PHYSICAL DESIGN OF FUNCTION UNIT BLOCK USING RANDOM LOGIC SYNTHESIS METHODOLOGY

Major Project

Submitted in Fulfilment of the Requirements
for the Degree of
Master of Technology
in
VLSI Design
by

BrijeshKumar Savaj
12MECV05

Department of Electrical Engineering
Electronics & Communication Programme
Institute of Technology
Nirma University
Ahmedabad
May-2014

PHYSICAL DESIGN OF FUNCTION UNIT BLOCK USING RANDOM LOGIC SYNTHESIS METHODOLOGY

Major Project
Submitted in Fulfilment of the Requirements
for the Degree of
Master of Technology
in
VLSI Design
by

BrijeshKumar Savaj
12MECV05

Dr. N M Devashrayee
Internal Guide

Mr. Murali S Seshadri
Engineering Manager

Department of Electrical Engineering
Electronics & Communication Programme
Institute of Technology
Nirma University
Ahmedabad
May-2014

Declaration

This is to certify that

1. I, BrijeshKumar Savaj, a student of Master of Technology in VLSI Design, Nirma University, Ahmedabad hereby declare that the project work "Physical Design Of Functional Unit Block Using Random Logic Synthesis Methodology" has been independently carried out by me under the guidance of Mr. Murali Seshadri, Engineering Manager, Intel Technology India Private Limited, Bangalore and Dr. N M Devashrayee,PG Co-ordinator, Department of M.tech(VLSI Design), Nirma University, Ahmedabad. I would also like to give a vote of thanks to Dr. N. M. Devashrayee for providing valuable support for project work. This Project report has been submitted in fulfillment of the requirements for the award of degree Master of Technology(M.Tech.) in VLSI Design, Nirma University, Ahmedabad during the year 2013 - 2014.

2. I have not submitted this work in full or part to any other University or Institution for the award of any other degree.

BrijeshKumar Savaj
12MECV05

Certificate

INTEL Technologies India Pvt. Ltd.

This is to certify that Mr. Brijeshkumar Savaj, a student of M.tech EC (VLSI Design), Institute of Technology, Nirma University was working in this organization since 05/06/2013 and carried out his thesis work titled "PHYSICAL DESIGN OF FUNCTIONAL UNIT BLOCK USING RANDOM LOGIC SYNTHESIS METHODOLOGY". He was working in RLS team under the supervision of Mr. Murali S Seshadri, Engineering Manager. In my opinion, the submitted work has reached a level required for being accepted for examination. He has successfully completed the assigned work and is allowed to submit his dissertation report.We wish him all the success in future.

Dr. N M Devashrayee
Internal Project Guide
Institute of Technology,
Nirma University, Ahmedabad

Mr. Murali Seshadri
Engineering Manager
Intel Technologies India Pvt. Ltd.,
Bangalore

Dr. K Kotecha
Director
Institute of Technology,
Nirma University, Ahmedabad

Dr. P. N. Tekwani
Head of EE Dept
Institute of Technology,
Nirma University, Ahmedabad

Date:15/05/2014

Place:Ahmedabad

Abstract

Server Chip is demanded more as the large use of social networking and the numbers of users are increases day by day. Intel is still covers 90 % segment of the server chip market. I wrote this the with the experience of working with good server chip design team at Intel. The project on which I worked is a very high server SoC chip which is designed using 14nm technology.

This thesis is focused on the method which is used for backend design. The word physical design refers to the conversion of RTL code (Which is coded in Hardware Description Language to implement some functionality) in to Graphic Database System (GDS) format file (Which is used for manufacturing process) by satisfying several constraints like area, power, speed and performance. However these are different ways for this purpose, but the demand to implement most of the functionality in a single chip increases the complexity and hence an efficient method to design such complex chip is required.

RLS (Random Logic Synthesis) is one of the techniques which are very much useful for such a complex design. The competition in the market is huge and so TTM (Time To Market) plays an important role for winning the market, so to meet this TTM and produce the chip in very effective way there is a specific synthesis approach which Intel follows. And this method is a key part of that process.

RLS flow includes logic synthesis, physical synthesis, timing analysis, timing optimization, FEV (Formal Equivalence verification),noise analysis and manufacturability (DRC/DFM) ready. In the design of high speed microprocessor, meeting the timing requirement is critical. Timing has to meet both setup as well as hold time for the design in GHz frequency space. Later Formal Equivalence Verification is done on the post-synthesis, post physical-synthesis and routed database with RTL to check the formal equivalence. The main responsibility of RLS team is to synthesize the code given by RTL owners to physical level by reducing the setup and hold violations that are produced during the various stages.

This project report is mainly focus on the basic RLS design flow and some techniques use for each steps of the flow to meet the constraints of the design. I have tried to include more about the limitation and more advantages of this methodology in this thesis.

Acknowledgement

First and foremost, sincere thanks to Mr. Murali Seshadri, Engineering Manager, Intel Technology India Private Limited, Bangalore. He assigned me this project from which i learnt a lot. I owe him lots of gratitude for having a profound impact on this report.

I would like to thank my Mentor, Mr. Raghavendra K , Design Engineer, Intel Technology India Private Limited, Bangalore for his valuable guidance. Throughout the training, He has given me much valuable advice on project work which I am very lucky to benefit from. Without him, this project work would never have been completed.

I would also like to thank Dr. K.R.Kotecha, Director, Institute of Technology, Nirma University, Ahmedabad for providing me an opportunity to get an internship at Intel Technology India Private Limited, Bangalore.

I would thank to my Project Co-ordinator, Dr. N.M. Devashrayee, Professor, VLSI Design, Institute of Technology, Nirma University, Ahmedabad for giving valuable support for project work.

I would also like to thank my internal guide Dr. N M Devashrayee for his continuous guidance to make my project and presentation effective.

I would like to thank my all faculty members for providing encouragement, exchanging knowledge during my post-graduate program.

I also owe my colleagues in the Intel, special thanks for helping me on this path and for making project at Intel more enjoyable.And at last i would also like to take this opportunity to thank to my family members for giving me such a great support throughout the project work.

BrijeshKumar Savaj
12MECV05

Brief Content

Physical Design of Function Unit Block using Random Logic Synthesis methodology is the title of this report. This report mainly focus on some basic theories of the physical design and RLS methodology which is I used to understand the method. The theory is mainly gathered from the internal material of Intel and I have tried to written in my own words.

The report is logically divided in two parts. The first part contains some introduction things having the overall idea of where this method is applied during the chip design process. And the latter part contains some of the techniques which are used to understand the problems occur during the design process and how to solve those problems. As I worked as an intern at Intel Corporation, I could not include most of the process and some confidential content. In this thesis I have tried to include some more information which may give more idea about how this method is useful but at the same time in order to not violate any corporate rule I have presented the results just for reference only.

Contents

Declaration iii

Certificate iv

Abstract v

Acknowledgement vi

Brief Content vii

1 Introduction 1
- 1.1 Area of Work . . . 1
- 1.2 Motivation . . . 1
- 1.3 Aim of the Project . . . 2
- 1.4 Internship Organizer . . . 3
- 1.5 About rest of the report . . . 3

2 Design styles used in the CPU design 4
- 2.1 Programmable logic devices . . . 5
  - 2.1.1 Field Programmable Gate Array(FPGA) . . . 5
  - 2.1.2 Gate array design . . . 7
- 2.2 Standard cell based design . . . 7
- 2.3 Full Custom design . . . 8

3 Random Logic Synthesis Flow 10
- 3.1 Different steps in the RLS flow . . . 11
- 3.2 Logic Synthesis . . . 14
- 3.3 Floorplanning . . . 17
- 3.4 Placement . . . 17
- 3.5 Clock Tree Synthesis . . . 19
- 3.6 Routing . . . 20
- 3.7 Chip Finishing flow . . . 21

4 Timing Analysis and optimization **23**
4.1 Timing . . . 24
4.1.1 Internal timing . . . 24
4.1.2 External Timing . . . 24
4.2 Violations . . . 25
4.2.1 Setup or max violation . . . 25
4.2.2 Hold of min violation . . . 26
4.3 Timing Optimization methodology . . . 28
4.3.1 Timing Exceptions . . . 28
4.3.2 Path Grouping . . . 29
4.3.3 Optimizing paths with positive margin . . . 30
4.3.4 Timing Optimization for external Paths . . . 31
4.3.5 Bind2ports . . . 32
4.3.6 Timing Checks . . . 33
4.4 Results of timing optimization . . . 34

5 Miscellaneous Optimizations **36**
5.1 Placement Optimization . . . 37
5.2 Power Optimization . . . 40
5.2.1 Results of Power Optimization . . . 41
5.3 Noise Convergence . . . 42
5.3.1 Noise Violations Summary . . . 43

6 Tools and technology used **44**
6.1 Design Compiler (DC) . . . 44
6.2 IC Complier (ICC) . . . 46

7 Conclusion and Future scope **49**
7.1 Conclusion . . . 49
7.2 Future Scope . . . 49

List of Figures

2.1 Chip design flow 4
2.2 Block diagram of FPGA 5
2.3 CLB overview 6
2.4 Programmable switch 6
2.5 Gate array design process 7

3.1 Chip hierarchy 10
3.2 RLS flow 12
3.3 Example 15
3.4 Synthesis 16
3.5 Floorplan 18
3.6 Placement 18
3.7 Before clock tree synthesis 19
3.8 After clock tree synthesis 20
3.9 Steps of routing 21

4.1 External input delay 25
4.2 External output delay 25
4.3 Paths between two flip-flops 26
4.4 Timing Diagram for setup check 26
4.5 Typical case of hold violation 27
4.6 Timing Diagram for a hold check 27
4.7 Multicycle timing relationship 28
4.8 An example of Path Grouping 29
4.9 Equation 1 30
4.10 Equation 2 30
4.11 Timing constraints for external paths 31

5.1 Congetion and short snapshot 38
5.2 Final Placement snapshot 39
5.3 Latch based Gated Clock 41

6.1 Desogn Compiler overview 45
6.2 ICC overview 47

List of Tables

1.1 Time management of internship 3

4.1 Timing statistics at Initial Stage 34
4.2 Timing statistics at Intermediate Stage 35
4.3 Timing statistics at Final Stage 35

5.1 List of DRC's without ICCDP flow 37
5.2 Without Power optimization techniques 42
5.3 With Power optimization techniques 42
5.4 Noise Violations 43

Chapter 1

Introduction

This chapter will give the basic understanding of the flow of how chip is manufactured. Also gives the area of my work at Intel Corporation, the motivation behind this method, the objectives of this methodology and my schedule of internship at Intel.

1.1 Area of Work

Server Development Group (SDG) of Intel India is involved in designing and manufacturing high end multi-core servers which is primarily used in datacenter business across the world. Today most of the worlds datacenter is being powered up by Intels high performance Xeon servers. As technology advances, the design is becoming more and more complex and also the consideration for low power is also becoming more significant.

There are different teams in the same organization; all are responsible for different works. I belong to the design team called Micro-processor Design Group India RLS. There are different teams which are mainly for tool maintenance and modification as well as noise team, power team, special circuit team etc. The whole chip is divided in to this different team according to their function and at the end the integration team and validation team together end up with the database having all constraints satisfied and ready to go in the fabrication lab for manufacturing. Once the sample is ready then it will come in the testing lab and power on will happen, and if any bugs are found in this then re-synthesis will happen and bug will be fixed. After this one cycle this process continues until the product quality is not achieved then chips comes out in market for selling.

1.2 Motivation

In todays era of high volume of data , need of high speed performance as the data traffic over the network is very huge and so the requirement of such a high performance

device within the very less time to market frame we have to use the methodology which helps us really to meet all our requirements. And so the Random Logic Synthesis (RLS) methodology we choose for the microprocessor design is very suitable and it has the advantages of Low TTM and high performance design. As we know it is completely tool based methodology, one most important thing with this design stile is that "Best the constraints you provide, Tool will produce the Most optimized design."

1.3 Aim of the Project

Meet timing requirement. In a microprocessor design, typically there is a need to communicate over long interconnects. Since these long interconnects cause huge RC delay there is requirement of repeater units to improve delay and to increase the drive strength of this interface in order to meet timing constraints. There are huge number of path initially in the design which do not meets the timing constraints so to achieve the small time to market these type of timing paths are fixed using some tool automation in RLS methodology.

Overcome the Noise Violations. Noise can be defined as any deviation of an evaluation node voltage from nominal high or low values as determined by the logic. With the continuous scaling of CMOS digital technologies signal integrity and noise issues have become a metric of comparable importance to area, timing, and power for deep submicron digital integrated circuits. In these design different types of noises violations originating from different sources were seen. Few techniques were implemented to reduce those violations.

Integrated circuits (ICs) on silicon chips are constructed out of two fundamental components transistors and interconnect. These components experience stress, wears out over time and eventually fails .So reliability verification needs to be performed to detect potential failure sites and to ensure adequate lifetime.

In the VLSI design apart from operating frequency, reliability, noise etc. power is also a very important design parameter. Since cordless handheld computing is booming like anything, examples being laptop, smart-phones, palmtops, notebooks etc., one of the design targets is power in order to increase battery life otherwise which is of no use. Thats one of the motivations to make processor power efficient. In the design flow, we do power optimization, the idea of this experiment is to check whether there is any loose end in the process itself or not. That means checking whether we are over optimizing the design or putting redundant constraints and thus losing power benefits. Most importantly all of these constraints do not have very strong and clear base. Most of them were decided long back and scaled up or down with process shift. We targeted some such constraints

and checked the impact of changing them. In this design we have targeted the leakage power and dynamic power while ensuring the performance.

1.4 Internship Organizer

Table 1.1: Time management of internship

Task assigned	**Time frame**
Initial Ramp up and getting access to project	2 months
Learning UNIX and scripting language	2 months
Gathering the specification and setting the Environment for my design (FUB)	1 month
Synthesis of my design with a goal to meet the timing requirements	3 months
Verification of all the FUBs with respect to project requirements (Power, External timing, Timing Violations, LVS)	2 months
Review of the FUBs with the reviewers.	1 month

1.5 About rest of the report

The remaining part of the thesis is categorized in the following manner.
Chapter 2 describes the different design styles used in the industry for the chip design.
Chapter 3 will gives the detailed view of the RLS methodology. And it also contained description of each and every stage of this design style in well-organized way.
Chapter 4 will gives idea of timing problem faced while designing the high frequency block and how can we overcome in RLS methodology.
Chapter 5 will gives understanding of other challenges related to Noise, Power and Area while designing the high frequency block and the solutions of that.
Chapter 6 will describes the tools used in this method and only brief idea of the technology.
Chapter 7 contains conclusion and future scope of this project.

Chapter 2

Design styles used in the CPU design

Before going to any design style it is very much needed to see where we are going to use this type of design styles in the chip manufacturing process. So the figure 2.1 will gives you idea about the different steps which are there in the design process of chip.

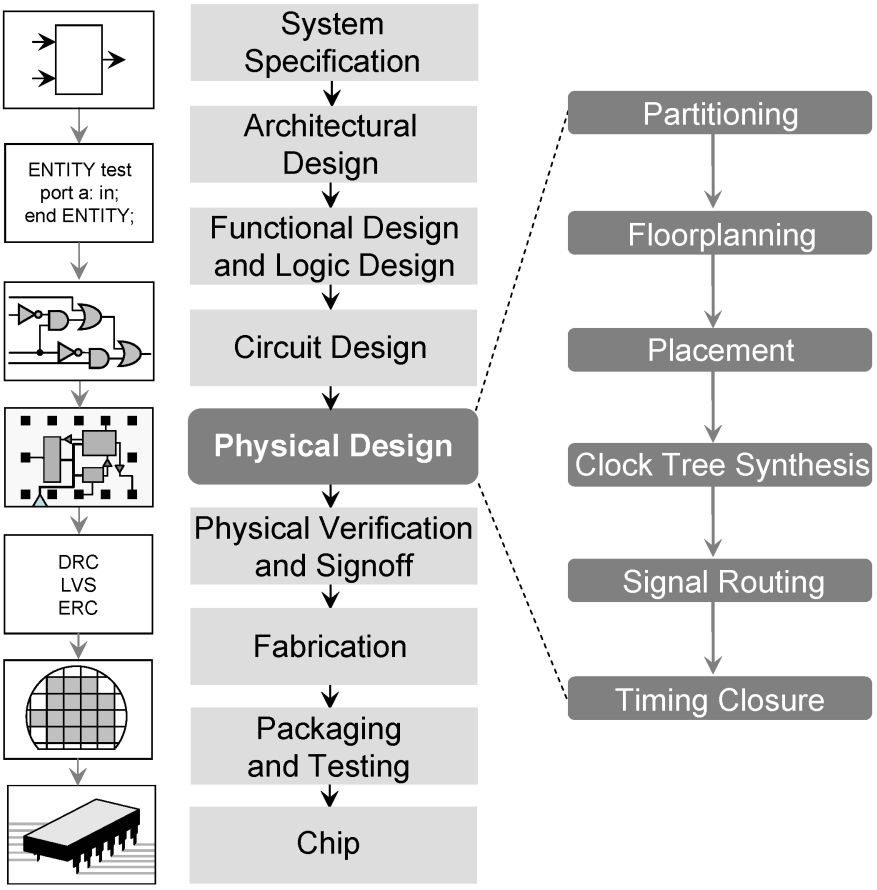

Figure 2.1: Chip design flow

In the above figure 2.1 it is clearly mentioned each and every steps with some graphical view which clears well the idea of the process. From the system specification to circuit design step called Front-End design and from there to physical verification and sign-off stage it is referred to as Back-End design. And in between there are verification steps at almost each and every step to ensure the design implemented is correctly match with the previous step.

There are three different categories of design style which are most widely used in the industry and all have some different advantages and disadvantages.
Three types:-

- Programmable logic devices
  - Field programmable gate array (FPGA)
  - Gate aaray design
- Standard cell based design
- Full custom design

2.1 Programmable logic devices

2.1.1 Field Programmable Gate Array(FPGA)

In this design style there are well organized pattern of logic gates in the prefabricated chip. It includes input/output buffers, configurable logic blocks (CLB), and programmable interconnects. It do not require process step for any logic realization.

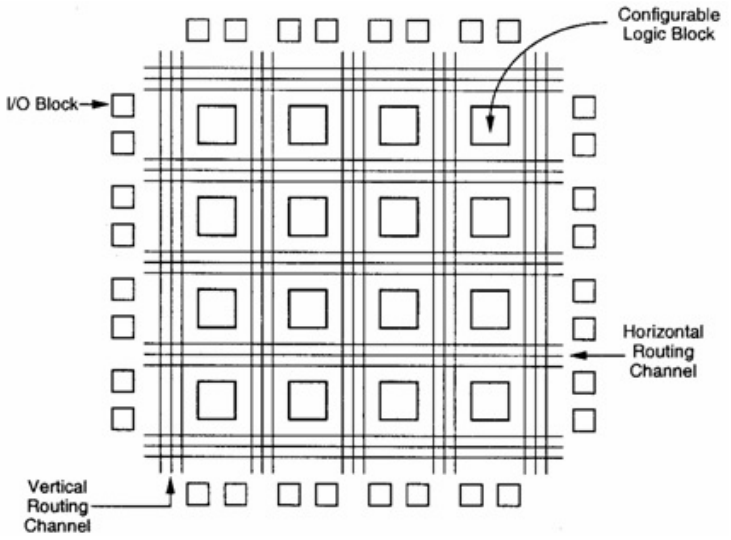

Figure 2.2: Block diagram of FPGA

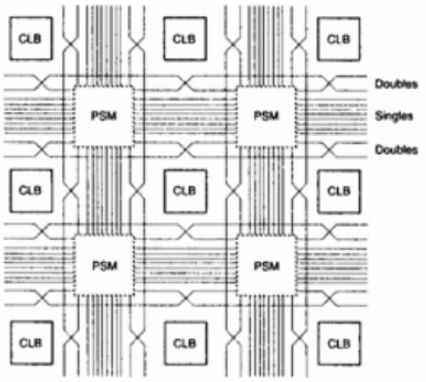

Figure 2.3: CLB overview

Configurable logic blocks contains independent combinational function generators which are like memory look up tables, clock signal terminals, user programmable multiplexers, and flip-flops.

Programmable interconnects are consist of six pass transistors per one switch matrix interconnect point. And all this points are accomplished by data in RAM cells.

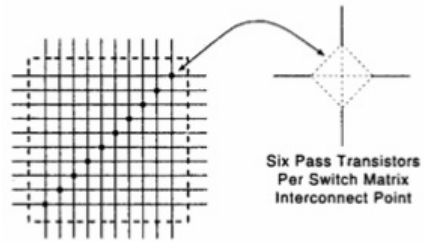

Figure 2.4: Programmable switch

Its Design flow is as below:-

- Behavioral description of its functionality In this step the functionality is coded in hardware description language using some tool which may be provided by the FPGA vendor itself.
- Technology mapped into circuits or logic cells In this step the HDL code is converted in to different logic cells connectivity using the tool given by vendor which is specifically tailored for that vendor FPGAs only.
- Assigns logic cells to FPGA CLBs and determines the routing pattern. In the last the tool will do the assignment of different logic cells among different CLBs and try to satisfy the constraints of power and timings. Then it does the routing and after that the FPGA is configured with the file provided by the software tool.

It is most suitable for the fast prototyping and small volume ASIC production so it has small turn-around time.

2.1.2 Gate array design

In this design style the layers of uncommitted transistors which are separated by routing channels. It is used in two phase, the first phase there are generic masks for uncommitted transistors are stored on each GA chips. in the second phase these all uncommitted transistors are connected according to functionality by multiple times metal fabrication process.

It ranks second after the FPGA in terms of the turn-around time of a few days. Chip utilization factor is higher than that of the FPGA.

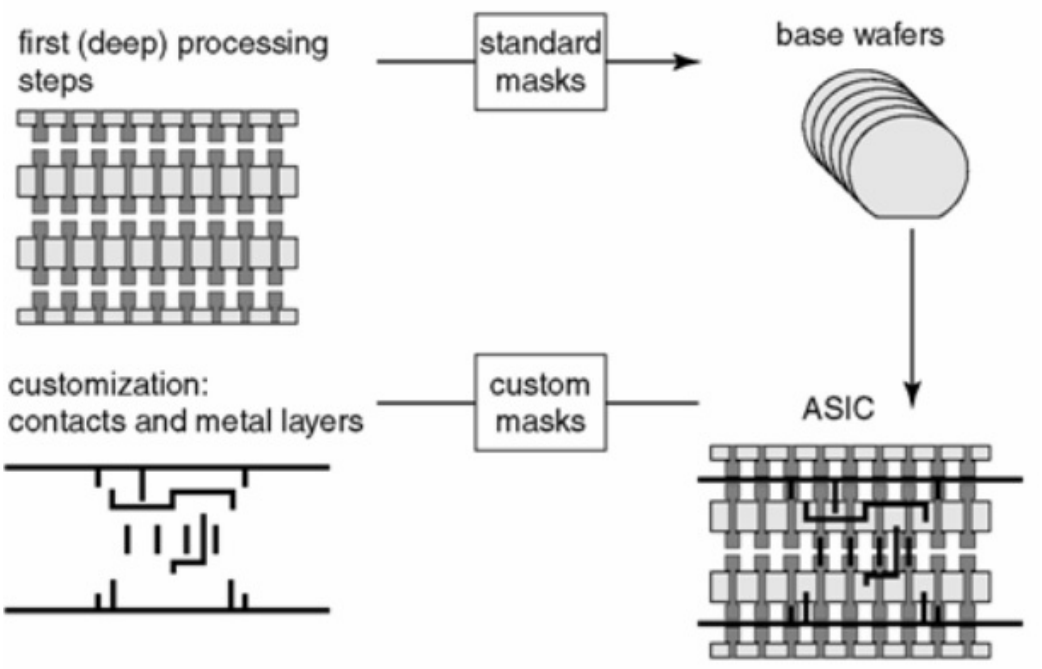

Figure 2.5: Gate array design process

2.2 Standard cell based design

This design style is very popular now a day. In this style the commonly used logics are developed, characterized and stored in a standard library. The required logic circuits are realized using the cells in the library. Complete mask sets are developed for chip fabrication. It is one of the most prevalent design style for ASIC applications.

- Cell library includes:-
  - Delay time versus load capacitance
  - Circuit simulation model
  - Timing simulation model

  - Fault simulation model
  - Cell data for place-and-route
  - Mask data

- Standard cell arrangement:-
  - Fixed cell height
  - Parallel power and ground rails
  - Input and output pins are located on the upper and lower boundaries
  - Cells are placed side by side in standard-cell based design

2.3 Full Custom design

This is the design style which is used for the high performance and high accuracy design style. In this design style the whole design is developed from the scratch. Using this method it is possible to achieve the highest performance compared with other design styles, but it has highest development cost and more design cycle time. But it has been some sort of improvement in this style is that design reuse. It is coming popular to alleviate the design efforts. It is mostly suitable for the high volume production and high performance design.

The design style used in this project is somewhat called semi-custom design style and the standard cell based design style. There are several sections and design we use from the previous project and some we design newly and for my part I used standard cell based methodology. But the methodology which we are using at Intel does not call standard cell methodology. In general, the physical design methodology for digital circuits on our project can be divided into three major design styles.

1. SDP (Structured Data Path)
2. RLS (RTL-to-Layout Synthesis)
3. RFA (Register Files Automation)

- Structured Data Path (SDP) for timing critical portions of the design, for which the designers want complete control over the circuits. SDP methodology assumes that the designer draws the schematic by hand and instructs a CAD tool on how the cells should be placed in the layout. The cells are then routed using an automated tool (the manual pre-routes are done beforehand for the critical wires)

- RTL-to-Layout Synthesis (RLS) for large portions of logic that is less timing critical. The schematic, placement and routing of cells is completely automated, and the schematic and layout are generated by the synthesis tools from the RTL, based on timing constraints and the area available for the given portion of the chip. The control over the schematic and placement is relatively low compared to the SDP style, but as long as speed and power requirements of the circuit are satisfied
- Register Files Automation (RFA) for the design of the register files i.e. structured arrays of registers used on the chip for wide range of purposes. The goal of this flow is to automate the creation of these highly structured arrays as much as possible, to cover many different types such as RAM, CAM, with multiple read and write ports, with and without decoders, etc. The RFA flow as such is typically not used by many designers on the project.

So till now I gave you the complete methodology overview in general and specific to Intel also. Now onwards the main focus is on the RLS methodology as the thesis is completely based on that method. In the next chapter the complete methodology is described.

Chapter 3

Random Logic Synthesis Flow

This chapter is more focused on the main method which I used for my physical design. The method is named in multiple ways as this is Intel internal terminology, RLS stands for Random Logic Synthesis and RTL To- Layout synthesis also. The RLS flow is a Intel specific flow which is used in the backend chip design process in order to convert the RTL in to manufacturing ready layout file by meeting the tight high frequency constraints.

Before going to the flow itself I want to show(figure 3.1) the chip hierarchy for the chip which gives you idea about how the chip is divided and where I was working.

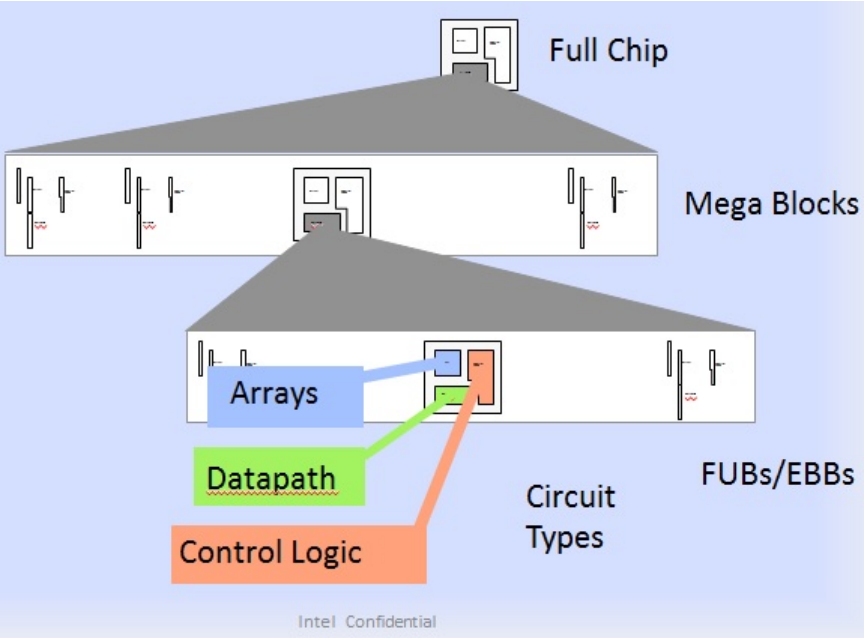

Figure 3.1: Chip hierarchy

- Arrays are basically some logic gates array not programmable but designed according to requirement.

- Data paths are nothing but the logic signals which are timing critical and mostly responsible for the functionality.
- The control logic will have the control over all the functionality, and it mainly decides which signal needs to be generated based on some control signal.
- FUBs/EBBs are nothing but the smallest hierarchy of the design which individually carried out by design engineer and converge the design.
- FUB stand for Functional Unit Block.
- EBB stands for Embedded Black Box.

Now the rest of the section of this chapter the flow will be discussed.

3.1 Different steps in the RLS flow

The diagram shown in figure 3.2 shows the most important steps in the RLS design flow and in between this steps there is always some verification is done to make sure the output of the step is perfectly matched with the input of that step.

However each step has lots of things to explain and lots of techniques to discuss but the scope of the document is limited and will give only the overview of each steps.

The RLS flow is nothing but the use of different tools for the accomplishment of one particular design with constraints. So below is some requirement for the flow to start the synthesis process.

The major inputs required for the implementation of any functional unit block are the following.
- RTL code
- Timing Constraints
- Floorplan Collaterals

- RTL
  Register Transfer Logic (RTL) code is nothing but a hardware description language which describes the functionality to be implemented. In this project the hardware description language used is system Verilog. RTL code is written with the help of various macros which corresponds to a particular flop or latch or any other combinational device. Usually the RTL code for all the FUBs are archived in a central area and it can be accessed from that area for implementation.

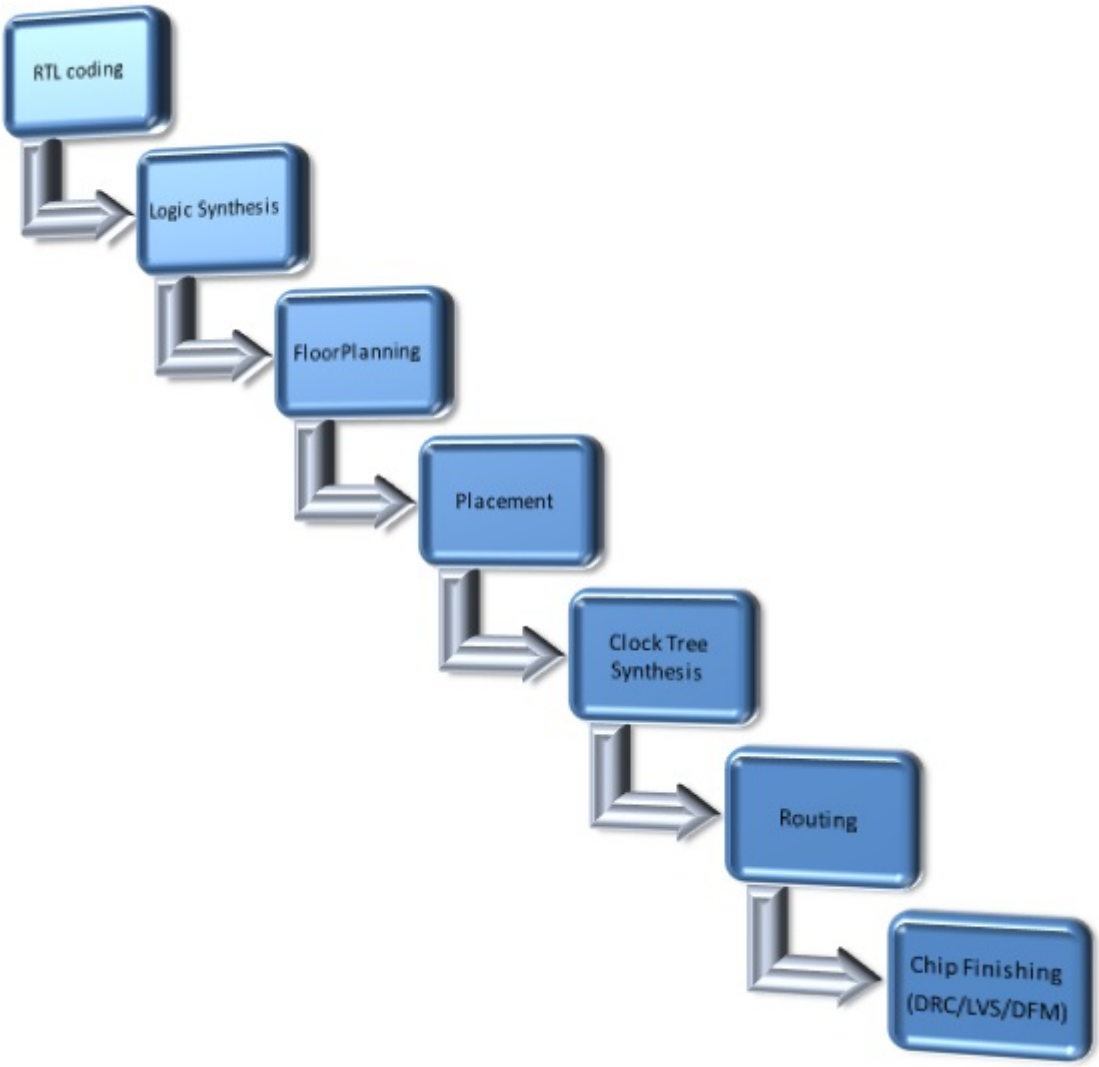

Figure 3.2: RLS flow

- Timing Constraints
  Timing constraints are the external constraints that are applying on particular design in order to achieve the required results. The design must be implemented by meeting all the timing constraints. Usually it contains information like setup time, how much time the particular design can take to accomplish a particular task, slope information, Effective capacitance, Total capacitance, Effective resistance etc. While physically implementing the design it has to meet all the slope, capacitance and timing constraints.

- Floorplan collaterals Floorplan collaterals are one of the integral inputs for the physical implementation of these designs. It is required for the floorplanning. Floorplan collaterals are generated by the person who owns the layout for that particular section. These collaterals contain various files which comprised of all the information required for floorplan. It contains information regarding input and output port locations, its corresponding metal layer to be used, grid clock location and its corresponding metal layers, keep out regions which cannot be used for placement as well as routing.

For generating the collaterals the layout owners have to place the input and output pins on the layout. Generally these pins are on the two sides of the FUB. In order to obtain better we are putting a constraint that the input and output should be aligned on both sides of the design. This can be achieved through some automation. The layout owners have to create the pins on one side and pins on the opposite side can be automatically generated with some automation scripts with the help of map files. With the help of map files we can replicate the corresponding pins on the other sides if the pins are placed/routed on the other side. So for the generation of floorplan collaterals generation of map file is an inevitable part. Some twenty percent of this project is about the generation of map files for all the designs centrally so that all the layout owners can access it generate the collaterals.

These are the different files which are provided by the layout team to the designer which depends on the FUB provided to designer.

- Floorplan.tcl
- Rkor.tcl
- Preroute.tcl
- Pkor.tcl
- Gclk.tcl
- Power.tcl

- Floorplan.tcl
  It contains all the pins/port names, its location, the direction of the pins/ports and the metal layer to be used for the corresponding pins/ports. This file will be used in the floorplan stage by the flow to create the ports in corresponding locations.

- Rkor.tcl
  This file gives the information regarding the routing keep out region. Based on this file all the locations mentioned in this file will be black boxed and it cannot be used for any kind of routing for the FUB.

- Preroute.tcl
  This file has some information regarding the section routes which are very critical and so routed by the section layout owner before giving to the designer for implementation. So while routing the designer cannot do any change in these routed tracks.

- Pkor.tcl
  This file contains information regarding the placement keep out region. The tool cannot place the library cells in the region mentioned in this file.

- Gclk.tcl
  This file contains the information regarding the grid clock. In floorplan the grid clock should overlap with design for extracting the clock inside the FUB. gclk file gives information about the length, width, orientation and location of the grid clock. This is information is used in the floorplan stage for implementing the grid clock.

- Power.tcl
  It contains all the information regarding the power. The ground VSS and supply VCC will be placed in the design according to the information based on this file. For level shifters there will be one more additional power which is the secondary power.

3.2 Logic Synthesis

The logic synthesis refers to the transformation of the behavioral description (HDL code) in to a structural description. It is also called design refinement which adds an additional level of details that provides information needed for the next level of manufacturing process of the design. During the synthesis process constraints are applied to ensure that the design meets the required functionality and speed. Only after the netlist generated through this synthesis is verified for the functionality and timing then it in used for the rest of the flow.

Synthesis process contains the following tasks:

- compilation and minimization

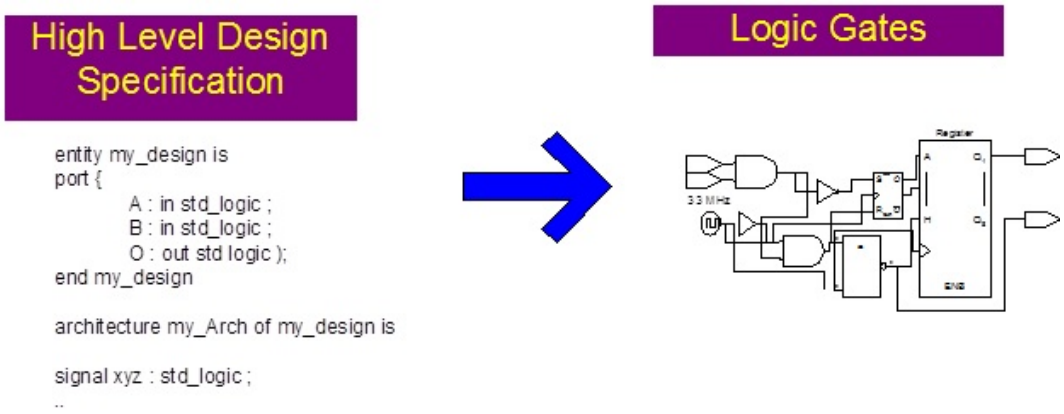

Figure 3.3: Example

- Technology mapping
- Optimization
- Transistor sizing

The figure 3.4 gives you the complete idea of synthesis.

Core cell Library is contains the information of all the cells which should be used by the tool to implements the design and it also have the information of size , power, and all timing related info as well as the characteristics of the cells.

The EB library has the information for the embedded blocks like RF and RAM and all such blocks which are predesigned and are need to integrate in the current design.

Wire load models are used to estimate the timing behavior of the pre-layout database so that we can get best optimized design when we actually route the design. And this WLM are function of fanout also so while synthesizing it is needed to choose cells with appropriate driving capabilities.

Designware components are technology independent, reusable intellectual property blocks (such as adders, comparators, fifos etc...) that are synthesized to gates from the target library. Due to this designer can use arithmetic operations without worrying about the hardware implementation details, DC (Design Compiler) will choose the best implementation based on speed/area tradeoffs.

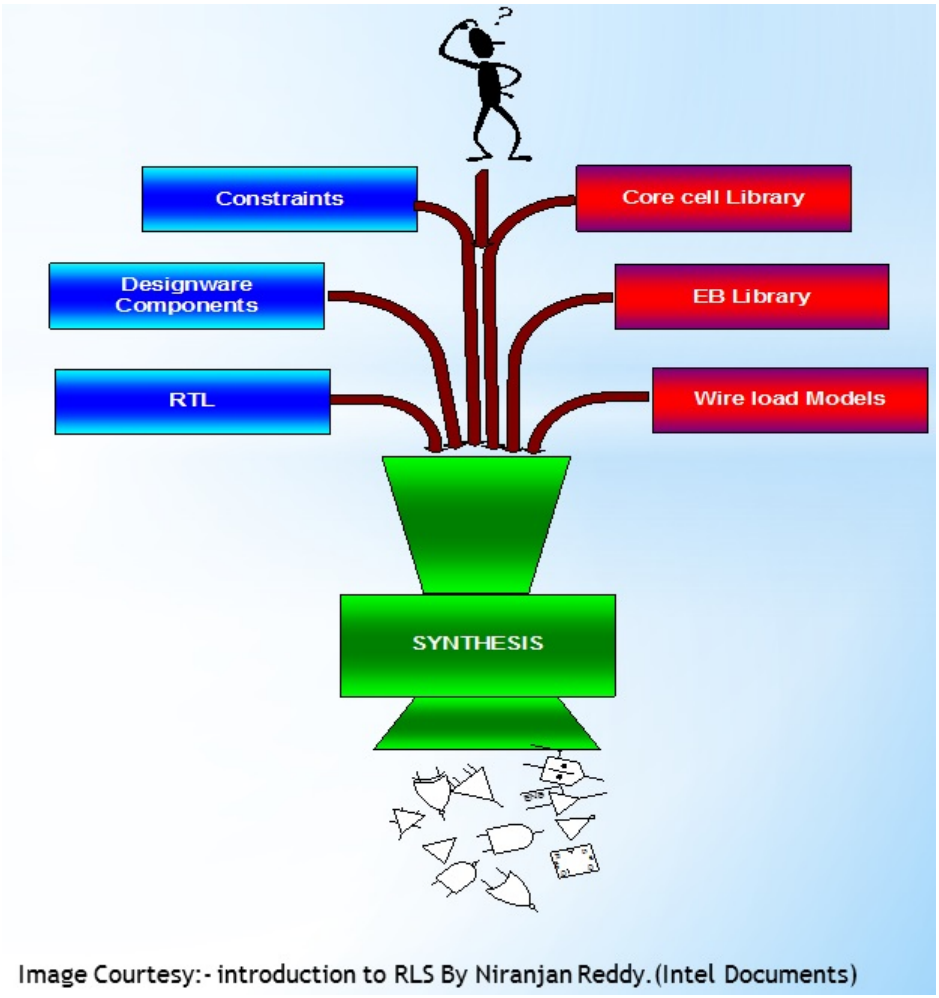

Figure 3.4: Synthesis

For example: Designer wants to implement adder like out <= A+B;

DC may use a Ripple Carry or Carry Look Ahead adder (or any other) depending on speed/area constraints. These components are part of Designware. Design constraints are those which must meet in order to finish the design. These include clock information, operating conditions, input/output delays, timing exceptions, power requirements.

So in this synthesis step the whole optimization is done in a manner that the synthesized netlist is best of needed. It uses some architectural optimization, resource sharing, arithmetic optimization, logic level optimization, Gate level optimization.

3.3 Floorplanning

This step of the design will defines the overall topology of the design. However do not confuse with the floorplan of the full section or full chip as it is taken care by the section layout owner, who gives us the floorplan collaterals and from that collaterals we do floorplanning for our design. This floorplan will give the relative placement of the modules, global wiring like clock network, power supply network, input/output port locations etc.Floorplanning is an iterative process which fits in between synthesis and physical synthesis; and transforms a logical netlist into a semi-physical form. Main objective of floorplanning is to establish locations for I/Os, power, hard-macros, pre-routes, and partitions to achieve design objectives of performance, area, and power.

The inputs required to floorplanner are netlist (gate level netlist), timing constraints (clock definition etc), area constraints, physical definitions of standard cells, EBBs etc. The IO location file if your IOs are determined earlier itself.

The figure 3.5 is only gives the idea of the floorplan. It is not related to my design at Intel Corporation.

3.4 Placement

Floorplanner provides all the information needed to this placement tool. The information contains all sort of physical info, Die size estimation, I/O placement, and power structure creation, block size estimation and creation, and pin placement. The main objectives of the placement is to place the cells within the area floorplanner provided, timing should be matched after considering actual net delay, to address congestion issues, and make the database ready for the router.

For easy placement of standard cells, every floorplan is divided into rows. Standard

Figure 3.5: Floorplan

cells are placed on 1 or more rows, such that their edges align with the row edges. This is the reason why the height of a standard cell is maintained as a quantum of row height. Rows typically contain the power straps, so that when standard cells are placed on a row, they are automatically connected to the power supply, VCC and VSS.

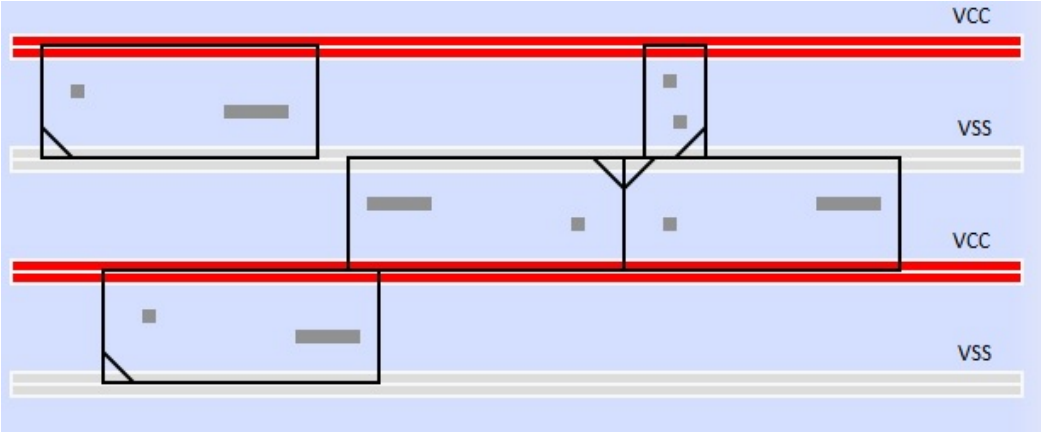

Figure 3.6: Placement

In this step the tool place the standard cells based on the following two manners depending upon the type of optimization needed.

- Timing based

All standard cells that lie on critical timing paths are kept as close as possible. This will minimize the net lengths, and hence net delays and transition degradation. This can often lead to too many wires in one area and hence can cause congestion in that area and affects final timing.

- Congestion based
  In this type of placement the standard cells are placed such that the density of cells remains uniform across the floorplan, and net lengths are kept minimum, while minimizing the congestion. This can cause longer net lengths for timing critical paths and adversely affects timing.

3.5 Clock Tree Synthesis

This step in the design style gives a good improvement in terms of meeting the timing requirements of the design. Now a day the complexity is much higher so we have to do this clock distribution in a manner that will help us to improve the performance also. CTS is the process of distributing clock signals to clock pins based on physical/layout information provided through the previous steps. After placement of cells the tree of synchronization is synthesized.

Before the CTS all the clock pins are driven by a single clock source. All clock pins are from a source of clock pulses in various geometric distances. This situation is shown in the below figure 3.7

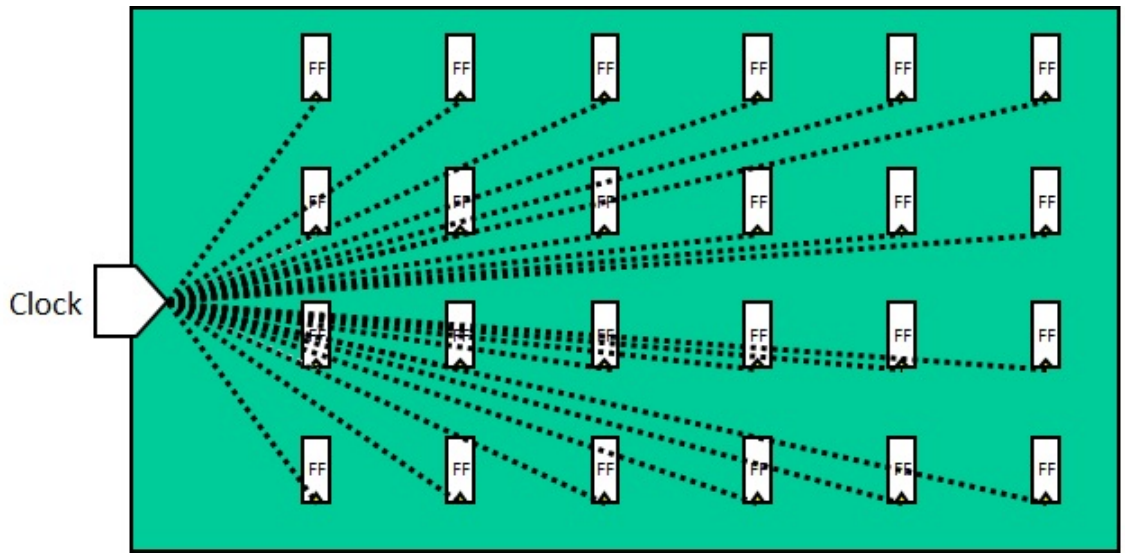

Figure 3.7: Before clock tree synthesis

After the CTS step this clock tree is synthesized in the well-organized manner which

gives the improved result in terms of skew issue. In CTS buffer tree is built to balance the loads and minimize the skew. The figure 3.8 shows the after CTS result of the above design.

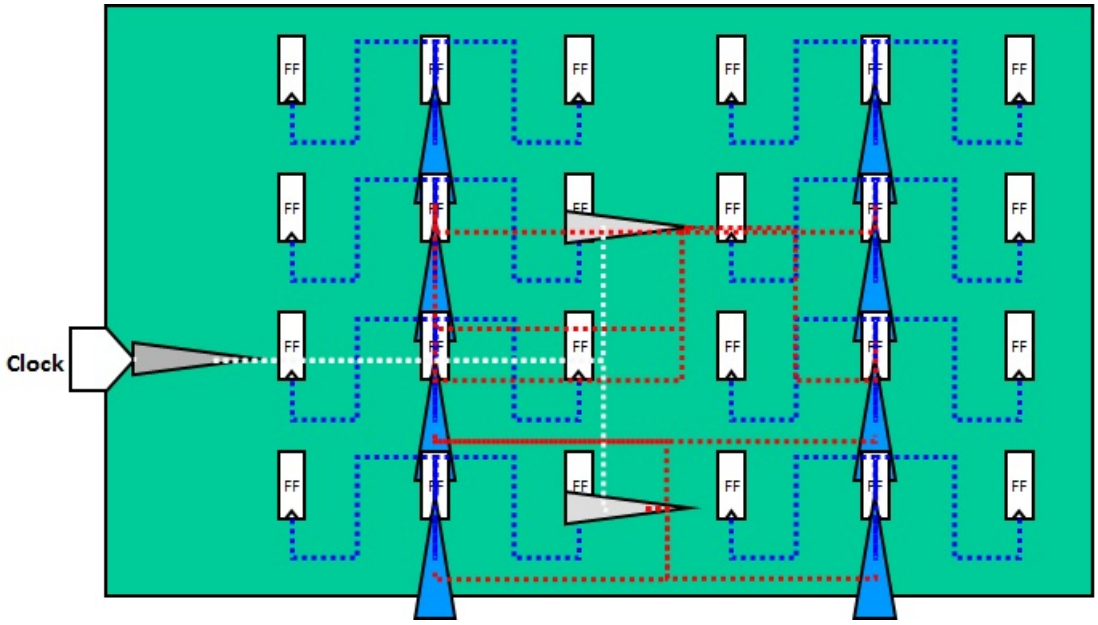

Figure 3.8: After clock tree synthesis

3.6 Routing

After Placement, now its time to connect all blocks and all the modules in order to make them work. Routing creates physical connections to all clock and signal pins through metal interconnects. Routed paths must meet setup and hold timing, max cap/trans, and clock skew requirements as predicted and provided in the earlier steps and it satisfied in those steps. Routed Metals must meet physical DRC requirements in order to make design manufacture able. Routed design should be LVS (short/opens) clean to make it functionally true.

The below fig 3.9 give the idea of intermediate steps in this routing step.

- Global routing
  It assigns nets to specific metal layers and global routing cells. Global route does not lay down any metals, it creates only topology. Global Route takes into consideration Terminals, Routing Blockages, Power.

- Track assignment
  Assigns each net to a specific track and lays down the actual metal traces. It also

Figure 3.9: Steps of routing

attempts to make long, straight traces and reduces the no of vias as possible. This track assignment does not check or follow the physical DRC rules.

- Detail route
  Detail route attempts to clear DRC violations using a fixed size box. Due to this fixed box size, detail route may not be able to clear all the DRC violations.

- Search and Repair
  Search and Repair fixes remaining DRC violations through multiple loops using progressively larger box sizes. Even if the design is DRC clean after search and repair, we must still run a sign-off DRC checker to ensure that all DRCs are cleaned or not. Routing DRC rules are a subset of the complete technology DRC rules.

3.7 Chip Finishing flow

After the complete flow up-to this point we have a ready data base which is satisfied our speed, power, and performance constraints. But at the lase we have do some checks which proves that the converged design can be manufactured without any errors ignoring the errors occurs during the manufacturing, and then it will works fine. But in reality it not the case we have to do several checks on this database in order to make it reliable. These checks include Layout vs schematic verification (LVS), Design Rule Check (DRC), Reliability Verification (RV), and Design for manufacturability (DFM).

- DRC
  Design Rule Checks (DRCs) are a set of speculative design rules for a new layout, consistent with the initial expectations of the red book for the process that is being used. DRCs look for violations in spacing, width, and enclosure of polygons according to a specific set of process rules. Refer to the DRC redbook for detailed information.

- LVS

  This flow is used to check whether the layout drawn by Mask Design is according to the schematic description which is designed by the DEs (Design Engineers). LVS is the most required check in the chip finishing flow. It must come out clean then and then the database will go further for design. IT will checks for the number of devises on both the side of each type, no of nets and their connections, property of devises must match and also the ports should match. Most of the common errors in this check are shorts/opens, missing devices, property mismatches.

- DFM

  This is the advance step now a days used in the industry which has the advantage of good productivity of the chip when it is manufactured in the fabrication lab. In this step some of the DFM guide lines are followed which makes the design more easy to manufacture and will reduce the chance of not manufacturable design. One of the most use full technique is metal dummyfication, In this the empty space is filled up by some dummy metal, which makes the density of metal across whole die uniform and will reduce the defects through manufacturing process.

Chapter 4

Timing Analysis and optimization

Timing analysis is carried out in order to ensure that design meets the target frequency. And if not then debug the issue and do the optimization accordingly. Timing analysis can be done using simulation method or some static timing method. The analysis carried out by simulation is called dynamic timing analysis but the analysis carried out by static methods is called static timing analysis (STA). Which method to choose is depends upon how big the circuit is. The below is the description of both the methods and for particular this project I have used static timing analysis and the techniques mentioned letter in this chapter is based on the results obtained from the STA and used in the project.

- Dynamic timing analysis
  Dynamic timing analysis uses circuit simulation for obtaining very accurate timing result of a path, using input waveforms and generating output wave forms.
  Example Tools: Spice
  It has the advantages like highest degree of accuracy possible and gives precise delay information. However it has some disadvantages like it requires a debug process to find the origin of the timing problem, requires a lot of CPU time for good coverage, inherent problem is the test vector generation.

- Static timing analysis
  Static timing analysis is a non-simulation approach, which break the circuit into sets of timing paths. Calculate the delay of each element, Propagate the signals with the below worst case assumptions: (1) Signal becomes stable at latest possible time, (2) Signal becomes unstable at the earliest possible time, and then it will check timing Constraints. If the design works at these extremes, we can guarantee that it will always work at these constraints.
  Example Tools: Primetime
  It has the advantages of Independent of Input Vectors, Points to exact origin of the timing problem, Ensures full coverage, Faster than Simulation, Relies on Algebraic modeling for delays of circuit. And also some of the disadvantages like Less Accurate than Simulation, Cannot Identify False Paths, Gives Worst Case Delays.

4.1 Timing

Timing for a design is broadly divided into two parts:
- Internal timing
- External timing

4.1.1 Internal timing

Internal timing is the timing of paths from register to register within the block. Internal timing is dependent on the transport delays of logic elements on register-to-register paths and the overall effects of parasitic capacitance, parasitic resistance, and crosstalk on routing connections between those logic elements.

To ensure that the timing of the device meets performance goals, the tool runs static timing analysis on the design database. For this timing analysis to be meaningful, all timing constraints and timing exceptions that you applied to the design must be constrained. If you did not use timing constraints or you used only partial timing constraints for the design, you must add constraints manually until the real constraints gets applied.

The following constraints must be included:
- Clock definitions
- Primary input port timing
- Primary output port timing
- Combinational timing
- Timing exceptions

However flow tries to do the best job to converge internal timing paths, but for larger designs sometimes it might not, if it sees high congestion, the interconnect delays will increase to route the adjacent cells as well which were sitting nearby .

4.1.2 External Timing

We must specify the interface port timing constraints for every primary input and output ports in the design. The following two subsections describe how to constrain input and output port timing.

-External Input Delay Specification (figure 4.1): To constrain the input port timing, describe the external timing environment in terms of the maximum and minimum arrival times of the external signals that drive the primary input ports of the design. Figure shows the external timing constraint that drives the primary input port. The static tim-

ing analysis tool can use this external input delay time to check if there is enough time for the data to propagate to the internal nodes of the device. If there is not enough time, a timing violation occurs.

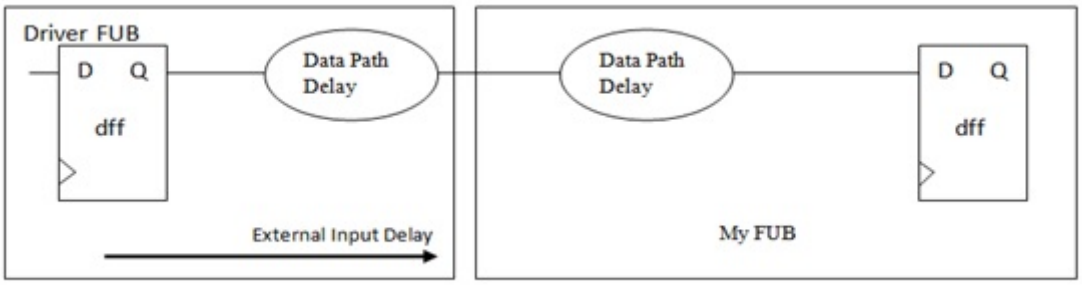

Figure 4.1: External input delay

-External Output Delay Specification (figure 4.2): One way to capture output port timing is to describe the external timing environment, which is the maximum and minimum delay times of external signals that are driven by the primary output ports of the design. Figure shows the external timing constraint driven by the primary output port. The static timing analysis tool uses this information to check that the on-chip timing of the output signals is within the desired specification.

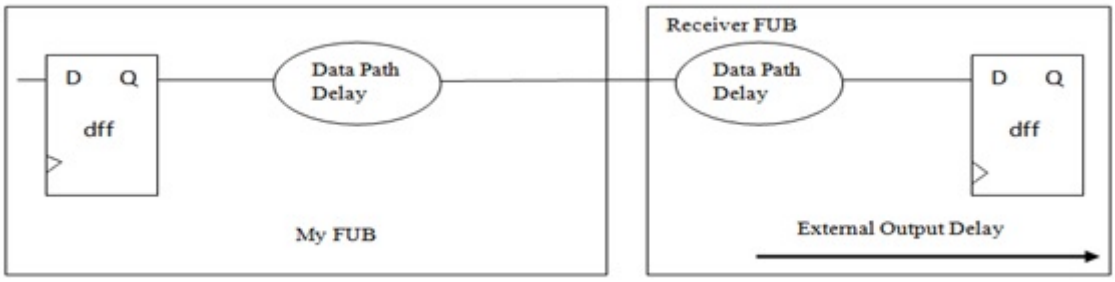

Figure 4.2: External output delay

4.2 Violations

4.2.1 Setup or max violation

Setup time is the minimum amount of time the data signal should be held steady before the clock event so that the data are reliably sampled by the clock. The figure 4.3 show paths between two flip-flops where many combinational paths might exist.

Key Considerations for setup checks are:
- Maximum Delay for the Data Path (Worst Path)

- Minimum Delay for the Clock Path

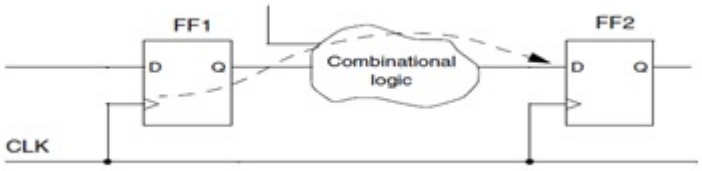

Figure 4.3: Paths between two flip-flops

Figure 4.4 timing diagram for setup check. The arrival of a clock edge at FF1 latches the data at the input FF1.D into the flip-flop. It also places that data on the flip-flop output, FF1.Q, after the clock-to-Q delay of the flip-flop. This is called the launch event for the timing path. This signal goes through the combinational logic with some delay. The output of the combinational logic is at the input of the second flip-flop, FF2.D. The time at which the signal value changes here is called the arrival time for the path.

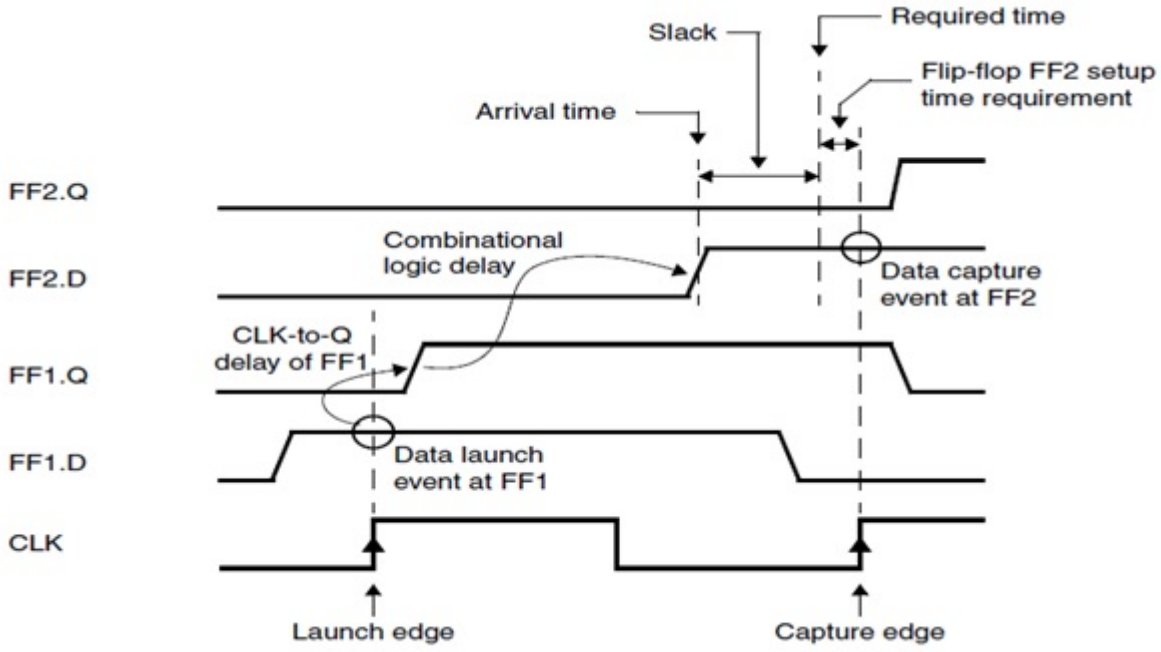

Figure 4.4: Timing Diagram for setup check

4.2.2 Hold of min violation

Hold time is the minimum amount of time the data signal should be held steady after the clock event so that the data are reliably sampled. The figure 4.5 shows typical case

of hold violation.

Key Considerations for hold checks are:
- Minimum Delay for the Data Path (Shortest Path)
- Maximum Delay for the Clock Path
-The optimization technique for hold violations is to add high delay buffers to the data path.
Rare cases we can increase the rc delay by routing in lower metal layers. You can also downsize the cell. Point being : we need to increase the delay.

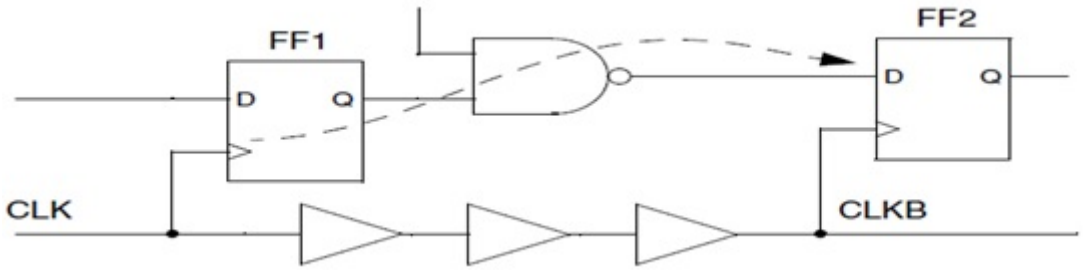

Figure 4.5: Typical case of hold violation

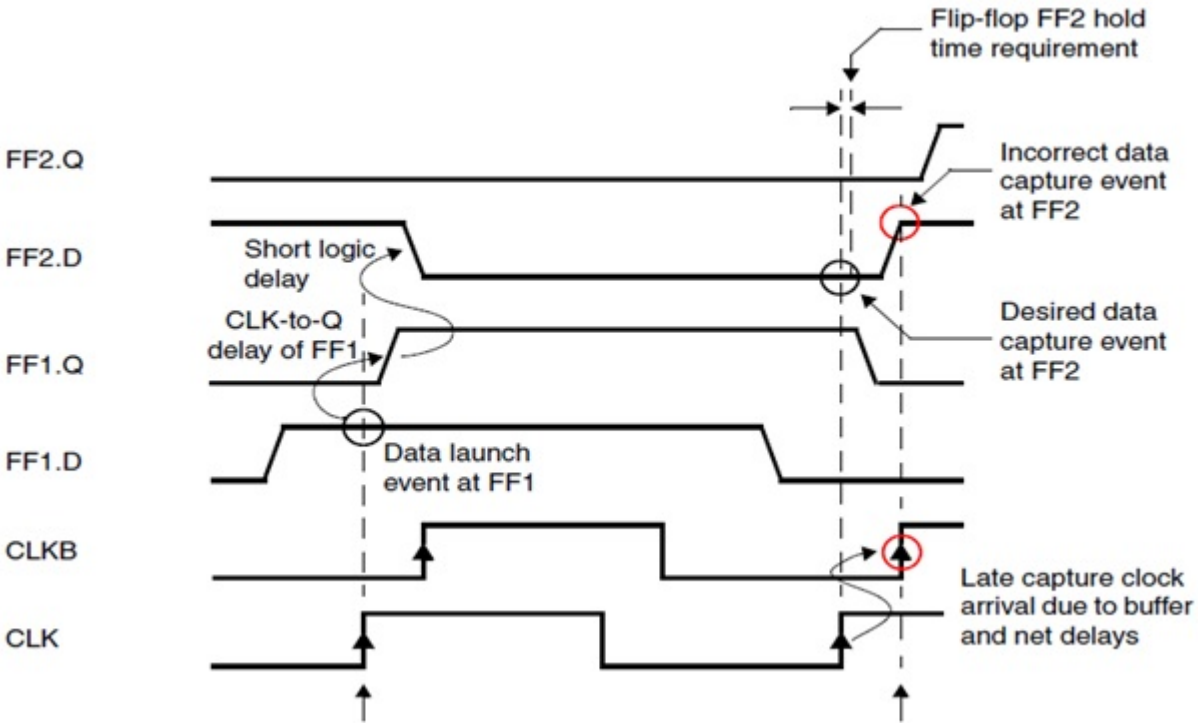

Figure 4.6: Timing Diagram for a hold check

4.3 Timing Optimization methodology

4.3.1 Timing Exceptions

In most designs there may be paths that exhibit timing exceptions. For instance, some parts of the logic may have been designed to function as multicycle paths, while others may simply be false paths. Therefore, before analyzing the design, tool must be made aware of the special behavior exhibited by these paths. Tool may report timing violation for multicycle paths if they are not specified as such. Also, path segments in the design that is not factual, must be identified and specified as false paths, in order to prevent tool from producing the timing reports for these paths.

- Multicycle Paths
  Timing tool by default, treats all paths in the design as single-cycle and performs the STA accordingly, i.e., data is launched from the driving flop using the first edge of the clock, and is captured by the receiving flop using the second edge of the clock. This means that the data must be received by the receiving flop within one clock cycle (single clock period). In the multicycle mode, the data may take more than one clock cycle to reach its destination. The amount of time taken by the data to reach its destination is governed by the multiplier value used in the command "set multicycle path".

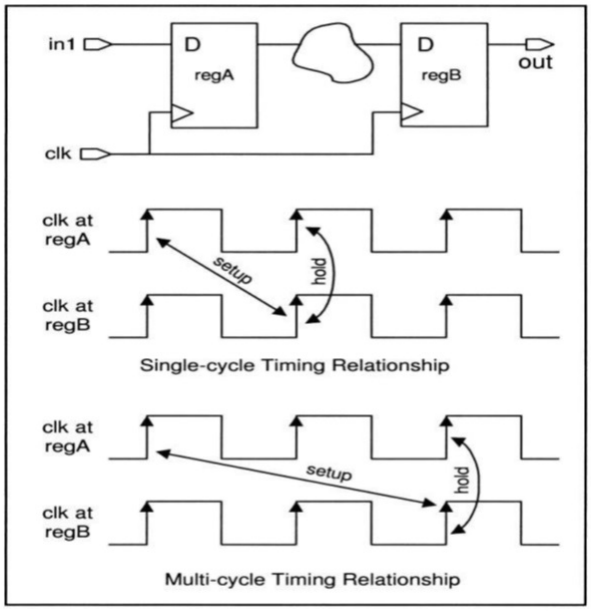

Figure 4.7: Multicycle timing relationship

Above figure 4.7 illustrates the comparison between the single-cycle setup/hold time relationship and the multicycle setup/hold-time relationship. In the multicycle def-

inition, a multiplier value of 2 is used to inform timing tool that the data latching occurs at regB after an additional clock pulse.
In case of generated clocks, Timing tool does not automatically determine the relationship between the primary clock and the derived clock, even if the "create generated clock" command is used. The single cycle determination is independent of whether one clock is generated or not. It is based on the smallest interval between the open edge of the first clock to the closing edge of the second clock (in this case generated clock).
In case of MCP it should not be coded by backend design engineers, as wrong MCP coding might lead to a scenario where the paths will always be in positive margin as there will be an extra clock cycle for the path to converge which is wrongly calculated positive path and hence it is always desired that MCP be coded at the RTL level by the RTL designers.

4.3.2 Path Grouping

Path Grouping is a technique of grouping critical paths giving more weightage where tool will put more efforts to optimize the design as shown in figure 4.8.

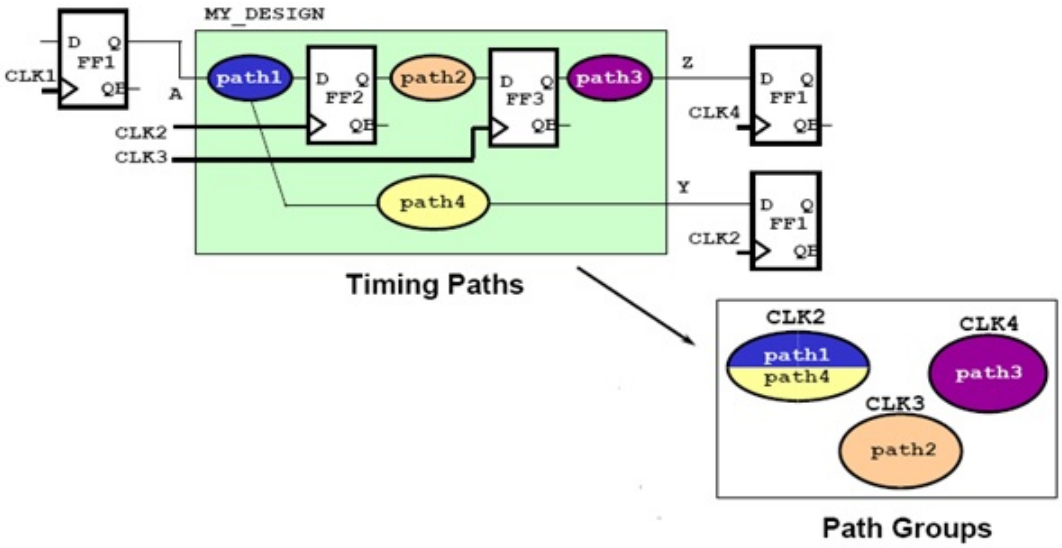

Figure 4.8: An example of Path Grouping

Example: grouppath -to ff[*]reg/d -weight 3 -name gp1

Here weight gives the amount of effort the tool should spend on a given path; by default the weight is 1, if the weight is more the tool will try to optimize that path even though the other path is violating by a higher WNS. Here the end point to ff[*]reg flops is given a weight of 3, so all the paths to this flop, the tool will spend extra effort in optimizing, this can be made limited by including from which starting path to ending

path.

Group path works good only when the path in optimizing tool has a negative margin in the path, if the path is in positive margin it is not advisable to use group path.

4.3.3 Optimizing paths with positive margin

If the path is in positive margin in the optimizing tool and its negative in the timing signoff tool then we need to make the path violate in the optimizing tool so that the tool can spend effort in optimizing the negative path, by default if the path is in positive margin then the tool doesnt spend more effort in optimizing the path.

Example: setmaxdelay -from flop[*] -to flop5/d 0.140

In the above scenario if the path is taking 200ps and having positive margin of 20ps in icc and its violating in signoff tool by 20ps, then with the above command the path will violate by 40ps. Then ICC tool will try to optimize the path and meet it in the signoff tool as well.

During optimization using the group path or setmaxdelay the tool will try to use higher width gates which have higher drive strength and use higher routing layers so that the delay is minimized as given by equations 1 and 2:

$$I_D = \frac{\mu_n \varepsilon_{ox}}{2T_{ox}} \frac{W}{L} \left(V_{GS} - V_{tn} \right)^2$$

Figure 4.9: Equation 1

n is the electron mobility, ox is the thin oxide (SiO2) dielectric constant, Tox is transistor thin oxide thickness, and W and L are transistor effective gate width and length.

$$R = \rho \frac{L}{A} = \rho \frac{L}{Wt}$$

Figure 4.10: Equation 2

Where A is the cross-sectional area and L is the length. The cross-sectional area can be split into the width W and the sheet thickness t .

Routing will also take straight jogs if it was taking many zigzags routing previously.

4.3.4 Timing Optimization for external Paths

Giving good timing constraints for the external path is very important for the FUB to converge section timing as well as internal timing. If all the external paths are over constrained too much then the tool might put lot of effort in the external paths to meet the positive margin on them and the internal timing might go bad on other hand if the external paths are very relaxed then the tool will not put more effort on those external paths and the external paths might not meet the timing specs at the section level. An example of giving good constraints is shown in figure: 4.11

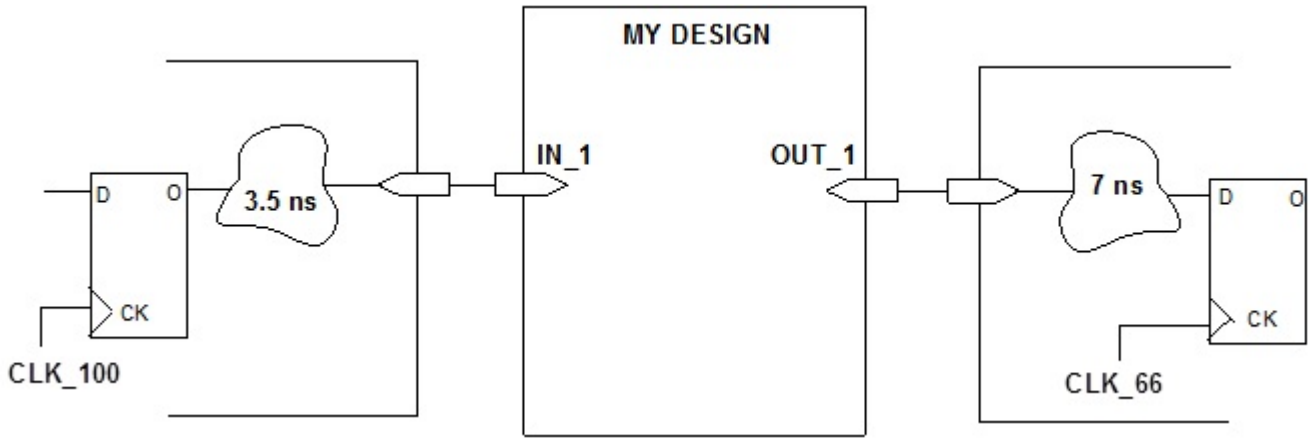

Figure 4.11: Timing constraints for external paths

Assuming clk100 time period is 5ns and clk66 time period is 10ns, then the following are type of constraints:

- Good constraints:
  setinputdelay 3.5 -rise -clock clk100 [getport IN1]
  setoutputdelay 7.0 -clock clk66 [getport OUT1]
  Here the input logic to the fub at the input side is given 1.5ns and at the output side is given 3ns, the inside logic present inside the fub will be optimized without have much impact on the internal timing.

- Relaxed constraints:
  setinputdelay 1.2 -rise -clock clk100 [getport IN1]
  setoutputdelay 2.5 -clock clk66 [getport OUT1]
  Here the input logic to the FUB at the input side is given 3.8ns and at the output side is given 7.5ns, the interface logic inside the FUB will not be optimized as the constraints are very relaxed these paths will not be optimized, but this will not have an impact on the internal timing in the FUB.

- Over constrained Paths:
  setinputdelay 4.9 -rise -clock clk100 [getport IN1]
  setoutputdelay 9.8 -clock clk66 [getport OUT1]
  Here the input logic to the FUB at the input side is given only 0.1ns and at the output side is given only 0.2ns, the interface logic inside the FUB will try to optimize these paths putting too much effort on these paths it might lead to degradation in internal timing as the tool will try focus more on external paths.

4.3.5 Bind2ports

If the input port and the output ports are sitting very far then the tool inserts lot of buffers in between this might lead to increase in the delay of the path. Bind2ports is a technique in which the register driving the output port and the register form the input port are made to place very close to the ports so that extra buffers are avoided, and the registers should be selected with maximum drive strength so that it is able to drive the ports as the capacitance of the I/O ports will be very high compared to that of the gates.

```
bind2ports port1[*] 60
setidealnet [ getnets  port1[*]  ]
setdonttouch [ getnets port1[*]  ]
changelink reg1[*]reg cc0lax00nn0i0
```

- Here bind2ports command places all the registers connected to the corresponding port1[*] within 60 microns.
- The setidealnet and setdonttouch commands avoid adding buffers in the path and hence minimize the delay.
- Changelink changes the width of the latch to i0, which will be able to drive the output ports if the width of the device is c0 or e0, it might not be able to drive the output port.
- Using these bind2ports technique two to three buffers were avoided and the external timing was improved by 80 to 90 ps.

So till now I have explained in detail each of the timing concern. Now I am summarizing this chapter by mentioning the techniques apart from the above basic things in order to meet the timing.

- Altered the timing constraints on failing path ports

- Over constraints the paths which have setup violations
- Relaxed the constraints for the paths having hold violations
- Sizing the standard cells
- Changing port locations
- Higher metal layer routing or priority routing
- By passing Buffers or Buffer Minimization/Addition

As this RLS methodology is tool based methodology so the tool is trying to optimize those paths which have higher negative slack and these paths are called critical paths. In the above first 3 techniques are used in order to get the best output from tool means that we can change the constraints of the critical paths and give it to the tool and tool will start working on those paths and it will try to meet the timing for those paths. So after this there are some handful numbers of paths which are failing in timing and then I can analyze those each individual and try the remaining appropriate technique for them.

4.3.6 Timing Checks

Apart from these setup and hold violations there are lots of other project specified checks which needs to be satisfied. These checks are needed because the limits set by project is based on the past silicon experience and if the values are above the limit then the behavior of the silicon is unknown and it may be affect the functionality. These checks include slope limit on output pin of standard cell, maximum delay limit on standard cell, maximum effective capacitance of standard cell, maximum fan-out limit. Below I have explained some of the checks and related fixes for those.

- **Slowslope :-**
  This rule is checks for receivers (at input pins) where input slope is greater than project specified slope limit. The rule reports both data and clock pins.
  Possible impact:
  Timing results may be inaccurate
  Increased Noise Sensitivity
  Increased VT Sensitivity (results in extreme variations between PV models and real silicon)
  Fix:
  Increase slopes through increasing drive strength of driver and/or wire topology changes.

- **Cmax:-**
  This rule is checks for drivers where Ceff (Effective capacitance) is greater than

CMAX of characterized standard cell.
Possible impact:
Timing results may be inaccurate.
Fix:
Upsize the driver
Reduce load seen by the driver.

- **SlowCellDelay:-**
  This rule reports cells with delay greater than project default.
  The cause for this violation is the cell has higher fanout or due to capacitive overloading at output it has too slower slope at output pin.
  Fix:
  Reduce the cell fanout or upsize the cell.

4.4 Results of timing optimization

Till now I have discussed everything about optimization of the timing and how we can improve the timing using these techniques. But for the reader to believe it is needed to give statistics. Below I have included three sets of results which are real numbers which I have taken for reference while I am designing the blocks. I have not considered individual technique impact on results but I have used all techniques for the improvement in timing.

Initial Stage:–

Table 4.1: Timing statistics at Initial Stage

Parameter	Value
Total Cell count (sequential, Inverters, buffers)	183685
Utilization and Shorts	42.76% with 27 shorts
Hold Violations(Min internal Paths)	300 paths violating with Worst Negative Slack of -27ps
Setup Violations(Max internal Paths)	100 paths violating with Worst Negative Slack of -18 ps
Timing Checks (Cmax, Slowcelldelay, Slowslope)	745 (395, 218, 132)

Intermediate stage:–

Table 4.2: Timing statistics at Intermediate Stage

Parameter	Value
Total Cell count (sequential, Inverters, buffers)	185274
Utilization and Shorts	43.2% with 3 shorts
Hold Violations(Min internal Paths)	100 paths violating with Worst Negative Slack of -4ps
Setup Violations(Max internal Paths)	10 paths violating with Worst Negative Slack of -9 ps
Timing Checks (Cmax, Slowcelldelay, Slowslope)	156(32,110,14)

Final Stage:–

Table 4.3: Timing statistics at Final Stage

Parameter	Value
Total Cell count (sequential, Inverters, buffers)	185825
Utilization and Shorts	43.31% with 0 shorts
Hold Violations(Min internal Paths)	No paths violating. Worst Negative Slack is 0 ps
Setup Violations(Max internal Paths)	No paths violating. Worst Negative Slack is +42 ps
Timing Checks (Cmax, Slowcelldelay, Slowslope)	8 (8,0,0)

Chapter 5

Miscellaneous Optimizations

In the last chapter I have mentioned most of the analysis which are used by each designer in order to meet timing requirement. Now in this chapter I am going to discuss some other optimizations related to noise, power and area. And at the last I have also included some checks of layout and reliability.

This project report is more focused on the particular section of the chip in which I have worked. In the server chip this section is designed differently as it is a big section and the size of the section is big concern. This section is called IIO (Interface Inputs Outputs) and it has the area of almost 30 % of the total chip area and this section is more critical as the TTM is more depended on this section. The section is divided in to 26 blocks and all are of different sizes and I have worked on two medium size moderate complex fubs. There are lots of inter connects between all the fubs in the section so we are using a different methodology called ICCDP. In this the section is implemented with abutted blocks that means there are no channels in between the two fubs so that the area can be reduced. But the problem comes when the collaterals for the individuals needs to be generated. There are lots of interconnect nets between fubs which is in normal flow will be routed in the section routes (in channels between the FUBs), but in this ICCDP flow these signals are routed over the FUBs with the help of repeaters if the signal is too long. The main aspect is generating the collaterals due to two reasons. First is section routes are pushed down to FUB. And second is generating the port location for FUB as the location of the port is more critical in order to meet the timing at FUB level. So in ICCDP flow along with normal flow collaterals like block dimension, power grid tracks, port locations, and macro dimensions, We also have section routes as pre-route and repeaters placement.

Now I am going to discuss the optimization techniques in the different area which I have used while designing my Functional Unit Blocks.

5.1 Placement Optimization

The total cell count is about 200k (which includes sequential cells, buffers, and inverters) for my design and I have many register files in my design so the placement of those are very important in order to minimize the routing congestion and number of shorts to be less. So I have used below mentioned techniques.

- Use of placement blockages
- Use of routing blockages
- Proper placement of Embedded Black Boxes (EBB)
- Controlling pin density by proper partial placement blockages.

As I have mentioned earlier that we have used the ICCDP methodology for this section, to give an idea of the advantage of this method I have given the results of DRC violations below (Table 5.1) without use of ICCDP flow. You can see that there are lots of shorts and spacing violations in this.

Table 5.1: List of DRC's without ICCDP flow

Count	Violation
28422	Diff net spacing
74	Same net spacing
28800	Diff net via-cut spacing
439	Same net via-cut spacing
22	Less than minimum width
1213	Less than minimum length
4951	Less than minimum edge length
344	Needs fat contact
731	Enclosed via spacing
5	Connection not within pin
1086	Off-grid
42	Non-preferred direction route
976	Illegal width route
343	Multiple pin connections

Total DRC violation = 106794

Number of Shorts = more than 9998 (flow doesnt report more than 9998 shorts)

Now we used the ICCDP flow in which we have routed the interconnection nets between the fubs in the section floor plan stage itself. And then I have been provide the collaterals with the pre-routes from section and then I started my FUB design and I have seen the large decrease in the short count it came down to around 300. The below is the figure 5.1 which gives the idea of complete placement and short locations at this stage.

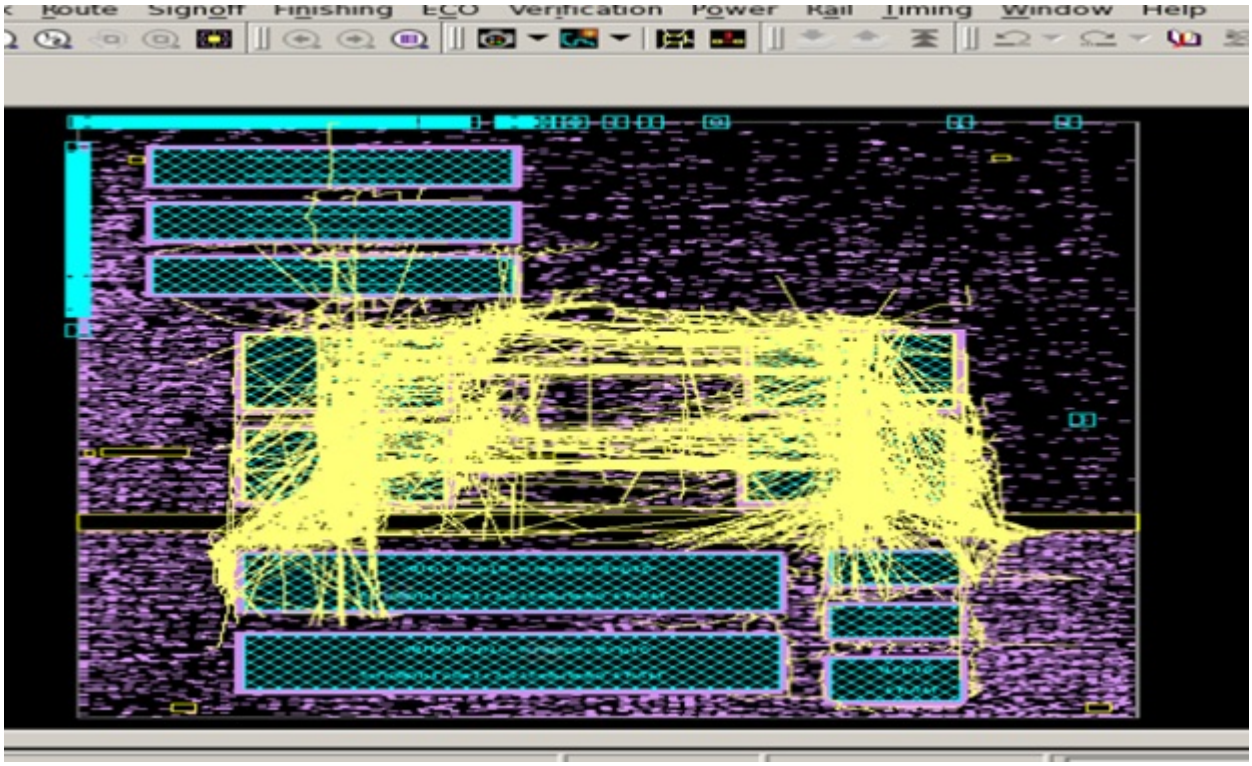

Figure 5.1: Congetion and short snapshot

We can see from the figure 5.1 that these shorts are more around center macros (EBBs). And so If we can move that macro by analyzing the connections between them in order not to create any timing violations. And after trying some few experiments each with the different placement of the center 4 macros I found the most suitable placement for these macros and It is shown in the below figure 5.2.

And after the this location I am able to reduce the short count from 300 to around 60 for such a 200k large block which is very handful and can be cleaned at the time of layout clean up. The yellow outlined parts in the above figure are placement blockages and those are called PKORs (Placement Keep Out Regions). These PKORs are due to some repeaters and level shifter, which will be placed when the FUB is integrated with the section.

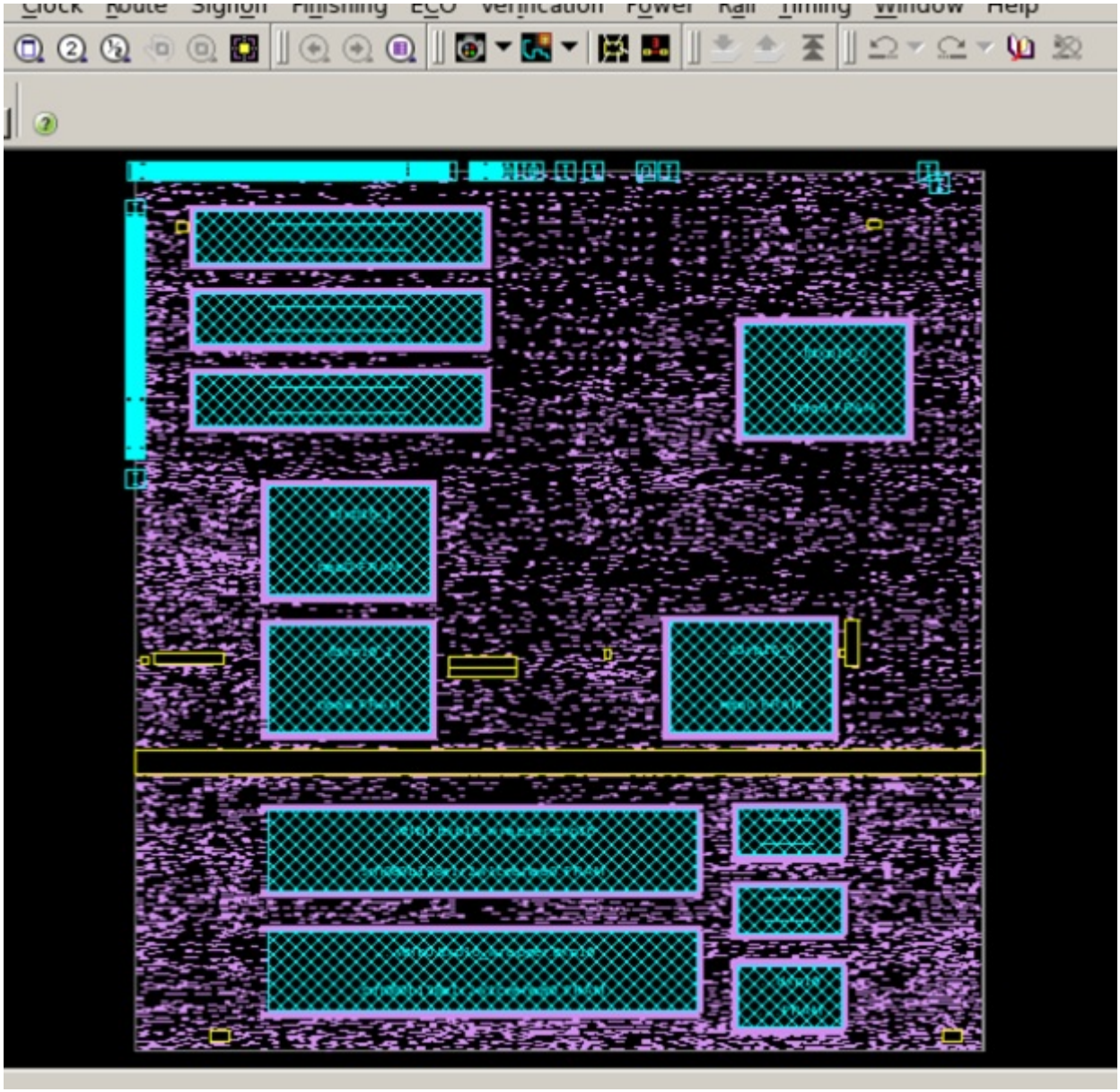

Figure 5.2: Final Placement snapshot

5.2 Power Optimization

One of the most important constraints in the current shrink technology is POWER. As most if the devices are battery operated the amount of power need by the chip must be minimized in order to extend the battery life. However the requirement in the server domain is slightly different, here the power must be lowered to minimize the use of cooling arrangements. The cost of cooling is more for server chip so the power minimization is second critical aspect for sever chip design. In the current era power is try to reduce mainly at the architectural level, device level, RTL level and design level. There are two power dissipations which are focused and those are leakage power and active power (dynamic power). As the technology is shrinked the leakage power must be controlled in order to minimize the damage caused by hot electron or to protect the oxide under the gate area. The below are techniques which I used in my design to minimize the power.

- **Clock gating:**
  In a SoC, most of the dynamic power is dissipated in clock path because it has high switching activity. Clock gating is the technique used to reduce switching activity in clock path registers when register input data is not changing. To implement clock gating we need to put special cells in clock path, these cells functionality is similar to and gate. Placing one clock gate cell for every register which has low activity on input data is not efficiently because it will increase clock gating cells which intern increase area of implementation and leakage power, so in this clock gating was implemented by identifying group registers which has similar data activity, common enable signal and common source of clock.

  The most common circuit used for this technique is latch based clock gate shown in the figure 5.3. Clock gate has been widely used to reduce dynamic power of clock tree network. The clock gate is constructed by using a AND gate and a latch. This structure will prevent the glitch from being propagated from input of the clock gate to the output of the clock gate during shut down stage and turn on stage.

  After implementing clock gating, formal verification will fail between implemented netlist and corresponding RTL because for some clock gating cells enable signal is coming from asynchronous reset, so we excluded those registers from clock gating to pass formal verification. As this clock getting will not affect the functionality in rare case It does then It will catch by the section FEV run.

- **Vectoring:**
  Vectoring is used to reduce leakage power in clock path. Vectoring is the technique to combine multiple register which are placed closed and has common clock source

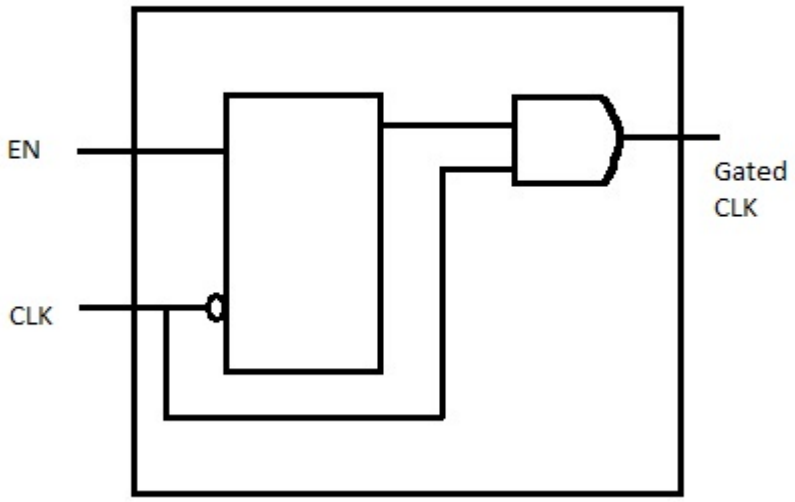

Figure 5.3: Latch based Gated Clock

and common enable signals into single register having multiple inputs and single clock input so it will reduce number of cells in clock path. But disadvantage of this technique is sometimes it will affect the timing of design. However there is an advantage of this technique is it will reduce the area at some extent.

- **Using high Vt cells using TANGO-LR:**
  The cell with lower threshold voltage (Vt) has high leakage power compared to high threshold cell. As the threshold voltage is lower the device can be turned on with low voltage and so it is more sensitive to noise and any high spike of noise can turn on the device, However the leakage current of low Vt cell is more and hence it has high leakage power.

  TANGO-LR is an Intel specific tool which will work on those paths which have high max timing margin (meeting setup time with more positive margin) and then it will swap high leakage cells with low leakage cells in those paths.

5.2.1 Results of Power Optimization

The below Table 5.2 is the results for one of my block where I have used the above techniques and I got better improvement from that. I am not allowed to present this numbers as it is Intel confidential so I am giving numbers for reference purpose only. From the below (Table 5.2) statistics I can say that there is a significant improvement of around 37% in power consumption due to these optimization techniques. And this improvement means a lot if we think the overall improvement with respect to chip level which has many such blocks.

Without Clock gating, vectoring and Tango-LR:–

Table 5.2: Without Power optimization techniques

Parameter	Value
Active Power	19.3308 mW
Leakage Power	5.1309 mW
Total Power	24.4617 mW

With Clock gating, vectoring and Tango-LR:–

Table 5.3: With Power optimization techniques

Parameter	Value
Active Power	12.5368 mW
Leakage Power	3.0422 mW
Total Power	15.579 mW

5.3 Noise Convergence

Noise is another unwanted aspect for deep sub-micron technology. As the deep submicron technology has operating voltage around 1 volt, the noise convergence is dominant. The functional failures due to some noise spikes are highly unwanted. These are the main sources of noise which includes interconnect noise, propagated noise, dc droop noise, and charge sharing noise. I have used noisesim tool to analyze my designs and I have found many noise violations due to the bus structure implementation of section routes. I have root caused the violations and come to the conclusion that the below are the main cause of the noise in my design.

- Long parallel routes
- Bad tail routing at sectional repeater output pins (Many jogs and zigzag routes)
- Missing of via ladders at high drive strength output pins.

After seeing these violations I have fixed those depending on the place of origination. For finding the place to fix I have to back trace the net if it is propagated noise and then find the net which has high interconnect noise. Then there are two fixes for this if the noise is low then I can route the net in higher layers or upsize the drive strength of driver

of that net so it will reduce the noise. If the noise is very high then I have to break that net and insert one or two buffer depending upon the route length. One more technique is to shield the victim net with the help of power tracks. Half shielding is to route that net with one side power tract and if power tracks are at two sided then it is full shielding. But the more critical task here is identifying the failing net. If we insert the buffer in the wrong net then it might amplify the noise instead suppression.

5.3.1 Noise Violations Summary

The below table 5.4 contains the noise results for my FUB. All the violations are must fix in order to make sure the real silicon will not affect by any noise. I have used the buffer insertion, and routing the net in higher metal layer and shielding the net with power tracks technique in order to overcome the noise violations shown in the below table 5.4.

Table 5.4: Noise Violations

Type of Violation	**No of Violation**
Clock Node	1
Storage Node	9
Cmos	14

Chapter 6

Tools and technology used

Up to this point I have mentioned about the concepts and methodology I followed during my project. And I also give some idea about the timing related problems in the design which commonly I faced. As I earlier mentioned that RLS methodology is tool based methodology then now a time to gives you some brief over view of the tool I used during my project.

However there are lots of EDA tools in the market right now for the same purpose of synthesis but the Intel Corporation follows some standard tool from Synopsys EDA vendor. These tools are Design Compiler (DC) and IC complier (ICC) from the Synopsys. There are lots of internal Intel Corporation tools which I could not mention here which are integrated and may be used in parallel with these tools. Here is some brief overview of both the tools from Synopsys.

6.1 Design Compiler (DC)

The Design Compiler tool is the core of the Synopsys synthesis products. Design Compiler optimizes designs to provide the smallest and fastest logical representation of a given function. It comprises tools that synthesize your HDL designs into optimized, technology-dependent, gate-level designs. It supports a wide range of flat and hierarchical design styles and can optimize both combinational and sequential designs for speed, area, and power.

Design Compiler also provides topographical technology, which allows you to accurately predict post-layout timing, area, and power during RTL synthesis without the need for timing approximations based on wire load models. It uses Synopsys placement and optimization technologies to drive accurate timing prediction within synthesis, ensuring better correlation with the final physical design.

The below figure 6.1 show a simplified overview of design compliers tasks and where

it fits in to the design flow.

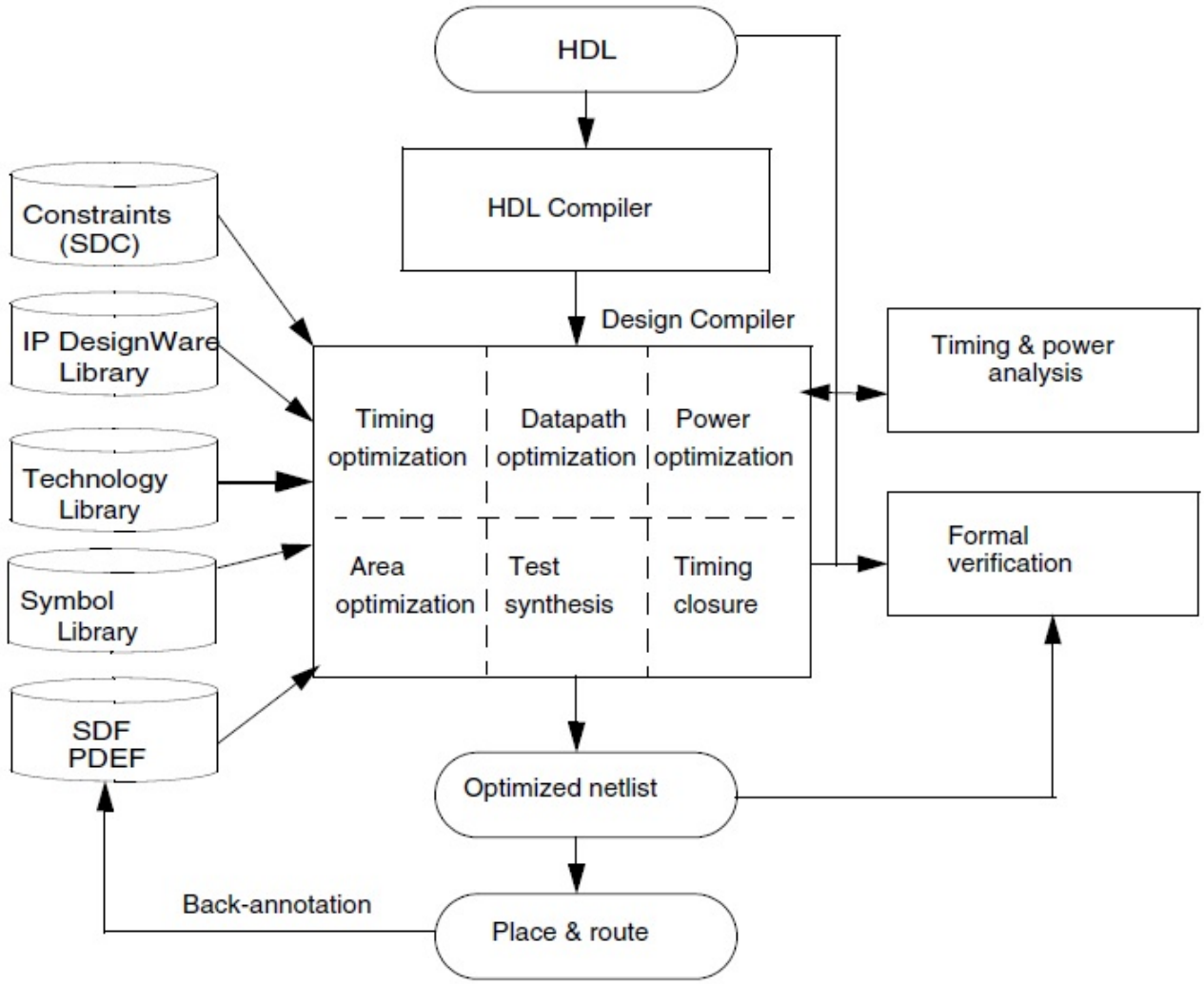

Figure 6.1: Desogn Compiler overview

These are the steps in which the synthesis process is performing.

- The input design files for Design Compiler are often written using a hardware description language (HDL) such as Verilog or VHDL.

- Design Compiler uses technology libraries, synthetic or DesignWare libraries, and symbol libraries to implement synthesis and to display synthesis results graphically. During the synthesis process, Design Compiler translates the HDL description to components extracted from the generic technology (GTECH) library and DesignWare library. The GTECH library consists of basic logic gates and flip-flops. The DesignWare library contains more complex cells such as adders and comparators.

Both the GTECH and DesignWare libraries are technology independent, that is, they are not mapped to a specific technology library. Design Compiler uses the symbol library to generate the design schematic.

- After translating the HDL description to gates, Design Compiler optimizes and maps the design to a specific technology library, known as the target library. The process is constraint driven. Constraints are the designers specification of timing and environmental restrictions under which synthesis is to be performed.

- After the design is optimized, it is ready for test synthesis. Test synthesis is the process by which designers can integrate test logic into a design during logic synthesis. Test synthesis enables designers to ensure that a design is testable and resolve any test issues early in the design cycle. The result of the logic synthesis process is an optimized gate-level netlist, which is a list of circuit elements and their interconnections.

- After test synthesis, the design is ready for the place and route tools, which place and interconnect cells in the design. Based on the physical routing, the designer can back-annotate the design with actual interconnect delays; Design Compiler can then resynthesize the design for more accurate timing analysis.

6.2 IC Complier (ICC)

The IC Compiler tool is a single, convergent netlist-to-GDSII or netlist-to-clock-tree-synthesis design tool for chip designers developing very deep submicron designs. It takes as input a gate-level netlist, a detailed floorplan, timing constraints, physical and timing libraries, and foundry-process data, and it generates as output either a GDSII-format file of the layout or a Design Exchange Format (DEF) file of placed netlist data ready for a third-party router. The IC Compiler tool can also output the design at any time as a binary Synopsys Milkyway database for use with other Synopsys tools based on Milkyway or as ASCII files (Verilog, DEF, and timing constraints) for use with tools not from Synopsys.

The IC Compiler design flow is an easy-to-use, single-pass flow that provides convergent timing closure. Figure 6.2 shows the basic IC Compiler design flow, which is centered around three core commands that perform placement and optimization (placeopt), clock tree synthesis and optimization (clockopt), and routing and post-route optimization (routeopt).

This is how the flow goes in the IC Compiler. To run the IC Compiler design flow,

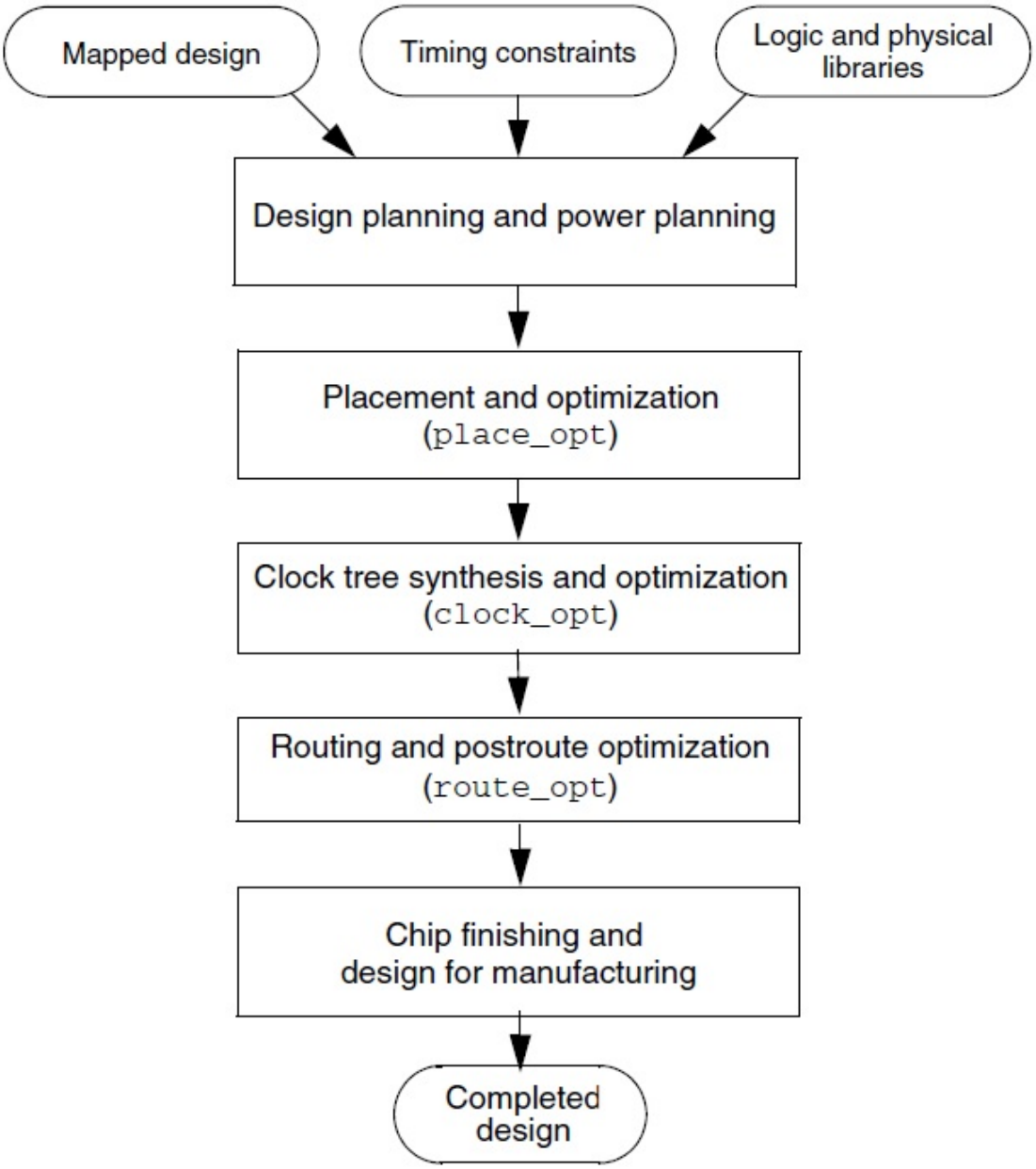

Figure 6.2: ICC overview

- Set up the libraries and prepare the design data
- Perform design planning and power planning.
  When you perform design planning and power planning, you create a floorplan to determine the size of the design, create the boundary and core area, create site rows for the placement of standard cells, set up the I/O pads, and create a power plan.
- Perform placement and optimization.
  The placeopt core command addresses and resolves timing closure for your design. This iterative process uses enhanced placement and synthesis technologies to generate legalized placement for leaf cells and an optimized design. You can supplement this functionality by optimizing for power, recovering area for placement, minimizing congestion, and minimizing timing and design rule violations.
- Perform clock tree synthesis and optimization.
  IC Compiler clock tree synthesis and embedded optimization solve complicated clock tree synthesis problems, such as blockage avoidance and the correlation between pre-route and post-route data. Clock tree optimization improves both clock skew and clock insertion delay by performing buffer sizing, buffer relocation, gate sizing, gate relocation, level adjustment, reconfiguration, delay insertion, dummy load insertion, and balancing of inter-clock delays.
- Perform routing and postroute optimization.
  As part of routing and post-route optimization, the IC Compiler tool performs global routing, track assignment, detail routing, topological optimization, and engineering change order (ECO) routing. For most designs, the default routing and post-route optimization setup produces optimal results. If necessary, you can supplement this functionality by optimizing routing patterns and reducing crosstalk or by customizing the routing and postroute optimization functions for special needs.
- Perform chip finishing and design for manufacturing tasks
  The IC Compiler tool provides chip finishing and design for manufacturing and design for yield capabilities that you can apply throughout the various stages of the design flow to address process design issues encountered during chip manufacturing.
- Save the design.
  Save your design in the Milkyway format. This format is the internal database format used by the IC Compiler tool to store all the logical and physical information about a design.

Chapter 7

Conclusion and Future scope

7.1 Conclusion

Physical design is become more complex as the integration of more functionality in the same chip is becomes basic need of every project however the TTM and product quality also have the same weightage, so the design methodology like RLS is very much helpful and best suited for the requirements. That is the reason why more number of blocks are designed by this method. However as we see in this report that RLS methodology gives the user more flexibility to change some flow controls and do some more customization using TCL scripting language.

"Best the constraints you provide, Tool will produce the Most optimized design." This is the main motivation for this design style. Timing, Power, Noise, RV and Area are most of the desired constraints of the design in this time, and Random Logic Synthesis process is assuring these all if you use the tools in a well manner.

7.2 Future Scope

Convergence of the Functional blocks by meeting the constraints of timing, power, area, noise is important for this project and I have converged four designs using this RLS method.

This project is a methodology usage so in future we can try to modify some tool related things which is basically some improvement for future which help in terms of best optimization in small number of iterations which essentially require less effort by the design engineer and one engineer can simultaneously handle more blocks. This all need knowledge of scripting language like TCL (Tool Command Language) as all the tools used in this method are using TCL interface. In order to fix the violations which are more in number in the earlier iterations we can use PERL to automate the parsing process in order to root cause and make a file with the fixes of those violations. More the

automation less the TTM and hence the chip can be produced in time. But at the same time quality is also important.

Bibliography

[1] Wikipedia.org

[2] intel confidencial documents

[3] User guide of DC and ICC

[4] http://www.facweb.iitkgp.ernet.in/ isg/CAD/SLIDES/06-VLSI-design-styles.pdf

[5] http://cc.ee.ntu.edu.tw/ lhlu/eecourses/DE/Chapter1.pdf

[6] http://www.design-reuse.com/articles/21288/set-top-box-architecture.html

[7] A Cost-Optimized Set-Top Box Architecture By Stuart Ryan, Andrew Jones, Robert Deaves STMicroelectronics R and D Ltd

[8] N.Parthibhan et. Al. (2012) Clock Skew Optimization in Pre and Post CTS, International Conference on Advances in Computing and Communications, VIT University