to reduce the power requirements of a design which is also increasing w.r.t. to the scaling of the design the more the design is scaled up and the more the transistors required hence the more power requirements [4]. How to reduce the power requirements and how to manage the power is being discussed in the coming chapters.

### 1.2 Problems

The power consumed in a device is consists of two types – dynamic called switching power, and static called leakage power. In

geometries less than 90nm, leakage power has become the main consumer of power consumption [5], while for larger geometries, switching is a larger contributor. A power reduction strategy can be used to minimize both the powers [3]

$$Power = \alpha * F_c * C * V_{dd}^2$$

To cover the entire physical design process from RTL(Register Transfer Language) to GDSII (Graphic Database System Information Interchange) generation, there are many challenges to complete each step completely and accurately. Modern System-on-Chip demands more power. In both logic and memory, static power is increasing fast and dynamic power is also rising. Overall power is dramatically increasing. If the semiconductor integration continues to follow Moore's Law, the power density inside the chips will reach far higher than the rocket nozzle.

### **1.3 Motivation & Objectives**

The main focus is to learn the different functions of physical design and the details behind each step involved in Physical Design. From RTL design to synthesis to the PnR process, covering the physical design process. After we design the processor according to all steps of ASIC design, we can observe that, consumption of dynamic power in the design is very important in the total power consumption.

Although the main goal here is to adopt different technologies for example clock gating, power gating, and the use of the UPF and with the use of the modified synthesis flow plus the UPF concept, the power consumption was reduced.

## **1.4 Power intent at the Initial stage of ASIC design cycle**

Earlier, power connections of the design were made at the PnR stage. Since the number of cells per chip is very small, even in the PnR stage, it is easy to handle the power pins of the cells to supply power and manage power-related issues. In a chip there are lot of IP's(Intellectual Properties) or so called blocks Also, all blocks or modules in the design are treated as always ON chip. Since the power architecture of design was simple, consisting of supply ports and nets, as there were no power reduction techniques applied to complicate it.

At present, the chip's power dissipation is exponentially increasing with technology nodes reducing and an increase in design complexity as discussed in 1.1. For reducing power, many power reduction methods can be applied, which include power gating, multi-voltage technique, multi-threshold technique, [4] Dynamic Voltage Frequency Scaling (DVFS), etc.. Few of these techniques require special cells like isolation cells, level shifter cells, power switches, etc. to be inserted into the design.

For instance, few blocks are turned off in the power gating technique if they are not active. If we take a scenario of an active block with a non-active to ensure valid signal transmission isolation cell has to be added. If in design, some modules are working at different voltage then level shifter is used. The second half of Chapter 2 details the commonly used power reduction methods and special units. It should be checked that the power reduction techniques applied to the design does not interfere with the design's functionality, which may cause certain modules to be shut down in certain operating modes.

As UPF becomes more complex as chip density increases. This results in a method to define UPF, including power management techniques. This makes it easy to verify these power intents and the power connection between the battery and power supply. However, if the UPF is defined and verified later in the ASIC design flow, it will take a lot of time and effort to fix the error at this stage. Therefore, the power architecture of the chip is defined at the RTL stage of the ASIC design cycle. If the technology to reduce power consumption is defined and verified as early as possible, we can have more choices so that the necessary changes can be made at a later stage without really affecting its design function.

## **1.5 Power intent Definition**

Logic and Functions of a design can be written by Hardware Description Languages (HDL) like VHDL(.vhd), Verilog(.v) and System Verilog (.sv). After designing RTL it is then verified by a verification department. Similarly, we need to use description or standard language to define and verify the power architecture of the chip.

Standard formats used to define the power intent are Common Power Format (CPF) & Unified Power Format (UPF). Therefore, the entire design described by the gate-level netlist defines its logic, function, and power consumption intent of the design, and gives information related to optimal design power consumption, as shown in Figure 1.1.

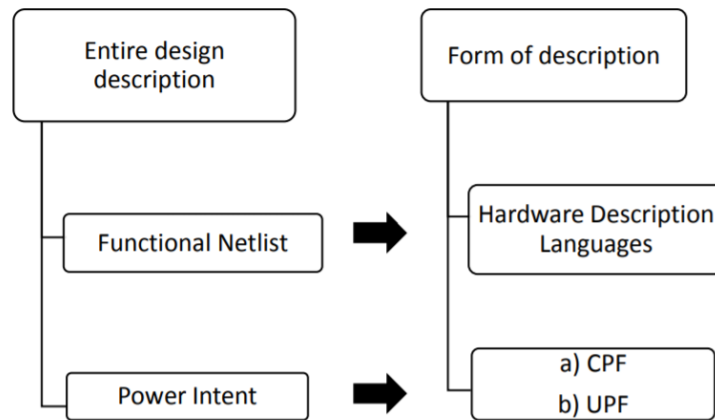
### **CPF**

Cadence developed CPF to specify the power distribution of design and to verify power modes. Low power flow is also defined by

cadence to define the power intent [6]. But the CPF is Specific to Cadence toolset, imposes restrictions on Synopsys' other tools, mentor, etc. leading to compromise inefficiency [7].

## UPF

To allow most EDA tools to use power intent to achieve interoperability during the design cycle [8], the Accellera organization used certain features of CPF in 2007 to propose a more organized universal power format called UPF 1.0. UPF is used for power supply simulation and static verification [9]. It supports TCL language and semantics. UPF is constantly evolving to fulfill requirements, resulting in



**Figure 1.1 Design Description**

The main functions of UPF is to effectively manage power of the design using power intent commands, and Switching activity exchange format (SAIF) for storing activity data for power analysis [11]. IEEE 1801 UPF standard in 2009, called UPF 2.0 version followed by UPF 2.1 in 2013 and UPF 3.0 in 2015 [10].

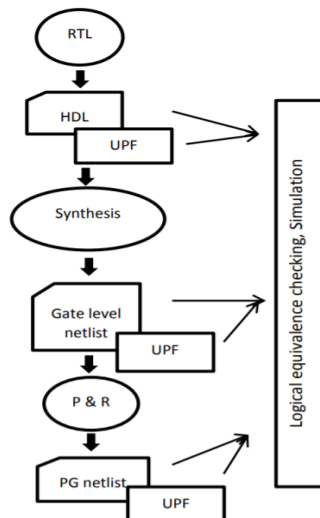
## 1.6 UPF flow with the design cycle

Once the power intent file has been developed using UPF at the RTL stage, the design will be verified for conformance. Because the

design has undergone many changes during the design cycle, The power intention developed during the RTL phase will no longer be consistent with the design [12][13]. Therefore, the UPF should also have a process corresponding to the design change based on the design cycle.

Now, let us have a broader understanding of the ASIC design process, which is often referred to as an application-specific integrated circuit (ASIC) process. The design is divided into three stages for conversion.

RTL stage (front end), synthesis, and physical designing stage. Based on the specification, a design is written in an HDL at the RTL stage [14]. Then It is verified functionally and logically by the designers. In the synthesis stage, it will be changed into a gate-level netlist. The netlist has to pass a functional check and timing verification. Finally, in the physical implementation phase, the design is physically implemented.



**Figure 1.2 UPF Flow**

Let us now discuss the merger of the UPF process and the basic design process, as shown in Figure 1.2 [15]. According to the power supply specifications, UPF is used to define the power supply intent during the RTL phase. The power intent has been verified with dynamic and static RTL level design (2.3). Please note that in the RTL stage, there are no special power management units in the design, such as isolation units, retention units, and level shifters. However, the management strategy is defined in the UPF format, which can be verified by this format. According to the strategy described in the UPF, insert special management units into the design during the synthesis phase.

Since the design has some added cells and undergoes some changes. The UPF is described with the gate-level netlist. In the physical design phase, it is rare to add a unit such as a clamping unit to the design. UPF is made according to the specification are given when the RTL is designed.

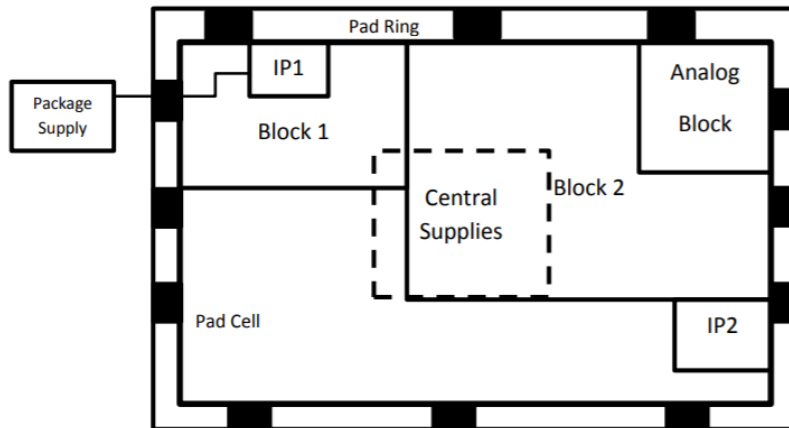
## **1.7 Challenges in the development of UPF**

- There will be some problems when developing during the entire design cycle of UPF

### **Undependable port declarations:**

The power architecture of the chip contains a pad ring from which the main power enters, digital module connection, analog module connection, Padcell connection, various IP power specifications, central bump power supply (if flip-chip

connection package is used [16] As shown in Figure 1.3.



**Figure 1.3 Power architecture of SoC**

Actually, all data with all specifications is planned and written by various professional teams. However, if it is integrated at the SoC level to achieve the entire power consumption intent, it may cause a format error due to different sources involved. Problems such as undependable port declarations can't be detected at an earlier stage, leading to sneaking into a later stage, resulting in a short circuit and opening in the final implementation stage. It is very cost-effective to fix these errors in advance.

- **Bus management**

A bus is a collection of bit lines with ports of a certain width. E.g. a bus with a 32-bit width has a 16-bit IN/OUT port. The bus in the design, especially the bus in the hard macro, is regarded as a single input or output pin. But UPF won't understand that and it will consider other pins as floating pins.



- **Mixing of signal and power connections**

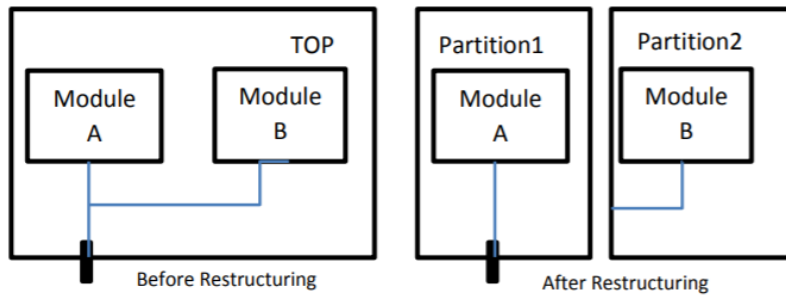
The cell has pins for powering the unit and signal pins for input and output operations. The signal pin belongs to the power supply and is used to define the voltage range. However, we need to ground the signal pins and ground the power supply voltage. Therefore in this case, if UPF considers it as a new pin, the tool used for verifying UPF will be confused between a signal pin or a power pin and mark it wrong.

- **Developing UPF is time-consuming**

Real-life SoCs have millions of gates, and it is very complicated to understand the entire summary. The power architecture of this design has many components to deal with, such as power network, Multi Vt power supply, power domain, level shifter, isolation unit and state keeping. Integrating all data and converting it to a top-level UPF, while also processing all components in the Power Intent file, is very time-consuming.

- **Restructuring UPF**

Now a project is broken into many sub modules and all submodules are handled parallelly at smaller level and finally integrating at top level. When the design is reorganized, the UPF at top-level should also be organized according to the changes, so it becomes important at the partition level to meet the specification of each design unit. This results in each unit having a dedicated power intent format, so at the partition level, the UPF can be brought to a further stage along with the design partition and can be modified and verified accordingly.



***Figure 1.4 Restructuring Problem***

We know that a reorganization has occurred and a level change has occurred. This poses a problem in UPF, because the changes in the connection takes place so UPF should also be modified accordingly to suit the scope and power domain. In Figure 1.4, module A, module B, TOP, partition 1, partition 2, module A, and the module before reorganization are created on the top layer. Also, if the supply network is changed and connected to different modules according to the reorganization , the supply network will be suspended in that partition unit , because declaration the supply port was not being done in UPF. This results in the erroneous realization of zoning power intent [17]. It takes a lot of time to analyze and deal with all these problems manually.

#### ○ **Static verification of power intent is Time Consuming**

The static verification of UPF/CPF is usually completed in the 3 stages of the ASIC design cycle first RTL stage second synthesis, and third physical design stage, because in these stages, the design will undergo major changes. The method of using the RTL itself to update and make the following changes is very time-consuming.

## 1.8 Proposed Idea

Due to the increased complexity of the chip and the high market demand for low-power devices, power has become one of the primary design constraints.

- In addition to the design function description, the core idea of developing power intents for this complex chip was introduced in recent years. Many years ago, organizations were in the process of effectively adapting and using them.
- Simultaneously developing the design power architecture will eventually lead to a lot of inconsistencies and provide the basic knowledge of power intent for the top UPF integrators. Any misstatements of components belonging to UPF may not be discovered early, resulting in an faulty power-intent design.
- The conventional way of defining UPF is not fully mature and it becomes difficult to solve practical implementation challenges, such as reorganization and downgrade.
- While developing top level of UPF, the structural reorganization and degradation of each design partition, and If performed manually, the static verification of UPF is consumes a good amount of time and is not that efficient. Due to this there is a delay in development, thereby increasing time to manufacture.

In this work, some solutions are proposed to effectively reduce a certain amount of power and obtain the best knowledge to overcome the challenges in UPF Flow.

- Input power data alignment: The main cause of the inconsistent description of power supply is analyzed, and the alignment process of power data received from various sources is proposed.
- Reorganization and demotion process: an integral part of the top power intent Analyzed the factors that were affected during the reorganization process and established a universal UPF The process of reorganization and demotion is proposed. Some parts of the method can also be executed automatically to avoid wasting time. Synthesizing using the conventional synthesis process can later compare and analyze different reports, why do we need to use UPF and its effectiveness in the conventional process and its importance in the future.
- A low power consumption method is proposed, and the UPF and its components are appropriately defined according to the design requirements, and the report is analyzed.

## 1.9 Thesis Organization

The organization of the next chapter is as follows. Chapter 2 provides insights on the synthesis flow using UPF to reduce power consumption and the modified synthesis flow and compares the two results. Chapter 3 gives an abstract view of the design through Innovus, that is, layout planning and effective use of the design, where we can see the actual power domain and its respective modules. Chapter 4 gives the conclusion of the whole project.

## Chapter

### 2. SYNTHESIS

It is the process where we convert HDL(Hardware Description Language) into specific Gate connections such as NAND, NOT, OR, AOI, OAI, etc. to form a Gate level Netlist including nets, sequential and combinational cells, and their connectivity. Each logic gate is typically available in different sizes in the technology library (.LIB) corresponding to different drive strengths. When the gate drives other gates, the gate output node faces some amount of Load Capacitance depending on the fanout and size of different gates. Hence Time Constant (RC) coefficient contributes to the major part of delay which varies according to the length of wire and the load Capacitance.

Resistance	$2 \times 10^{-4}$	R per Unit Length
Capacitance	$3 \times 10^{-4}$	C per unit length
Area	$1 \times 10^{-1}$	Area Per Unit Length
Slope	1.3	Used for extrapolating linearly
Fanout_length	(1, $2 \times 10^{-3}$ ) , (2, $7 \times 10^{-3}$ )	at fanout 1 length of the wire is $2 \times 10^{-3}$ and at 2 it is $7 \times 10^{-3}$ .

*Table 1. Wire load Model*

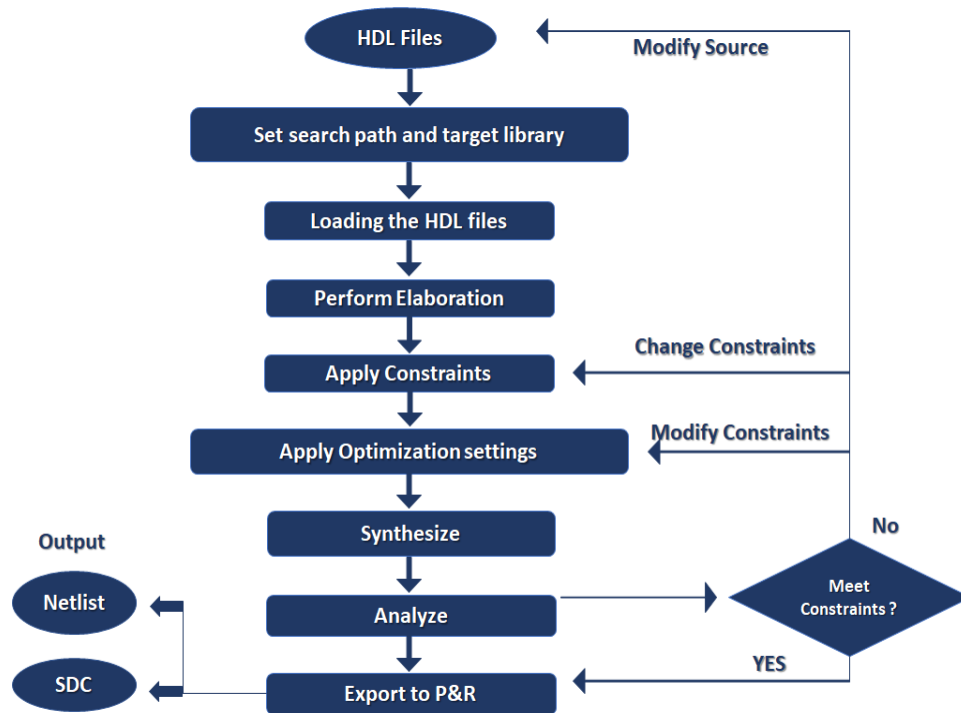
Suppose the fanout is 5 and we only have the net length till fanout 2, So to calculate net length we use the following expression :

$$\text{Length of Net} = [\text{net length at fanout 2}] + (5-2) * \text{Slope}$$

These all arithmetic operation happens in Synthesis tool, the tool used for Synthesis is Cadence Genus tool. To do Synthesis we need to first have some input files such as RTL (.v or .vhd), technology library(.lib), Liberty Exchange Format(.lef), Design Exchange

Format(.def), Synopsys Design Constraints (.sdc), Cap-table. According to the industry standards first, we do normal synthesis and then after generating a def file from the PnR(Place and Route) tool we again do the synthesis using def information called Physical synthesis.

## 2.1 Normal Synthesis Flow



**Figure 2.1 Normal Synthesis Flow**

The above synthesis process is conventional [18][19]. This is a crucial step between logic and layout. It is the process of converting the synthesis output (not the real gate in your technology library) into a real logic gate. Tech Mapper performed this important step, which is a surprisingly elegant algorithm that involves recursive coverage of trees. Another place to learn other practical computer science knowledge in VLSI CAD.

## **2.2 Elaboration**

While doing Elaboration the first thing that tool does is checking any syntax error after that all the code and arithmetic operation is converted into technology-independent logic gates and complex cells like counters FIFO etc. [19]. It also analyses design hierarchy like top-level design and its sub hierarchies. It also removes empty switches and dead branches or leaf nodes. It also detects any asynchronous reset and converts synchronous to D flipflop or D latch. Detecting FSM logic and extracting the information about the number of inputs, outputs bits, state bits, and converting the FSM logic to basic logic [19]. Creating memory cells, merging DFF to memory write and memory read and mapping memory cells to basic logic are the important steps which is done while elaborating the design.

## **2.3 Design Constraints**

Design Constraints are of two types one is Timing based constraint and the second one is based on physical constraints. Design constraints make sure that design meets its performance goals at any PVT corner (Process Voltage and Temperature).

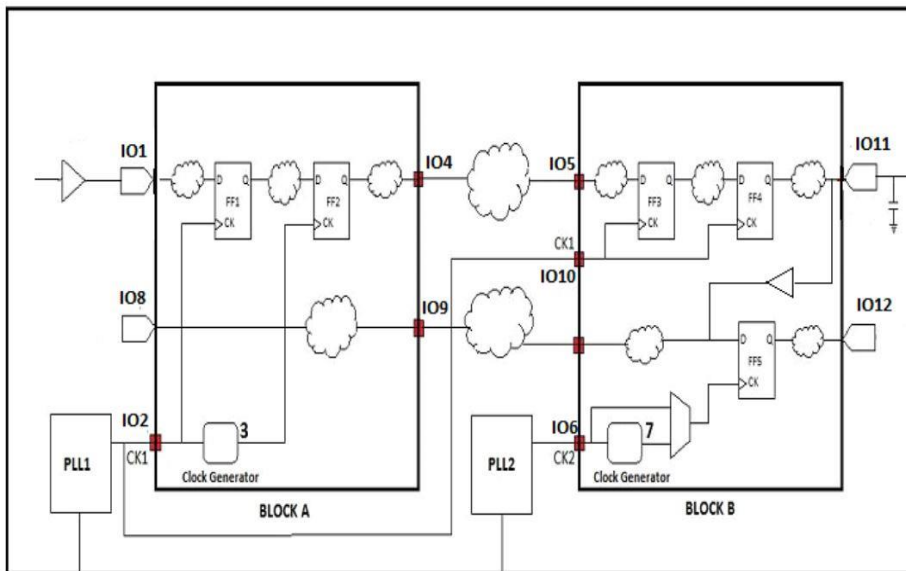
The Synopsys Design Constraints (SDC) format is a timing-based format used to specify performance goals i.e. design intent, including timing, power, and area constraints of the design [21]. SDC format is used in EDA tools for a design to synthesize and analyze. SDC is particularly written in TCL. With the help of an example, we will understand how we write SDC and some of its commands. The information specified in design constraints is the constraints, operating conditions, timing exceptions, wire load models, timing constraints, system interface, area constraints, multi-voltage & power optimization constraints, and logic assignments.

In fig 2.2, we can see that there are two blocks and two PLL, PLL is used to generate clock and constraints are defined at I/O pins [22].

- **I/O pins (2,6,10)**

As there are two PLL's which means two clocks are present, hence we need to define two different clocks in a design. So we use “create\_clock ” for I/O 2 and I/O 6, clock CK1 is also going to I/O 10 i.e. Block B so same we need to define for I/O 10.

```
create_clock -period 10 -waveform {0 6} -name
my_clk1 [get_ports IO2]
```



**Figure 2.2 SDC Block Diagram**

### **I/O pin 1**

It is an input pin driving the port so we use “set\_driving\_cell.”, after giving this constraint the port will have certain delay provided in the cell library which we will be using while synthesis.

```
set_driving_cell -lib_cell BUF [get_ports IO1]
```



### Clock Generators (3,7)

The clock is generated from the master clock with a new frequency. “*create\_generated\_clock*” is used for this purpose which is defined w.r.t. master clock.

```
create_generated_clock -divide_by 2 -source  
[get_ports IO2] -name clk_div [get_registers  
FF2
```

### I/O pin 4

It is an output port so we need to put a constraint on output delay . “*set\_output\_delay*” this command specifies the delay between the output port and the next sequential cell.

```
set_output_delay 1.7 -clock my_clk1  
[all_outputs]
```

### I/O Pin 5

Similar to output delay, input delay is mentioned. “*set\_input\_delay*” tells the timing requirements at the input port, telling the timing of the external path to the input.

```
set_input_delay 1.7 -clock my_clk1  
[all_outputs]
```

### I/O pin 11

In the fig, we can see that the at I/O pin 11 load capacitance is present so we need to define the “*set\_load* ” command which describes an external load capacitance connected to top-level port including pin as well as wire capacitance.

```
set_load 20 [get_ports IO11]
```

In the design, we can see that there are two different asynchronous clocks, CK1, and CK2 so it's better to define separate clock groups by the command “*set\_clock\_group* ”.

## Some Timing Exceptions

### False path

In any design some path are of no use . there can be number of reason that we need to mention false path but in the above design we can see that there are two different asynchronous clocks CK1 and CK2 so hence we need to set any path from CK1 And CK2 a false path [22]. “*set\_false\_path*” command is used for this purpose.

```
set_false_path -from [get_clocks my_clk1] -to  
[get_clocks my_clk2]
```

### Max and Min path

I/O pins 8,9: In the design, we can see that IO pin 8,9 constitutes to input to output delay. So for this, we need to specify min/max delay by the commands “*set\_max\_delay*” and “*set\_min\_delay*”.

```
set_max_delay 0.15 -from IO8 -to IO9
```

### Multicycle path

Data takes one clock cycle to propagate from launch flop to capture flop. But it can also take more than one clock cycle. We should set such paths to the multi-cycle path using the “*set\_multicycle\_path*” command. The multi-cycle path is always a multiple of the clock cycle.

## 2.4 Defining Optimization Settings

### 2.4.1 Preserving Instances and Modules

By default, Genus will perform optimizations that can result in logic changes to any object in the design. We can prevent any logic changes in a block while still allowing optimizations in the surrounding logic [18], by using the *preserve* attribute:

```
set_db object .preserve true
```

where the *object* is a hierarchical instance, primitive instance, module, or submodule name.

Genus can also preserve instances and modules while allowing certain special actions to be performed on them. This allows for greater flexibility.

```
For example : set_db [vfind / -inst
g1] .preserve size_ok / preserve
delete_ok / preserve size_delete_ok.
```

## 2.4.2 Grouping and Ungrouping

Grouping and ungrouping is a comprehensive strategy which is useful when we need to change the design hierarchy. Genus gives u a set of commands that allow us to *ungroup/group* any existing instance, design, or module [18][23].

- *Grouping* makes a level of hierarchy around a group of instances.
- *Ungrouping* releasing a level of the hierarchy

### Grouping

If our design includes many modules, we can group some of the module instances into another single module for placement or optimization purposes using the group command.

```
group -name CRITICAL_GROUP [vfind / -
inst I1] [vfind / -inst I2]
dir:/designs/top_counter/hinsts/CRITICAL_
GROUPi Or
hinst:top_counter/CRITICAL_GROUPi
```

## Ungrouping

**Manual Ungrouping :** `ungroup instance`

`ungroup [vfind / -inst CRITICAL_GROUP]`

If we defined an exception on a pin of a hierarchical instance that we ungroup, Genus tries to preserve the exception when we ungroup that hierarchical instance.

In most cases, Genus adds a `CDN_EXCEPTION` buffer for each hierarchical instance pin for which an exception was defined to retain the exceptions on these pins.

In the following cases, no buffer is added:

- The driver of the hierarchical pin has only one fanout and the driver is the pin of a sequential or hierarchical instance. The exception of the instance pin will be moved to the driver pin.
- The driver of the hierarchical pin has a multiple fanouts but the load pin of the hierarchical pin is either a sequential or hierarchical pin.
- The exception of the hierarchical pin will be moved to the load pin.

## Automatic Ungrouping

Genus can also automatically ungroup user hierarchies during synthesis . During the high effort, RTL optimization (`syn_generic` with `syn_generic_effort` set to high) Genus ungroups user hierarchies containing only datapath hierarchies. During high effort global mapping (`syn_map` with `syn_map_effort` set to high) Genus explores ungrouping of user hierarchies to show more

opportunities for structuring, mapping, and other optimizations [18][23].

The tool takes the wire load model under consideration before ungrouping the instance to create sure that the timing doesn't degrade after ungrouping. The tool distinguishes between critical and non-critical instances.

Automatic ungrouping won't be performed during medium or low effort mapping.

- to stop automatic ungrouping, set the subsequent root attribute before synthesis:

```
set_db auto_ungroup none
```

If the attribute is ready to both (default), automatic ungrouping will happen during high effort mapping.

- to stop ungrouping of all instances of a module, set the ungroup\_ok attribute for the module to false.

```
set_db [vfind /des*/design -  
module name] .ungroup_ok false
```

- to stop ungrouping of a particular instance, set the ungroup\_ok attribute for the instance to false:

```
set_db [vfind /des*/design -  
inst name] .ungroup_ok false
```

### 2.4.3 Partitioning

Partitioning refers to the process of breaking down (partitioning) a design into more manageable block sizes. This can speed up the running time and improve the memory footprint without sacrificing the accuracy of the synthesis

result. Enable partition, set the `auto_partition` attribute to `true` before synthesis:

```
set_db auto_partition true
```

### Setting Boundary Optimization

Genus performs boundary optimization for all hierarchical instances within the design during synthesis.

- Constant propagation across hierarchies
- Removing undriven or unloaded logic connected
- Collapsing equal and opposite pins
- Hierarchical pin inversion
- Rewiring of equivalent signals across the hierarchy

If the output pin always has the same (or inverted) logic value as the input pin, the layered boundary pin is the feedthrough pin. These feedthrough pins are routed around the module, and these pins do not require connections or logic inside the module.

We can control boundary optimization during synthesis using the subsequent attributes:

- `boundary_opto`
- `delete_unloaded_seqs`
- `boundary_optimize_constant_hpins`
- `boundary_optimize_equal_opposite_hpins`
- `boundary_optimize_feedthrough_hpins`
- `boundary_optimize_invert_hpins`

► To disable boundary optimization on the module, type the subsequent command:

```
set_db [vfind /des* -module  
name] .boundary_opto false
```

► to stop Genus from removing flip-flops and logic if they're not transitively fanning out to output ports, use the `delete_unloaded_seqs` attribute:

```
set_db[modules or/]
.delete_unloaded_seqs false
```

#### 2.4.4 Mapping to Complex Sequential Cells

Genus's sequential mapping function uses complex flip-flops in the library to increase the number of cells in the design, sometimes increasing the domain or timing (depending on the design).

When inferring the trigger in RTL, Genus performs sequential mapping. For instantiated triggers, Genus does not perform any optimization other than size.

Asynchronous flip-flop inputs are automatically inferred from the sensitivity list and also the conditional statements within the always block.

► to stay the synchronous feedback logic immediately ahead of the sequential elements, type the subsequent command:

```
set_db hdl_ff_keep_feedback true
```

Setting this attribute may harm the realm and timing.

```
set_db optimize_constant_1_flops false
```

#### Deleting Unused Sequential Instances

Genus optimizes sequential instances that transitively don't fanout to a primary output. This information is generated within the log file. this can be especially relevant if we see unmapped points in formal verification.

```
set_db delete_unloaded_seqs false
set_db optimize_constant_0_flops false
```

```
set_db optimize_constant_1_flops false
set_db optimize_constant_latches false
```

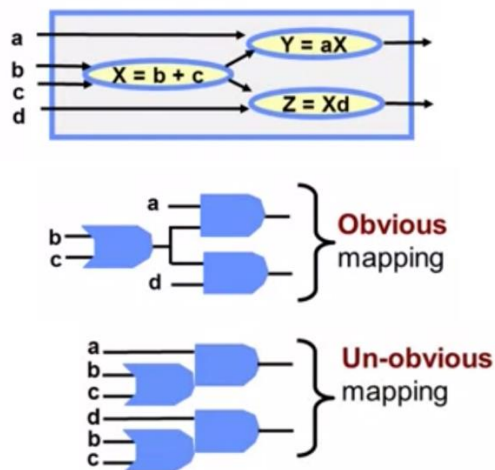
### 2.4.5 Technology Mapping

The technology library consists of cells such as NAND, NOT, OR, AOI, OAI, AND etc.



*Figure 2.3 Technology Gates*

The HDL mainly consists of Boolean logic consisting of millions of operations happening to get some output. Let's take a small Boolean logic abstract. Each cell in the library has some Cost and depending upon the Cost we chose the best mapping [20].



*Figure 2.4 Boolean Logic*



#### 2.4.6 Optimizing Total Negative Slack

By default, Genus optimizes Worst Negative Slack (WNS) to attain the timing requirements. During this process, it tries to repair the timing on the foremost critical path. It also checks the timing on all the opposite paths. However, Genus won't work on the opposite paths if it cannot improve timing on the WNS.

```
set_db tns_opto true
```

Genus work on all the paths to scale back the overall negative slack (TNS), rather than WNS.

#### 2.4.7 Making DRC the very best Priority

By default, Genus tries to repair all DRC errors, but not at the expense of timing. If DRCs don't seem to be being fixed, it may be due to infeasible slew issues on input ports or infeasible loads on output ports. we'll be able to force Genus to repair DRCs, even at the expense of timing, with the `drc_first` attribute.

► to confirm DRCs get solved, even at the expense of timing, type the subsequent command:

```
set_db drc_first true
```

By default, this attribute is false, which implies that DRCs won't be fixed if it introduces timing violations.

- Creating Hard Regions
- Deleting Buffers and Inverters Driven by Hard Regions
- Preventing Boundary Optimization through Hard Regions

### 2.5 Performing Synthesis

Synthesis is the process of reworking our HDL design into a gate-level netlist, given all the required constraints and optimization settings [18].

In Genus, synthesis involves the subsequent four processes:

- RTL Optimization
- Global Focus Mapping
- Global Incremental Optimization
- Incremental Optimization (IOPT)

### **2.5.1 RTL Optimization**

In the RTL optimization process, Genus performs optimizations such as data path synthesis, resource sharing, speculation, multiplexing optimization, and carry-on arithmetic (CSA) optimization. After completing this step, Genus will perform logic optimization, such as structured and redundant deletion.

### **2.5.2 Global Focus Mapping**

Genus performs global focus mapping at the top of optimization independent of RTL technology (during the `syn_map` command). This step includes simultaneous reorganization and mapping plans, including optimizations such as splitting, pin swapping, buffering, pattern matching, and isolation.

### **2.5.3 Global Incremental Optimization**

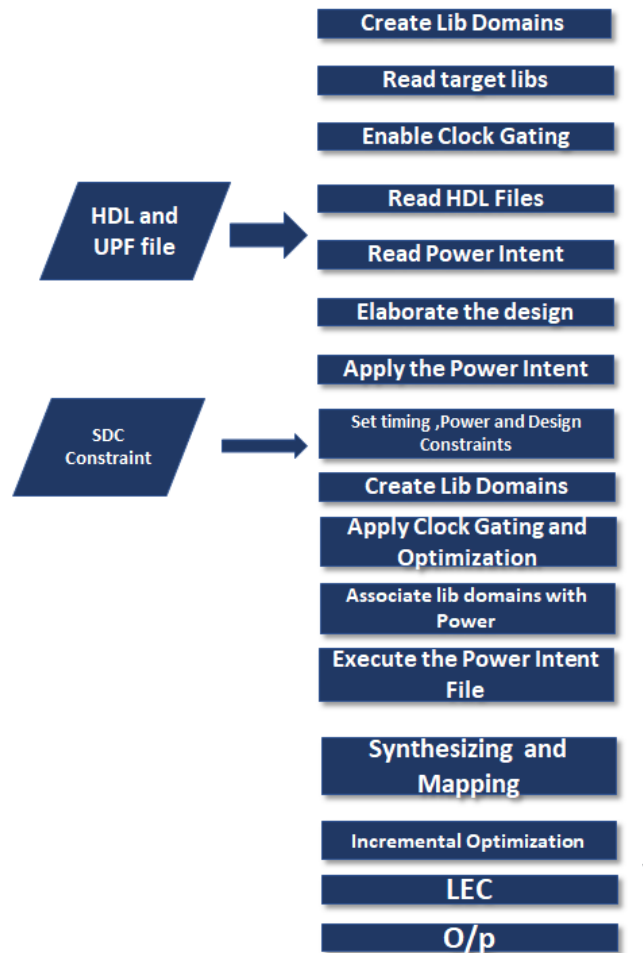
After global focus mapping, Genus performs comprehensive global incremental optimization. This stage is mainly for regional optimization and power optimization (if enabled). The optimization performed at this stage includes global adjustment of the unit size and optimization of the buffer tree.

### **2.5.4 Incremental Optimization (IOPT)**

The final optimization performed by Genus is incremental optimization. The optimization performed during IOPT can improve timing and area, and fix DRC violations. The optimizations performed at this stage include multi-bit cell

mapping, incremental clock gating, incremental retiming, contact cell insertion, and allocation deletion. By default, timing has the highest priority, and if Genus causes timing violations, Genus will not fix DRC violations. This priority can be overridden by setting the drc\_first attribute to true. In this case, all violations will also be fixed because those paths have positive slack.

## 2.6 Modified Synthesis Flow

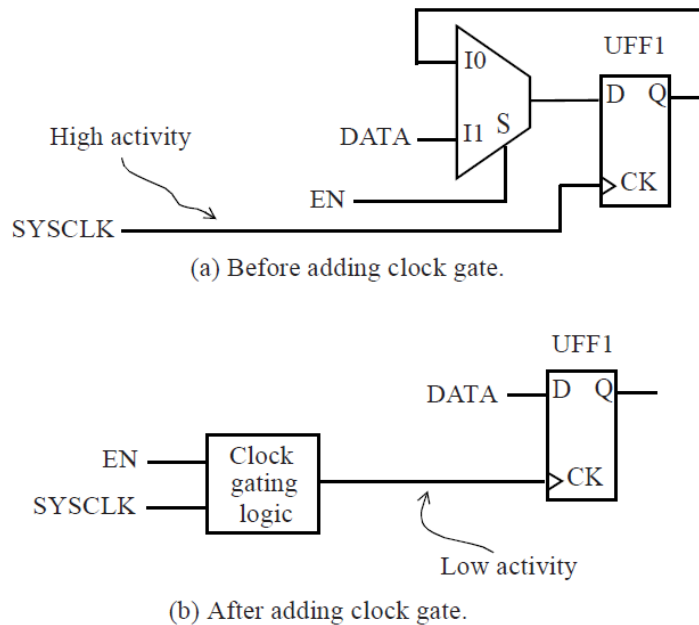


*Figure 2.5 Modified Synthesis Flow*

Above figure shows the modified synthesis flow which gives the better results w.r.t. to old normal synthesis flow. In this flow we use power management techniques such as clock gating, Power intent information such as UPF or CPF. We also use the library in which power management cells are present such as isolation cell, level shifter cell, state retention flip flop etc.

## 2.7 Concept of Clock Gating

Clock activity at the flip-flops contributes to a significant component of the total power [21]. A flip-flop dissipates power due to clock toggle even when the flip-flop output does not switch. The purpose of clock gating is to minimize this contribution by eliminating the clock activity at the flip-flop during clock cycles when the flip-flop input is not active



**Figure 2.6 Clock Gating**

In the above fig 2.6 a flip-flop receives new data only when the enable signal  $EN$  is active otherwise it retains the previous state. During the time  $EN$  signal is inactive, the clock toggling at the flip-flop do not cause any output change though the clock activity still results in the power dissipated inside the flip-flop.

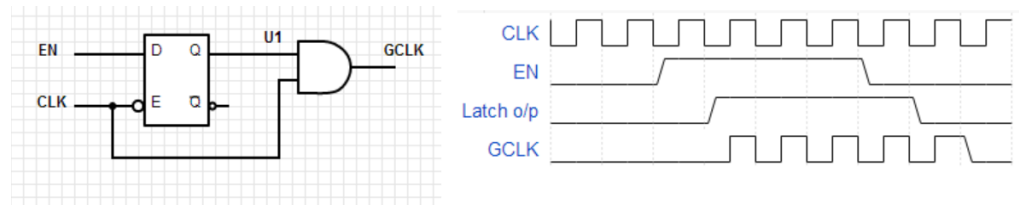
The logic reassembling through clock gating introduces gating of the clock at the flip-flop pin. An example of the transformation due to clock gating is illustrated in Fig 2.6 b. The clock gating thus ensures that the clock pin of the flip-flop toggles only when new data is available at its data input.

For Clock Gating we mostly use AND or OR gate. Suppose we use AND gate with the clock. The high EN edge may come anytime and increase the possibility that the EN edge doesn't coincide with the clock edge, hence we use Integrated clock gating cell.

### 2.7.1 Integrated Clock Gating cell

- AND Gate with High Enable

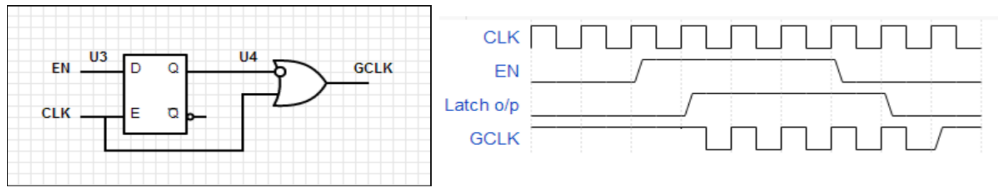
In the fig negative edge-triggered latch synchronizes EN signal to the clock. The GCLK is on only when the latch o/p is high and held low when EN is low [23].



**Figure 2.7 AND Clock Gate and Waveforms**

- OR Gate with High Enable

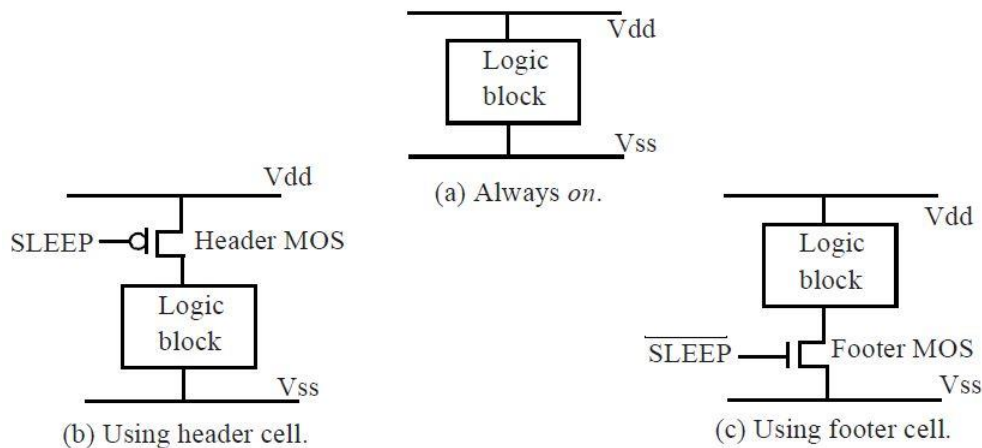
In the fig negative edge-triggered latch synchronizes EN signal to the clock.  $GCLK$  is held high when  $EN$  is low and the latch output is inverted at the OR input, hence  $CLK$  is passed when the input gets low.



**Figure 2.8 OR Clock Gating with waveforms**

## 2.8 Power Gating

In power Gating, we switch off the power supply so that power to the inactive blocks can be switched off. In the fig, we can see a MOS device which is added in series with the power supply. The control signal *SLEEP* is configured so that the footer or header MOS device is *on* during normal operation of the block[].



**Figure 2.9 Power Gating Techniques**

The footers or headers are normally implemented using multiple power gating cells that correspond to multiple MOS devices in parallel. The footer and header devices introduce a series of resistance to the power supply. If the value of the *on*-resistance is not small, the IR drop through the gating MOS device can affect the timing of the cells in the logic block [25].

## **2.9 Power Intent**

We specify power intent information in the form of UPF or CPF. CPF is the format that can be used only in Cadence tools whereas UPF is accepted by all tools and is universal. So we will see how to provide power intent information in UPF [10].

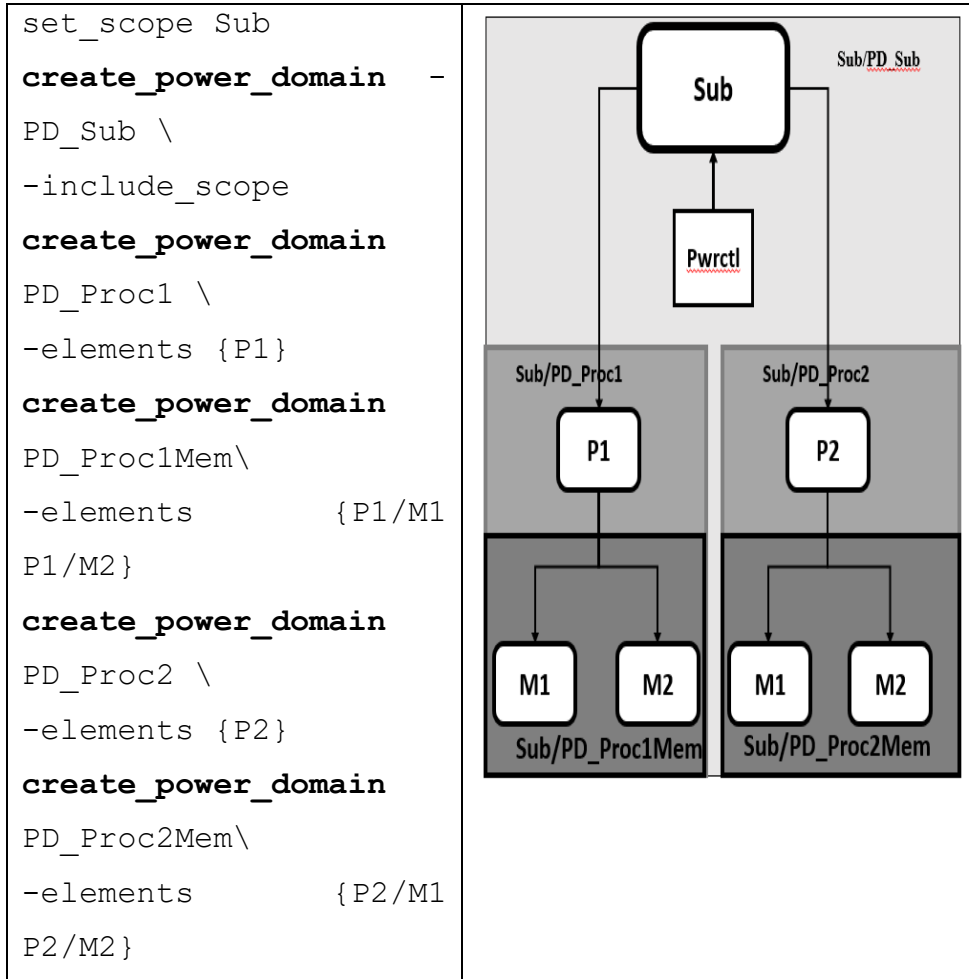
### **2.9.1 UPF (Unified Power Format)**

The UPF specification was approved by IEEE as the IEEE-2009 standard, and the IEEE 1801 standard was released. UPF specification provides a standard format for specifying all power specific design, power constraints, and its functionality [11][12]. The UPF format captures all the design and technology related power constraints.

Before going to proper script of UPF including commands and constraints and all the power intent information we will first see the significance of WHY UPF? Since power is the major issue for any design and all the chip designer admire the lowest possible power, so reducing power is the major issue in day to day design and which can be sufficiently achieved by using UPF. The cells or technology cells which helps us to reduce power are Isolation cell, state retention cell, Level shifter cell etc. Now consider a chip of millions of gate , in those millions of gates we can say that there will be groups of gates sharing a common power supply requirements called a specific power domain.

## Components of UPF

**Power Domain** : Groups of elements having common power supply requirements



*Table 2 Power Domain*



## Power Supply Network : Distribution of ports, nets, set and switches.

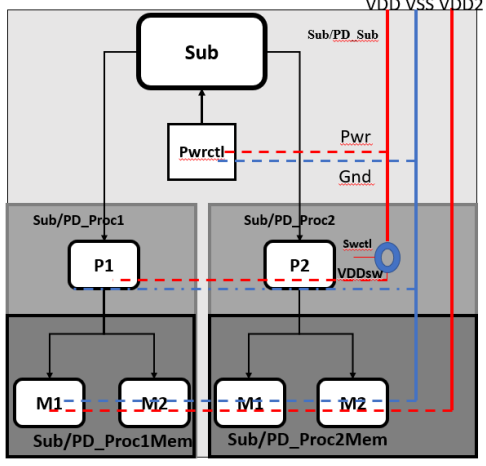
<pre> <b>create_supply_net</b> VDDsw          -domain PD_Proc  <b>create_power_switch</b> SW\ -domain PD_Proc \ -output_supply_port (swout VDDsw) \ -input_supply_port {swin Pwr} \ - control_port{swctl}\ -off_state {SWoff !swctrl}  <b>set_domain_supply_net</b> PD_Proc \ -primary_power_net VDDsw\ -primary_ground_net Gnd </pre>	 <pre> <b>create_supply_port</b> VDD2 - domain PD_Mem <b>create_supply_net</b>      Gnd reuse -domain PD_Mem <b>set_domain_supply_net</b> PD_Mem \ -primary_power_net VDD2 -primary_ground_net Gnd </pre>
--	---

Table 3 Supply port and Switches

### Power State table

It specifies the combination of state of each power domain.

Before defining power states for a domain we need to define port states by command

e.g. `add_port_state VDD -state {ON_1.0}`

	PD_Sub	PD_Proc	PD_Mem	
<i>State/Supply</i>	VDD(port)	VDDsw(net)	VDD2(port)	VSS(port)
Normal	ON_1.0	ON_1.0	ON_0.8	ON_0.0
Sleep	ON_1.0	OFF	ON_0.8	ON_0.0
Hibernate	ON_1.0	OFF	OFF	ON_0.0

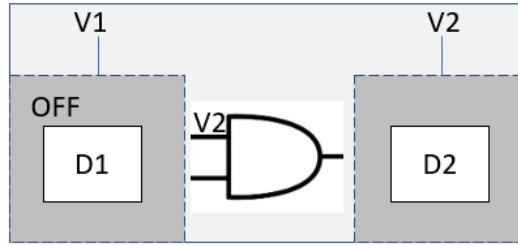
*Table 4 Port states*

```
create_pst PST1 -supplies {VDD VDDsw VDD2 VSS}
add_pst_state Normal -pst PST1 -state {ON_1.0
ON_1.0 ON_0.8 ON_0.0}
add_pst_state Sleep -pst PST1 -state {ON_1.0 OFF
ON_0.8 ON_0.0}
add_pst_state Hibernate -pst PST1 -state {ON_1.0
OFF OFF ON_0.0}
```

PD_Sub	PD_Proc	PD_Mem	
VDD(port)	VDDsw(net)	VDD2(port)	VSS(port)
ON_1.0	ON_1.0 or OFF	On_0.8 or OFF	ON_0.0

*Table 5 Power State Table*

**Isolation Strategies:** It is the strategy used to prevent the unnecessary flow of current into the block which is not in use and to differentiate from the Always ON block [12].



**Figure 2.10 Isolation strategy**

In the above fig, we can see the two power domains D1 and D2, D1 is the shutdown domain, and D2 is the always-on domain. Now if there are a few signals from D1 to D2, at any time if the D1 goes to in-active mode i.e. Switched OFF and if the signal goes from D1 to D2 gets some unwanted signal or noise from some source it can trigger the logic in the D2 domain, which will do unwanted functionality in the circuit, to prevent these Isolation cells are used.

In the table, we can see that PD\_Proc should be separated by an always ON domain PD\_Sub both in sleep and hibernate mode. so we'll see how to make it isolated from the Always ON domain.

Sleep	ON_1.0	OFF
Hibernate	ON_1.0	OFF

**Figure 2.11 Condition for Isolation**

```

set_isolation ISOpoc -domain PD_proc -applies_to
outputs -clamp_value 0 -isolation_power_control
PWR \
-isolatin_ground_control GND
set_isolation_control ISOpoc -domain PD_proc \
-isolation_signal pISO -isolation_sense high -
location parent

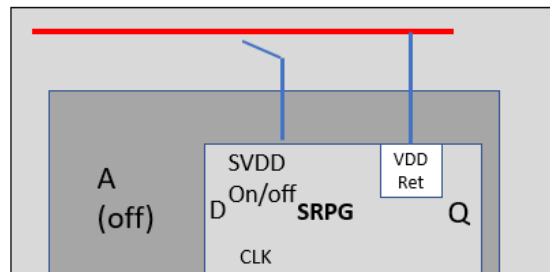
```

The control signal (pISO) must be already present in the HDL description.

Now we have isolated the shutdown domain from the always ON domain but we need to retain the last values stored in the register in the shutdown domain for which we use the retention cells.

### Retention Strategies

Retention cells are used to retain the values when the domain power goes into the OFF state. These retention cells are special low leakage flip-flops used to hold data of the main register of the power gated block [12]. Thus the internal state of the block during power-down mode can be retained and loaded back to it when the block is reactivated, retention cells are always powered up.



**Figure 2.12 Retention Cell**

In the table we need to retain the data of the domain which gets off after the isolation is enabled by the following commands

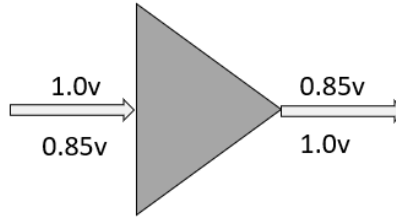
```
set_retention RET1 -domain PD_Proc -
retention_power_net PWR -retention_ground_net GND
set_retention_control RET1 -domain PD_Proc
-save_signal {SRb posedge} -restore_signal {SRB
negedge}
```

The retention signal has to be defined in the HDL description.

### Level Shifter strategies :

Level shifters are used in a design where Multi Vt supply is used, now the two voltage domain D2 and D1 in fig , if there are few signals from D2 (1.0 V) are travelling to D1 0.85V domain, and the

supply voltage is different then we need to put the level shifter in that domain.



**Figure 2.13 Level Shifter**

The main function of Level shifter is to shift the voltage from higher to lower or lower to higher. In UPF we need to define by the following commands for table 5.

```
set_level_shifter LSmem -domain PDmem -threshold  
0.1 -applies_to both -rule both -location self
```

## **2.10 Result and comparison**

After loading all the scripts mentioned in the Normal and Modified synthesis flow and giving specific commands for specific operation following results were obtained and compared.

### Power report for Normal Synthesis flow

Instance: /phy

Power Unit: W

PDB Frames: /stim#0/frame#0

Category	Leakage	Internal	Switching	Total	Row%
Memory	0.00	0.00	0.00	0.00	0.00%
Register	5.546 e-10	6.428e-06	1.36e-08	6.442e-06	89.92%
Latch	0.00	0.00	0.00	0.00	0.00%
Logic	5.096e-11	6.56e-07	3.167e-08	6.88e-07	9.60%
bbox	0.00	0.00	0.00	0.00	0.00%
Clock	8.69e-12	1.39e-08	2.02e-08	3.42e-08	0.48%
pad	0.00	0.00	0.00	0.00	0.00%
pm	0.00	0.00	0.00	0.00	0.00%
Subtotal	6.14e-10	7.09e-06	6.55e-08	7.16e-06	100%
%	0.01%	99.08%	0.91%	100%	100%

*Table 6 Results of Normal Synthesis*

In the above table, we can see that the total power taken by the design is  $7.16454 \times 10^{-6}$  . And the power management (pm) module is turned off. Now this design was then loaded with UPF and all the power-saving techniques involved to reduce power

### Power report with Modified Synthesis Flow

Instance: /phy

Power Unit: W

PDB Frames: /stim#0/frame#0

Category	Leakage	Internal	Switching	Total	Row%
Memory	0.00	0.00	0,00	0.00	0.00%
Register	1.58 e-10	1.799e-06	1.562e-07	1.96e-06	37.57%
Latch	0.00	0.00	0.00	0.00	0.00%
Logic	7.81e-11	4.03e-08	1.547e-07	1.946e-06	3.73%
bbox	0.00	0.00	0.00	0.00	0.00%
Clock	6.04e-11	1.30e-06	9.40e-07	2.24e-06	43.07%
pad	0.00	0.00	0.00	0.00	0.00%
pm	4.09e-10	5.41e-07	2.72e-07	8.13e-07	15.62%
Subtotal	7.05e-10	3.68e-06	1.52e-06	5.20e-06	100%
%	0.01%	70.74%	29.25%	100%	100%

*Table 7 Result of Modifies Synthesis*

- In the above table of Modified synthesis flow, we can see that the power is  $5.20700 \times 10^{-6}$ . Normal synthesis power which is  $7.16454 \times 10^{-6}$  .
- Total power is reduced by appx 28% ( pm module is still off in without UPF condition and is taking Zero power then also power is more in without UPF stage)
- Register power is reduced by appx 70%
- Clock power is increased.

Without UPF (Normal)	With UPF (Modified)																																																																																																		
<div><div>Gates</div><table><thead><tr><th>Gate</th><th>Instances</th><th>Area</th></tr></thead><tbody><tr><td>SEN_CKGTPLT_V5_1</td><td>1</td><td>3.528</td></tr><tr><td>total</td><td>1</td><td>3.528</td></tr></tbody></table></div> <div><table><thead><tr><th>Type</th><th>Instances</th><th>Area</th><th>Area %</th></tr></thead><tbody><tr><td>sequential</td><td>18</td><td>0.000</td><td>0.0</td></tr><tr><td>inverter</td><td>5</td><td>0.000</td><td>0.0</td></tr><tr><td>clock_gating_integrated_cell</td><td>1</td><td>3.528</td><td>100.0</td></tr><tr><td>unresolved</td><td>1</td><td>0.000</td><td>0.0</td></tr><tr><td>logic</td><td>43</td><td>0.000</td><td>0.0</td></tr><tr><td>physical_cells</td><td>0</td><td>0.000</td><td>0.0</td></tr><tr><td>total</td><td>68</td><td>3.528</td><td>100.0</td></tr></tbody></table></div>	Gate	Instances	Area	SEN_CKGTPLT_V5_1	1	3.528	total	1	3.528	Type	Instances	Area	Area %	sequential	18	0.000	0.0	inverter	5	0.000	0.0	clock_gating_integrated_cell	1	3.528	100.0	unresolved	1	0.000	0.0	logic	43	0.000	0.0	physical_cells	0	0.000	0.0	total	68	3.528	100.0	<div><div>Gates</div><table><thead><tr><th>Gate</th><th>Instances</th><th>Area</th></tr></thead><tbody><tr><td>HD040_LS40_LVLDBUFE1_PY2_4</td><td>1</td><td>11.290</td></tr><tr><td>HD040_LS40_LVLDBUFE1_Y2_4</td><td>10</td><td>112.896</td></tr><tr><td>HD040_S_SDPRB2RETIS00X1</td><td>8</td><td>149.587</td></tr><tr><td>SEN_ADDH_1</td><td>6</td><td>13.759</td></tr><tr><td>SEN_AN2_1</td><td>1</td><td>1.058</td></tr><tr><td>SEN_AN4_1</td><td>1</td><td>1.411</td></tr><tr><td>SEN_CKGTPLT_V5_1</td><td>1</td><td>3.528</td></tr><tr><td>SEN_E02_S_0P5</td><td>1</td><td>1.588</td></tr><tr><td>SEN_FDPRBQ_V2_1P5</td><td>8</td><td>33.869</td></tr><tr><td>SEN_FSDPRBQ_D_1</td><td>2</td><td>9.526</td></tr><tr><td>SEN_INV_1</td><td>1</td><td>0.529</td></tr><tr><td>SEN_INV_S_0P65</td><td>3</td><td>1.588</td></tr><tr><td>SEN_INV_S_1</td><td>2</td><td>1.058</td></tr><tr><td>SEN_NR2_1</td><td>1</td><td>0.706</td></tr><tr><td>SEN_NR2_S_1</td><td>1</td><td>0.706</td></tr><tr><td>SEN_OR2_1</td><td>1</td><td>1.058</td></tr><tr><td>SEN_OR3_1</td><td>1</td><td>1.235</td></tr><tr><td>total</td><td>49</td><td>345.391</td></tr></tbody></table></div>	Gate	Instances	Area	HD040_LS40_LVLDBUFE1_PY2_4	1	11.290	HD040_LS40_LVLDBUFE1_Y2_4	10	112.896	HD040_S_SDPRB2RETIS00X1	8	149.587	SEN_ADDH_1	6	13.759	SEN_AN2_1	1	1.058	SEN_AN4_1	1	1.411	SEN_CKGTPLT_V5_1	1	3.528	SEN_E02_S_0P5	1	1.588	SEN_FDPRBQ_V2_1P5	8	33.869	SEN_FSDPRBQ_D_1	2	9.526	SEN_INV_1	1	0.529	SEN_INV_S_0P65	3	1.588	SEN_INV_S_1	2	1.058	SEN_NR2_1	1	0.706	SEN_NR2_S_1	1	0.706	SEN_OR2_1	1	1.058	SEN_OR3_1	1	1.235	total	49	345.391
Gate	Instances	Area																																																																																																	
SEN_CKGTPLT_V5_1	1	3.528																																																																																																	
total	1	3.528																																																																																																	
Type	Instances	Area	Area %																																																																																																
sequential	18	0.000	0.0																																																																																																
inverter	5	0.000	0.0																																																																																																
clock_gating_integrated_cell	1	3.528	100.0																																																																																																
unresolved	1	0.000	0.0																																																																																																
logic	43	0.000	0.0																																																																																																
physical_cells	0	0.000	0.0																																																																																																
total	68	3.528	100.0																																																																																																
Gate	Instances	Area																																																																																																	
HD040_LS40_LVLDBUFE1_PY2_4	1	11.290																																																																																																	
HD040_LS40_LVLDBUFE1_Y2_4	10	112.896																																																																																																	
HD040_S_SDPRB2RETIS00X1	8	149.587																																																																																																	
SEN_ADDH_1	6	13.759																																																																																																	
SEN_AN2_1	1	1.058																																																																																																	
SEN_AN4_1	1	1.411																																																																																																	
SEN_CKGTPLT_V5_1	1	3.528																																																																																																	
SEN_E02_S_0P5	1	1.588																																																																																																	
SEN_FDPRBQ_V2_1P5	8	33.869																																																																																																	
SEN_FSDPRBQ_D_1	2	9.526																																																																																																	
SEN_INV_1	1	0.529																																																																																																	
SEN_INV_S_0P65	3	1.588																																																																																																	
SEN_INV_S_1	2	1.058																																																																																																	
SEN_NR2_1	1	0.706																																																																																																	
SEN_NR2_S_1	1	0.706																																																																																																	
SEN_OR2_1	1	1.058																																																																																																	
SEN_OR3_1	1	1.235																																																																																																	
total	49	345.391																																																																																																	

Table 8 Gates Report

## Area reports

Without UPF and With UPF

Instance	Module	Cell Count	Cell Area	Net Area	Total Area
phy		67	258.557	7.335	265.891

Figure 2.14 Area Report without UPF

Instance	Module	Cell Count	Cell Area	Net Area	Total Area
phy		49	345.391	74.318	419.710
u_tm	TopMod	43	316.638	57.267	373.905
x1	mod_SW	23	185.044	35.890	220.933
x4	mod_A0_1	8	62.446	6.708	69.153
x3	sync_mod_A0	7	54.684	4.890	59.574
x2	mod_A0	3	5.998	2.445	8.442

Figure 2.15 Area Report with UPF



## **2.11 Conclusion**

This chapter describes the synthesis process and the flow for synthesis. In the chapter, we discussed the two different flows one is Normal Synthesis Flow and the other one is Modified Synthesis Flow. In modifying the Synthesis flow we were able to reduce the power by approx. 28% which was the ultimate aim but having a trade-off between area and number of gates. This Modified flow is a low power flow that shows good results w.r.t. to the power considerations and in future of low power design it has greater prospects to show good results and the power value obtained after synthesis of the design can be more optimized in PnR flow.

## Chapter

### **3. Place and Route**

#### **3.1 Introduction**

This is an act of designing a chip by placing components of the chip (such as different doors) and arranging different macros in the chip that other designers have designed. By the name "Physical Design", it should be understood that certain physical activities are involved in the design. These sports activities are created here to meet the timing requirements. The way to create timing to complete certain sports activities is to place some buffers at certain locations to meet the transition requirements, increase or decrease the size of the unit to reduce the unit delay, and achieve some reduction in line length. Now we will discuss the various steps involved in physical design, such as layout planning, power planning, zoning, layout, layout legalization, clock tree synthesis (CTS), routing.

#### **3.2 Floorplanning**

The physical design starts with the layout plan, which is usually the most important step to complete the layout and this plan defines the size and allocation of chips/blocks including power planning, placing hard macros, and finally reserving space for quality units [27]. After that, all subsequent steps (such as placement, routing, and timing closure) will totally depend on the quality of the layout plan. The plan must be repeated many times to successfully obtain the best plan. The floorplan related definitions are described below.

### 3.2.1 Core Boundary

While floor-planning we define the shape and size of the chip/module. The digital design or a chip can have a rectangular/square shape, while the sub-blocks may have a rectangular or linear shape. Standard cells and other IP blocks refer to the placed area as the core boundary. The power wiring space is outside the core boundary. The floor-plan is can be optimized by parameters such as.

#### 3.2.1.1 Aspect ratio

It is the ratio of height to width, defining a rectilinear structure.

#### 3.2.1.2 Core Utilization

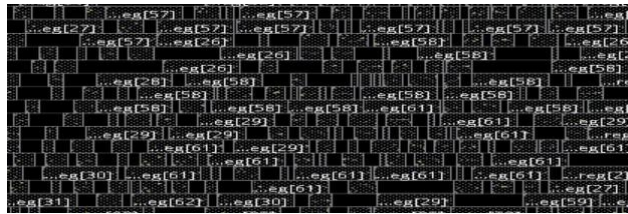
As mentioned above, the core area is the area where the standard unit is located. Outside this core area, the area used by the cell or macro defines the "core utilization" and is designated as

$$\text{Core utilization} = \frac{(S_a + M_a)}{T_a}$$

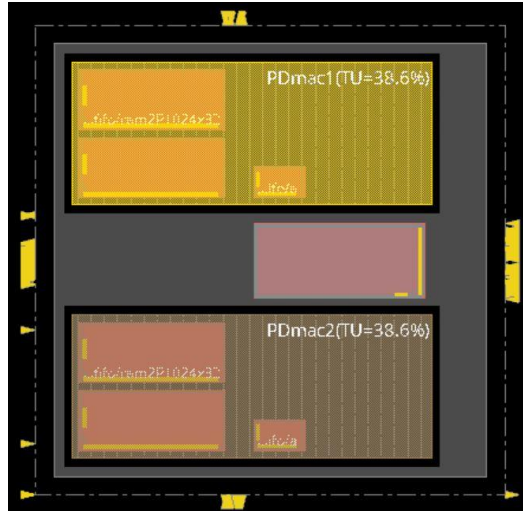
$S_a$  = chip area occupied by standard cells

$M_a$  = chip area occupied by Macro placement

A core utilization of 0.65 means that 65% of the total area can be used for standard cell placement and the leftover i.e. 25% can be used for chip routing.



*Figure 3.1 Cell Locations in a block*



**Figure 3.2 Floor plan**

### **3.2.1.3 I/O Placement / Pin Placement**

When designing the digital top, we hope to place the IO pad and IO buffer in the chip [27]. The RTL designer provided the side and relative position of the PAD. Also, the package is selected by selecting the minimum or maximum chip size. To place IO, some modifications need to be made in the RTL file to connect the PAD unit to the IO port through instantiation. If we want to design a module, we must place its pins around the boundary to connect to the higher wiring layer. The tool can also use DEF files for custom floor plans.

### **3.2.1.4 Macro Placement**

Once we have prepared a floor plan of size and shape to create rows for the mass unit, we can manually place the macro. The location where the macro is to be placed will depend on the flying lead connection between the macro and the IO port, thereby making the length of the connection line shorter.

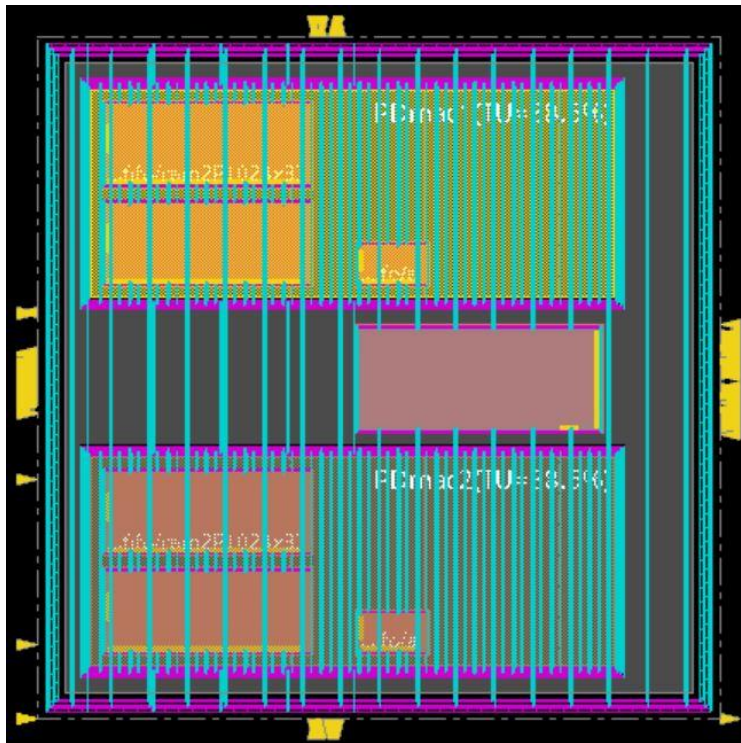
### 3.2.2 Power Planning

Power planning begins with the power stripes and ring, which provides support for chips with VDD and VSS. The power rails are planted on the top metal layers because the highest metal has a larger width, the resistivity of the metal is less, so the IR drop will be less. After that we perform special wiring to create a track for the correct placement of the mass cells. These traces are created by metal1 with alternating VDD and VSS to enable the unit to draw power from it. Therefore, to take care of the integrity of the metal1 line in the IR landing, we added a higher metal to the power strip to enable the strip to strengthen the metal1 line, because the track is horizontal and the strip is vertical [25]. Now introduce the connection and definition and importance of Tie and Tap units, as shown below:

There are many special units in the library. These units are essentially non-functional, but they have high requirements and importance. Well tap, End cap, De-cap, Spare, Filler and Tie cells.

- ◆ *Well Tap cells*: It connects the substrate and N-well to VDD and VSS, respectively, to avoid latch-up effects.
- ◆ *End Cap cells*: It is just an extension, used to confirm that there will be no gap between the hole and the implant layer to prevent DRC violation.
- ◆ *De-cap cells*: It is a capacitor for decoupling because no voltage source is ideal, so the gate draws a lot of power due to glitches in the power line. Therefore, to avoid the bottom and power rebound, a decap battery is used.
- ◆ *Spare Cells*: Support ECO changes after the tape out exceeds the design range.

- ◆ *Filler cells*: The filled cells are used to establish the continuity of the N-well, so the implant layer can be established on the high-quality cell
- ◆ *Tie cells*: It is used to connect certain gate ports to VDD and VSS of 1 or 0, respectively.



***Figure 3.3 Power Planning of Design***

Then there is the function of the power plan, that is, by connecting the Well Tap. The power rails are planted on the top metal layers because the highest metal has a larger width, the resistivity of the metal is less, so the IR drop will be less. Therefore, by keeping the metal wires high, all units will get less distorted power.

### **3.3 Placement**

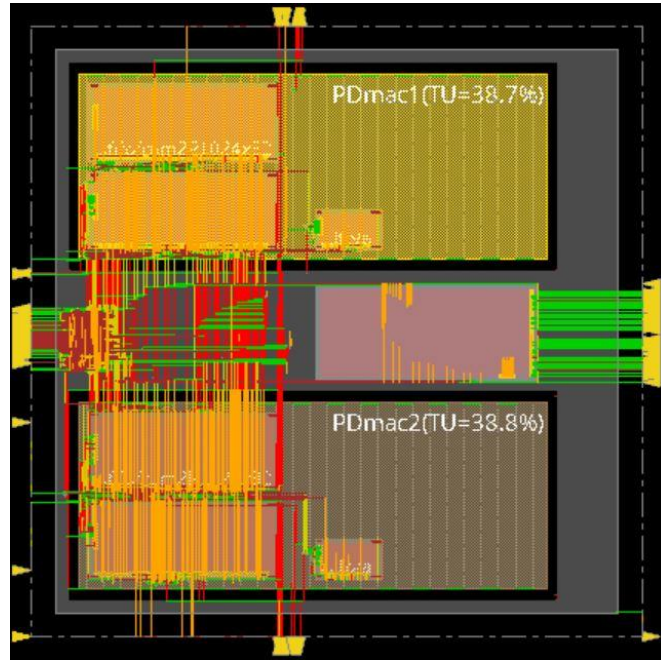
After preparing the plan according to the floor plan, we chose to place the cell in the core area of the chip[20]. In the process of placing cells, many things must be noted, such as density restrictions, overlapping of cells, and filling of cells. There are two types of placement, such as timing-driven placement, and the second is congestion-driven placement. The difference between the two is that in the case of timing drive placement, regardless of this increase in congestion, the placer will be very important for reducing the length of the wire, and in the case of congestion driven placement, The bit device will reduce the congestion through diffusion, which is very important[24] [27]. Does phasing out cells in the entire chip result in the worst timing performance

When performing the placement step, we must choose an optimization called in-place optimization, so we must also perform in-place optimization. During pre-placement optimization, the tool will optimize the netlist by folding the high fan-out net and deleting any invalid paths. When performing in-place optimization, the tool will optimize location by performing maintenance from a timing and congestion perspective

#### **3.2.1 Cell Padding**

When importing a floor plan into a floor plan, we will check if there are shelving violations. If there is any violation of hold before or after placement, the connection between the plan and the netlist is poor, because the design is Pre-CTS, which means that the clock is correct so that all triggers are reached at the same time. To avoid these holding conflicts, the tool will add buffers or delay cells, which will require space to place these cells. Therefore, to perform this buffer addition operation, some spaces should be filled around the

trigger, which is named "Cell Padding". Module padding means that additional space is required inside and around the module when it is placed. Modules which are more denser i.e. high utilization area need more space to avoid congestion and thus require filling. After placing the appropriate filler and optimizing, the chip is shown in Figure 3.4.

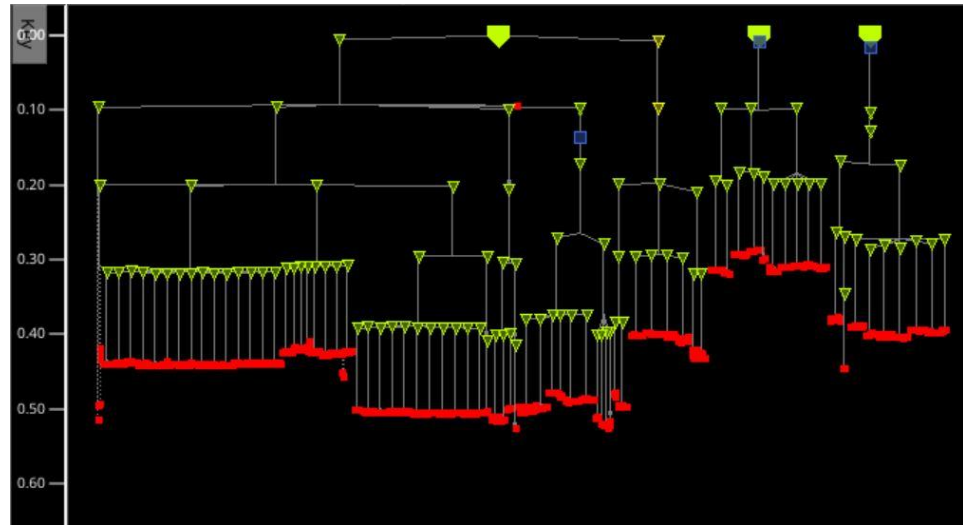


*Figure 3.4 Placement of Design*

### 3.4 Clock Tree Synthesis

Since the clock in the design is an ideal clock, we need to pay attention to the correct distribution of the clock to all registers. Therefore, we perform clock tree synthesis. By adding a buffer/inverter to the clock path, the clock can reach all flip-flops with minimal skew. In the technology library, there are some special and efficient buffers and inverters (called CLKBUF/CLKINV) used in CTS[30]. They are different from normal buffers/inverters in terms of conversion time, delay and size. There are many types of clock trees available, but we used H-tree type clock trees.





*Figure 3.5 Clock Tree of Design*

### 3.5 Routing

Routing is the stage after clock tree synthesis and optimization, where-

- Determined the exact path used to interconnect standard cells and macros and I/O pins.
- Create electrical connections using metal and vias in the layout, which are defined by logical connections in the netlist.

After CTS, we will get information on all placed cells, blocking, clock tree buffer/inverter and I/O pins. The tool relies on this information to electrically complete all connections defined in the netlist, thereby enabling-

- There is minimal DRC conflict when routing
- The design is 100% wired with a minimum LVS violation rate.
- The least violations are related to SI.
- No congestion hot spots or few congestion hot spots.
- Meet timing DRC.
- Timing QoR is very good

The different tasks performed during the routing phase are as follows:

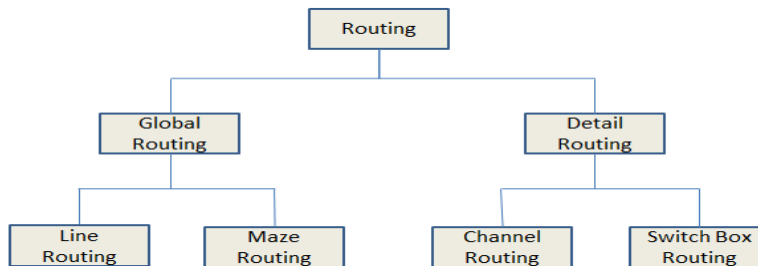
- Global routing (also performed during the placement phase).

- Track assignment.
- Detailed routing.
- Search and repair.

The technology involved in routing is shown in Figure 3.6.

### 3.5.1 Global Routing

In the global routing stage, the router uses a netlist (that is, logical connection) to route the connection. This is not the correct route because it is not optimized. This route only involves connections driven from the netlist. This route is also called "trial route". Through the "trial route", we check the congestion in the wiring by allowing a smaller number of metal layers to be routed and checked for congestion. If half of the available metal layers are allowed to enter the route and the congestion report is less than the design, it can be considered routable. In this way, the global routing takes effect.

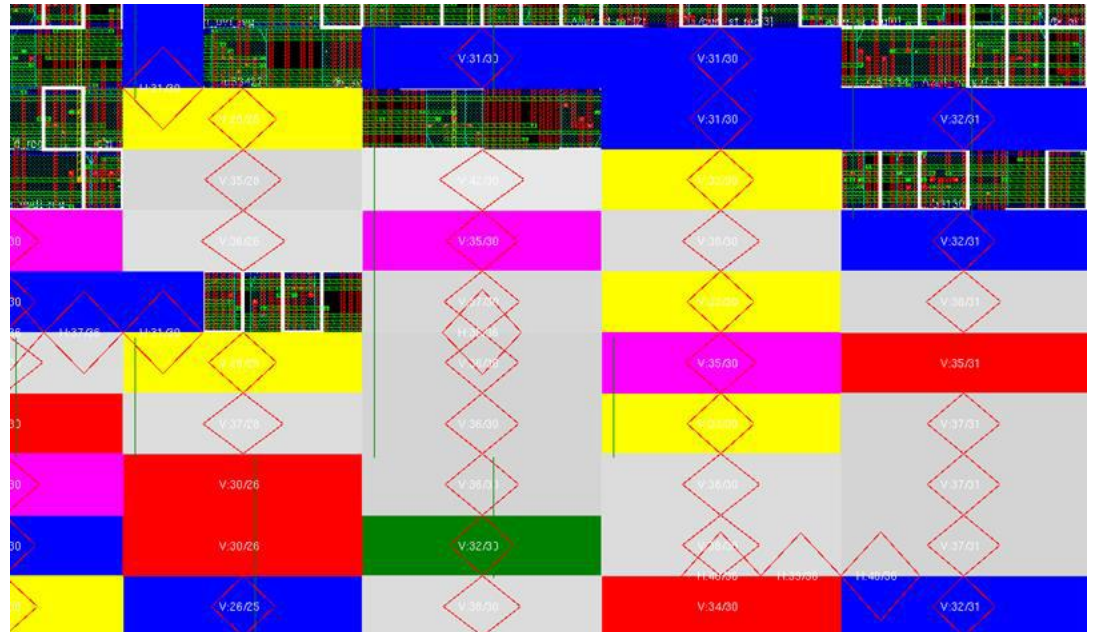


**Figure 3.6 Routing Techniques**

### 3.5.2 Detailed Routing

As the name implies, the routing of the router is more detailed, so the length of the routing can be shortened, and the design is optimized according to the designer's timing. To this end, the router first tries to find the channel to be

routed, and then reduces the wire length. When the wire length is minimum, the RC interconnect will have less RC delay, thereby reducing time and increasing operating frequency. If there is any routing congestion, the congestion looks as shown in Figure 3.7.



*Figure 3.7 Routing Congestion*

### 3.6 ECO Changes

An engineering change order or ECO is a way to incorporate the latest changes in our design. ECO saves money and time and is common in the industry. When we talk about ECO, we actually are talking about ECO in the layout. Therefore, we usually start with ECO on the gate-level netlist. The designer needs to edit the gate-level netlist, make the same changes in RTL, and then pass all the verification to the layout. Before starting to edit the layout, make sure that ECO has passed formal and functional verification. In addition to the variables or commands related to "freeze-silicon",

PnR tools also use two types of ECO, and a similar design process is also used for PnR tools.

In the design process, the chip has to go through all steps, such as layout plan, layout, CTS, and wiring, all steps must be completed at least 10 iterations. One year. Sometimes, only when incremental changes are made to existing chips, new chips must be adopted. In this case, the chip will not take more than one year to perform all steps again. Everything is done incrementally, which reduces engineering costs, time to market, and manufacturing costs.

There are two types of ECO Changes

- Pre-Mask/Unconstrained ECO/All Layer ECO
- Post-Mask/Freeze Silicon ECO/Metal Mask ECO

### **3.6.1 Pre-Mask/Unconstrained ECO/All Layer ECO**

An all layer ECO is typically done before mask generation. There is no restriction on the changes permitted in the layout. we need to have a robust ECO flow ready as we are gearing up for tapeout. ECOs can be done at any stage in the design flow, post-place, post CTS and post-route. ECOs are used to

- Fix timing violations – constraints may be lacking on specific networks. ECO can add buffers/delays to control the timing behavior of the design
- Fixing bugs – Last minute bugs are the norm. With the advent of the streaming spree, simulations may throw errors that everyone missed until then. If it can be repaired via ECO, engineers prefer to use it if the design's runtime and complexity are sufficient to ensure that existing results/databases are preserved

- Adding functionality – We may not see many signed off ECOs in this category. However, the use of ECO can complete smaller additions without having to redo the design, and this may happen

An unconstrained ECO typically has the following stages.

- Adding/Deleting Cells – At this stage, there are no restrictions on adding cells except for design/layout constraints.
- Updating the connections – The net connections have to be updated for the existing and newly added cells.
- Placement – Tools can automatically place ECO instances, but I find it best to manually place them for best performance.
- Routing – Today's tools can automatically identify and route changed networks.

Even if we usually use an unconstrained ECO process before the tape, and use it as part of the PnR flow, it can be done after the tape. There is no savings in mask generation costs, but if the required changes are minimal, the design cycle can be greatly shortened.

### **3.6.2 Post-Mask/Freeze Silicon ECO/Metal Mask ECO**

These are all done after tape-out, and by specifying only a few layers for the new mask generation layer, the mask generation cost is saved. The basic layers are frozen and cannot be changed. Change all or part of the metal layer to achieve the required function of ECO. The reasons for the ECO and process stages remain basically the same; but there are some significant differences.

- Adding/Deleting Cells – You cannot add or delete any cells. Technically, if the base layer remains unchanged (for example, different TIE units), we can replace the unit. However, this may be difficult to control.
- Updating the connections – The net connections needs to be updated.
- Placement – No ECO placement is done, as there cannot be any addition of cells.
- Routing – Today’s tools can automatically identify the changed nets and route them.
- Freeze silicon ECO usually uses backup batteries in the design. These are the cells scattered throughout the design for their intended use in ECO. Please refer to my article on battery backup for information on inserting and placing a battery backup.

## **3.7 Addition of Spare Cells, Filler Cells & Metal Filling**

### **3.7.1 Spare Cell Addition**

Spare cells are just other units placed in the layout, and they are only expected to be needed in the future. In the long run, this means after removing the silicon tape from the tape by hand. After the silicon test is complete, the appearance needs to be changed. If the appearance requires some logic changes, just consider the logic gates where the appearance logic changes require logic changes, then we will use the main logic gate that has no interference with the bottom layer. Only the metal mask is changed so that only the metal mask regeneration is required for subsequent manufacturing.

A spare cell was added to the design to achieve long-term ECO changes. Add all the cells present in the library by

spreading the cells throughout the design, thereby adding spare cells to the design. Otherwise, the modules of various units will be included in the appearance of the selected location. After the backup battery is inserted, whenever the ECO process requires those batteries located in different parts of the chip, they will be connected by a layer of metal to avoid interfering with other metal layers.

#### ***3.7.1.1 Spare cell Placement***

We need to consider where to place the spare unit in the layout. They are not important in time. If we do not provide any restrictions, the PnR tool will put them together. We do not know which layout area is needed to connect the spare parts. Therefore, we do spread the cells over the entire area of the chip.

### **3.7.2 Filler additions**

After completing the layout and routing, there are usually some blanks left in the layout, in which we do not have any standard cells. It is not possible to adjoin each available unit, as this will cause routing problems due to high congestion and will give us a poor layout in terms of timing. So if we say you have 70% utilization, then you can expect 30% of unfilled areas. If we perform a DRC check now (in a tool that can provide you with a base layer DRC), we may see a gap violation, such as "The NWell minimum gap has not been reached." This is where the filling unit enters. To get a neat layout, we need NWell continuity. It is imperative that our wellbore or moat spacing is not less than the minimum spacing (even if we can separate them with a spacing greater than the minimum spacing, it can be tricky in a row of standard cells).

In addition, in the tap-free library, we need fillers to continue the killing operation, even if we separate them at a distance greater than the minimum well spacing, unless each such harness has its own contact unit, the power supply is connected. There will be continuity errors. ) The filled unit is a unit with no logic function, but it continues the basic layer like NWELL, and the VDD / VSS pins match the remaining standard units. In the standard cell library, we will have a filler that has the smallest tile width and other dimensions. If our placement follows the placement grid, we can fill any gaps.

To simplify the generation of masks, this is one of the limitations of Fab. Generally, padding units are added after routing and timing closure and before LVS and DRC.

### **3.7.3 Metal Filling**

Whether performed by the designer or by the foundry, metal fill is a mandatory step at advanced nodes to ensure manufacturability and high yield. It involves filling the empty or white spaces near the design with metal polygons to ensure regular planarization of the wafer. In this later stage, the main goal is lithography requirements, while ignoring the effect of interconnect capacitance. Also, the padding added while doing planning process must follow effective rules to avoid disturbing time. Geometry sizes of 90 nm and below are currently generally unacceptable. Virtual metal fillers must be placed near the signal network to meet the minimal density and will affect overall performance to a greater extent. Also, the design has multiple voltage regions, so it is more difficult to encode the region-based fill in the physical verification tool.



### **3.8 Conclusion**

This chapter will introduce the steps involved in the physical design process. The steps covered include floor plan, layout, clock tree synthesis, routing, adding spare units, filling units, and metal filling. Things to consider when performing these steps. The above also introduces the concept of ECO changes and "how to make ECO changes".

## Chapter

# 4. CONCLUSION AND FUTURE SCOPE

## 4.1 Conclusion

This work proposes a method that can effectively develop and manage power intent at the SoC level by evaluating various shortcomings of the normal synthesis method. By introducing the adjustment process before developing the top-level UPF, the problem of inconsistency of power-related data that may be caused by the parallel development of powerful intentions is solved. Various power reduction techniques are introduced to solve the power-related problems in the conventional synthesis process, and the modified process is considered to obtain the best results. The design is more efficient in placement and routing and provides better timing reports for the "normally synthesized" flow netlist. The results obtained by applying the proposed flow to a complex design including memory chip prove that the proposed method can accelerate and simplify the UPF process at an early stage and throughout the design cycle.

## 4.2 Future Scope

The UPF standard (IEEE 1801) is updated regularly to represent the complex power architecture of the design as precise as possible. The new standards will take the advantage of advance feature. The UPF format is classified based on tools to achieve effective power intent interface between tools from different vendors. The impact of complex power intents on UPF standards should be checked and properly analyzed. It can be achieved by increasing power domains and by reducing technology. As a power domain, power states are written manually in UPF, and their automation can greatly reduce time. The bottom-up method of the

UPF method will be analyzed and optimized because not all design processes are top-down based processes.

## References

- [1] Moore, Gordon “Moore’s Law”, Intel company, 1965, 11 July 2016.  
Available: [https://en.wikipedia.org/wiki/Moore%27s\\_law](https://en.wikipedia.org/wiki/Moore%27s_law)
- [2] R. K. Krishnamurthy, Atila Alvandpour, Vivek De, Shekhar Borkar, “High-performance and low-power challenges for sub-70 nm microprocessor circuits” *CICC, 2002*.IEEE Proceedings, pp. 125–128, 2002
- [3] Neil Weste and David Harris, “CMOS VLSI Design: A Circuits and Systems Perspective”, Addison-Wesley Publishing Company, 4th edition, 2005.
- [4] Chadha, Rakesh, J. Bhasker, “An ASIC Low power prime analyser”, Springer Company, 1st edition, 2013
- [5] Nam Sung Kim, Todd Austin, David Blaauw, Trevor Mudge, Krisztian Flautner, Jie S.Hu, Mary Jane Irwin, Mahmut Kandemir, Vijaykrishnan Narayanan, “Leakage current: Moore’s Law Meets Static Power”, *IEEE computer society*, 2003.
- [6] Chi-Ping Hsu, “Pushing Power Forward with a Common Power Format - The Process of Getting it Right”, EETimes, 5 Nov 2006.
- [7] “A Practical Guide to Low-Power Design: User Experience with CPF”, Cadence, March 2008.
- [8] Richard Goering (September 18, 2006), "Accellera rolls power plan", EE Times, Retrieved July 7, 2011.
- [9] Qi Wang, “The Evolution of Power Format Standards: A Cadence Viewpoint”, Cadence Design, Systems, Inc, 2012.
- [10] Erich Marschner, “Evolution of UPF: Getting Better All the Time”, October, 2012. Available: <https://verificationacademy.com/verification-horizons/october-2012-volume-8-issue-3/evolution-of-upf-getting-better-all-time>
- [11] Erich Marschner, “Overview of UPF”, Mentor Graphics. Available: <https://verificationacademy.com/sessions/overview-of-upf>

- [12] "IEEE P1801 - Unified Power Format Standard", *Acclera Organization Inc*, February 22, 2007.
- [13] "VLSI Concepts". *Vlsi-expert.com*. N.p., 2016. Web. 31 May 2016.
- [14] Himanshu Bhatt, Harsh Chilwal, "Golden UPF: Preserving Power Intent from RTL to Implementation", *DVCon*, March
- [15] Venkatesh Gourisetty, Hamid Mahmoodi, Vazgen Melikyan, Eduard Babayan, Rich Goldman, Katie, "Low Power Design Flow Based on Unified Power Format and Synopsys Tool Chain", *3rd Interdisciplinary Engineering Design Education Conference, IEEE*, pp.28-31,4-5 March, 2013
- [16] "Flip chip c4b", Mantra VLSI, 2015. Available: <http://www.slideshare.net/DeepakFloria/flip-chip-technology>
- [17] "IEEE Standard for Design and Verification of Low Power Integrated Circuits", *IEEE Computer Society*, 19th March 2009.
- [18] Cadence, Genus User Guide, [www.supportcadence.com](http://www.supportcadence.com).
- [19] Batchu sri sai Chaitanya , "Synthesis", Signoff Semiconductors, Oct 17 2017. Available : <http://www.signoffsemi.com/synthesis/>
- [20] Rob Utenberg , "VLSI CAD Layout", Course Era.
- [21] Low Power VLSI Design and Implementation" Available: <https://asic-soc.blogspot.com/p/low-power-vlsi.html>
- [22] Sumeet Anwikar, "Introduction to SDC", Signoff Semiconductors, Nov 25 2019. Available :[http://www.signoffsemi.com/introduction\\_to\\_sdc/](http://www.signoffsemi.com/introduction_to_sdc/)
- [23] "VC Low Power User Guide", Synopsys, June.2016
- [24] Padmanabhan, Uday, Janet Meiling Wang, and Jiang Hu. "Robust clock tree routing in the presence of process variations." *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 27.8 (2008): 1385-1397.
- [25] "IEEE Standard for Design and Verification of Low-Power Integrated Circuits", *IEEE Computer Society*, 29th May 2013.

- [26] Freddy Bembaron, Sachin Kakkar, Rudra Mukherjee, Amit Srivastava, “Low Power Verification Methodology Using UPF”, *DVCon 2011*, Feb 28-Mar 3, 2011.
- [27] Yongseok Cheon, et al. “Power-Aware Placement.” Proceeding of 42nd Annual design Automation Conference. ACM,2005..
- [28] Rudra Mukherjee, Amit Srivastava, Stephen Bailey, “Static and Formal Verification of Power Aware Designs at the RTL Using UPF”, Mentor Graphics Corporation, 2009.
- [29] “VC LP Message Reference Guide”, Synopsys, June 2016.
- [30] Lu, Bing, et al. "Process variation aware clock tree routing." Proceedings of the 2003 international symposium on Physical design. ACM, 2003.
- [31] Desinghu PS, Adnan Khan, Erich Marschner, Gabriel Chidolue, “Refining Successive Refinement: Improving a Methodology for Incremental Specification of Power Intent”, *DVCON 2015*, March 2015.
- [32] “Synopsys® Low-Power Flow User Guide”,Synopsys, December 2011.
- [33] Emilie Garat, David Coriat, Edith Beigne, Leandro Stefanazzi, “Unified Power Format (UPF) methodology in a vendor independent flow”, *PATMOS,2015, 25TH International Workshop, IEEE*, pp.82-88, 1-4 September, 2015.
- [34] Lu, Bing, et al. "Process variation aware clock tree routing." Proceedings of the 2003 international symposium on Physical design. ACM, 2003.