PHYSICAL IMPLEMENTATION OF ASIC DESIGN AND FLOW OPTIMIZATION

Major Project Report

Submitted in partial fulfillment of the requirements

for the degree of

Master of Technology

in

Electronics & Communication Engineering

(Embedded Systems)

By

Rachit Patel

(12MECE16)

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

May 2014

PHYSICAL IMPLEMENTATION OF ASIC DESIGN AND FLOW OPTIMIZATION

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(Embedded Systems)

By

Rachit Patel

(12MECE16)

Under the guidance of

External Project Guide:
Roma Rudra
SGL & Principle Engg.,
ARM Embedded Technologies Pvt. Ltd.,
Banglore.

Internal Project Guide:
Dr. N. P. Gajjar
Professor (EC Dept.),
Institute of Technology,
Nirma University, Ahmedabad.

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2014

Declaration

This is to certify that

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

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

- Rachit Patel

Certificate

This is to certify that the Major Project entitled **"PHYSICAL IMPLEMENTATION OF ASIC DESIGN AND FLOW OPTIMIZATION"** submitted by **Rachit C. Patel (12MECE16)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded Systems, Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.The results embodied in this major project, to the best of our knowledge,haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Dr. N. P. Gajjar
Guide

Dr. N.P. Gajjar
Program Coordinator

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

Dr. K. Kotecha
Director, IT

this page is left blank intentionally to accomodate company certificate.

Acknowledgements

I would like to express my gratitude and sincere thanks to **Dr. P.N.Tekwani**, Head of Electrical Engineering Department, and **Dr. N. P. Gajjar**, Coordinator of M.Tech Embedded Systems program for allowing me to undertake this thesis work and for his guidelines during the review process.

I am deeply indebted to my thesis supervisors, **Ms Roma Rudra**, Principle Engineer at ARM Embedded Technologies Pvt. Ltd. and **Dr. Nagendra P. Gajjar**, Professor,EC Department, Nirma University for their constant guidance and motivation. I also wish to thank **Mr. Vijay Kishan Narayanan**, Manager, ARM Embedded Technologies Pvt. Ltd., **Sriram Venkatesh** and all other team members at ARM for their constant help and support. Without their experience and insights, it would have been very difficult to do quality work.

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

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

- Rachit Patel

12MECE16

Abstract

Physical implementation is gateway of an ASIC design to physical/real world. It is mainly driven by PPA (Power, Performance and Area) targets. Design is implemented using EDA tools with pre-defined constraints and targets. This implementation is divided in various small but significant steps like, synthesis, floorplanning, placement, power planning, clock tree synthesis, routing, etc. The flow starts with the RTL or Verilog files of the design and gives a GDS-II database for the same as final outcome. GDS-II database of ASIC design is sent for fabrication once physical implementation achieves PPA targets with given constraints.

This thesis covers the physical implementation of Memory Management Unit (MMU). As it is a configurable design, multiple configurations of MMU are implemented to ensure achievement of PPA targets and the same are discussed in detail. Primary goal of implementation trials is to push PPA numbers. Several modifications and improvements that are done on the native implementation flow to achieve and improve PPA numbers for the design are also discussed. Several Clock Tree Synthesis experiments, their results and recommendation from that are mentioned in this document. Implementation trials of design over couple of foundry platforms and across couple of implementation nodes are also discussed here. Basic information about physical implementation flow is also provided. Descriptions related to EDA tools given here are based on implementation of design on cadence EDA tools, RTL Compiler for synthesis and Encounter Digital Implementation for rest of the flow.

Contents

List of Tables

List of Figures

Abbreviation Notation and Nomenclature

ASIC ..... Application Specific Integrated Circuit
AXI ..... Advanced eXtensible Interface
CP ..... Common Platform
CTS ..... Clock Tree Synthesis
DRC ..... Design Rule Checking
ECO ..... Engineering Change Order
EDA ..... Electronics Design Automation
EDI ..... Encounter Design Implementation
FIFO ..... First In First Out
FP ..... FloorPlan
GDS-II ..... Graphic Database System
HDL ..... Hardware Description Language
IPA ..... Intermediate Physical Address
MMU ..... Memory Managment Unit
PA ..... Physical Address
PPA ..... Power, Performance and Area
PTW ..... Page Table Walk
RC ..... RTL Compiler
RTL ..... Register Transfer Logic
STA ..... Static Timing Analysis
TBU ..... Translation Buffer Unit
TCU ..... Translation Control Unit
TLB ..... Translation Look-aside Buffer
TSMC ..... Taiwan Semiconductor Manufacturing Company
VA ..... Virtual Address

Chapter 1

Introduction

Todays world is digital world. Digital world is driven by ICs. ICs are physical form of logic. Designing an IC is not simple task. Figure 1.1 explains the design cycle of any digital circuit or IC or chip.Physical Design is one of the most important step from various steps required to produce a chip. Design of the circuit is provided as input of this step and implemented design is expected as output. This implemented design is sent to the foundry to fabricate. Design at the input side of the step is in form of logical statements and logical connections. This is usually given in Hardware Description Languages (HDLs). The design after the step of physical Design is in form of description of Gates and nets with their locations on the chip. Once the physical designing is done, it is verified for its logical equivalence. After Physical verification, it is fabricated on silicon. Chip is then sold out to the market after post silicon verifications and packaging.

1.1 Motivation

Design is passed to physical design team in RTL format. This doesnt have any physical information. It is just a logical circuit. In physical design stage, physical information is added and a complete design with physical information of every cells, macros, blocks and each net is expected as output. Output of the physical design is

usually in GDSII format. As the scope of work for this stage starts from RTL and ends on GDSII database, this stage is also known as RTL to GDSII flow.

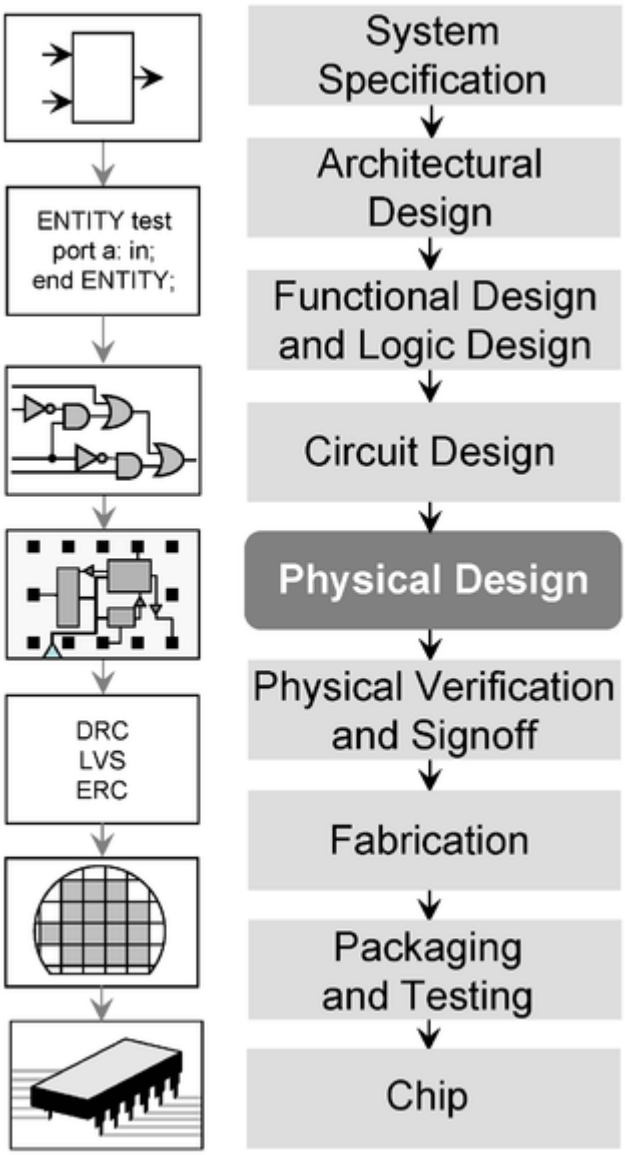


Figure 1.1: IC Design Flow [3]

RTL to GDSII flow includes many tasks like synthesis, floorplanning, routing. It is task or flow to covert a logical circuit to real circuit. It is a bridge between logical

world and real world. As the embedded industry has grown, RTL to GDSII flow is not just limited to VLSI designs. ASICs and SOCs are also equally utilizing the flow up to remarkable extent and it is increasing day by day.

1.2 RTL

RTL is acronym of Register Transfer Logic. It is used to describe synchronous digital circuits in terms of flow of digital signals. It defines all the logical operations performed on the digital signals also. It defines the direction and logic of flow of digital signal from one register to other.

It is used in Hardware Description Languages(HDL) to create higher level representation of circuit. From this representations, lower level representations can be obtained. Verilog and VHDL are such well known HDLs.

RTL does not have gate level information. Logic synthesizer in EDA tool synthesis the logic in RTL and converts it to appropriate gate.

1.3 GDS-II

GDS-II is a database format. It is an industry standard for data exchange especially for IC layout. Basically it is a binary representation of planer geometric shapes, labels and other information about layout in hierarchical form.[3] Foundries, who actually makes the IC from silicon wafer, accept data from their customers in GDSII format.

1.4 Problem Statment

Objective of this project is to take RTL of design and implement it with the physical implementation flow, convert it to GDSII and make it ready to be fabricated. This

includes all the steps that come across the flow of ASIC physical implementation i.e. synthesis, floorplanning, placement, CTS, Routing, Timing analysis, violation correction and if required, Engineering Change Order.

During the implementation of design, various experiments of optimizing the design to improve timing results, to improve area requirements and power requirements are to be carried out. Some optimizations based on the Clock Tree Synthesis are also to be carried out once design is timing and DRV clean.

Here, starting of implementation is done with a stable flow developed by Systems and Software Engineering Implementation team of ARM with the cadence Electronic Design Automation (EDA) tools, RTL compiler and Encounter Digital Implementation tools.

1.5 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, *Physical Implementation Flow* , describes the basic physical implementation flow first. Later in the chapter, all the steps of physical implementation flow are described one by one. It also covers some basic concepts and definations of terms used in Physical implementation.

Chapter 3 , *Implementation of Design* , describes the physical implementation flow used for implementing the design first. This chapter also includes information about different configurations which are implemented. It also enlighten the platforms and targets for each configurations.

Chapter 4, *Flow Optimization* , describes the optimizations done on the native flow of implementation used. It also includes some experiments and their results.

With this optimizations, it also gives some glimpses on issues faced during implementation and their solutions.

Finally in **Chapter 5** , *Conclusion and Future Scope* , concluding remarks and scope of future work is presented.

Chapter 2

Physical Implementation Flow

This chapter provides basic concepts of Physical Implementation flow which is also known as ASIC flow of Physical Design flow. It is also known as RTL to GDS-II flow. Physical Design Flow is basically a group of algorithms with several objectives like minimum area, minimum wire-length, power optimization, and many more. It also includes preparation of timing constraints. Ultimate aim of RTL to GDSII flow is to implement the design such that it meets all timing constraints and also meet target frequency if provided. Implemented design should also be DRC clean.

Figure 2.1 explains the Physical Design flow better with each steps explained later in the chapter.

2.1 Flow Requirements And Outputs

To complete the conversion of RTL to GDSII, there are some requirements of the Implementation tools. User need to provide these inputs to start the flow.

- Technology File (.techlef)
- Library Exchange Format (.lef)

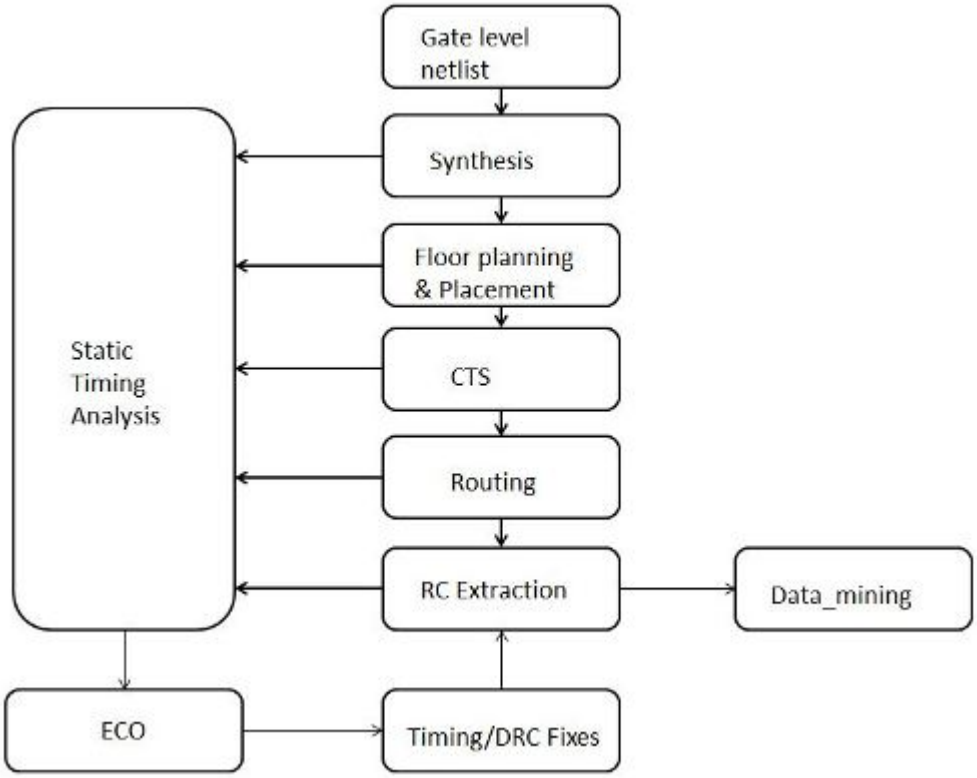


Figure 2.1: Physical Implementation Flow

- Timing and power libraries (.lib, .lib_ccs_t)
- Constraints (.sdc)
- Design Exchange Formate(.def)
- Netlist (.v, .vhd)

There are many more inputs like capacitance table file, design specific parameters, etc.

Among all those, two most important libraries are explained here.

Technology File Libraries

Technology file defines basic characteristic of cell library pertaining to a particular technology node.They are units used in the design, graphical characteristics like

colors, stipple patterns, line styles, physical parameters of metal layers, coupling capacitances, capacitance models, dielectric values, device characteristics, design rules. These specifications are divided into technology file sections.

Units for power, voltage, current etc. are defined in technology section. The color section defines primary and display colors that a tool uses to display designs in the library. Stipple pattern are defined in stipple sections. Different layer definitions like its current density, width etc are defined in layer section. Fringe capacitances generated by crossing of interconnects are defined in fringe cap section.

Similarly several other specifications like metal density, design rules that apply to design in library, place and route (P&R) rules, slot rule, resistance model are defined in their respective sections.

Standard Cell/IO/Special Cell Libraries

A standard cell library is a collection of pre designed layout of basic logic gates like inverters, buffers, ANDs, ORs, NANDs etc. All the cells in the library have same standard height and have varied width.

These reference libraries are technology specific and are generally provided by ASIC vendor like TSMC, Artisan, IBM etc. Standard cell height for 130 TSMC process is 3.65 um. In addition to standard cell libraries, reference libraries contain I/O and Power/Ground pad cell libraries. It also contain IP libraries for reusable IP like RAMs, ROMs and other pre-designed, standard, complex blocks.

The TSMC universal I/O libraries include several power/ground cells that supply different voltages to the core, pre-drivers and post drivers. Internal pull-up or pull-down is provided to some cells in I/O libraries.

After the flow is executed, following main outputs are received from the tool.

- Parasitic file (.spef)
- Routed Netlist (.v)
- Physical Layout (.gds)
- Design Exchange Format (.def)

These files are sent to the foundry for fabrication of the design.

2.2 Synthesis

Process of transforming HDL design to technology specific gate-level netlist is known as synthesis. This is performed in three substeps. In first step, tool translates the RTL to a technology independent representation. After that, in second step, Optimization is done on the logic. This step is important as it significantly reduces the size of the design as repeated or unused logics are removed from the design. Also hard efforts are made to combine logics and reduce the size. In the Third step, this optimized logic is mapped to logic cells using technology dependent libraries.[10]

2.3 Floorplanning

Floorplanning is the process of positioning blocks on the die or within another block, thereby defining routing areas between them. It is an important process of creating and developing a physical model of the design in the form of an initial optimized layout. Because floorplanning significantly affects circuit timing and performance, especially for complex hierarchical designs, the quality of your floorplan directly affects the quality of your final design.

Based on the area of the design and the hierarchy, a suitable floorplan is decided upon. Floorplanning takes into account the macros used in the design, memory, other IP cores and their placement needs, the routing possibilities and also the area of the entire design. Floorplanning also decides the IO structure, aspect ratio of the design. A bad floorplan will lead to bad utilization of die area and routing congestion.

- **Size of Layout**

  The first step in floor planning is to define the outline of the layout. If the layout is rectangular, only the length and the width of the layout are required. More coordinates are needed to define the outline of a rectilinear layout, such as an L-shape layout. Most of the P&R tools do not alter the size of the layout specified by the user.

- **Partitioning**

  Partitioning is a process of dividing the chip into small blocks. This is done mainly to separate different functional blocks and also to make placement and routing easier. Partitioning can be done in the RTL design phase when the design engineer partitions the entire design into sub-blocks and then proceeds to design each module. These modules are linked together in the main module called the Top Level module. This kind of partitioning is commonly referred to as Logical Partitioning.

- **Core Area**

  The core area is usually defined by specifying the distance between the edge of the layout and the core. All standard cells must be placed in the core area. I/O pads and macros do not have this restriction although it is common to place macros in the core area. The area outside the core can be used to place the I/O pads, the I/O pins, and the core power rings.

  Standard cells are placed in rows, similar to placing books on a book shelf. The rows are called cell rows and are drawn inside the core area. All cell rows have the same height. There are three common ways to arrange the cell rows.

The most common approach for layout with more than three metal layers is to flip every other cell row which does not leave a gap between the cell rows. The second configuration is to flip every other cell row, but leave a gap between every two cell rows. The purpose of the gaps is to allocate more resources for the inter-connect routings. The last configuration is to leave a gap between every cell row, and not flip the cell rows. This configuration is useful when only two or three metal layers are available for routing.

- **Placements of IO Pads and IO Pins Geometries**

  For a chip-level layout, the next step is to place the IO pads. The P&R tool can fill the gaps between the pads with pad filler cells and corner cells. For a block-level layout, the user needs to define the location and geometries (size and metal layer) of every IO pin.

- **Placements of the Hard Macros**

  The floorplan is complete if the design does not contain any hard macro. Otherwise, the next step is to place the hard macros. Placing the hard macros may not be a simple task. A good macro placement has the following qualities:

  - provides a compact layout
  - does not cause routing congestion
  - does not make timing closure difficult
  - allows robust power routing between the power pads and the macros

  The biggest challenge in placing the macros is in assessing the quality of the floorplan, which cannot be achieved without executing the rest of the flow. Thus, floorplanning is an iterative and time consuming process. The trick in performing floorplanning is to shorten the turn-around time of the iterations, and to reduce the number of iterations.

2.4 Power Planning

All connections to the power and ground nets are routed during power planning. The only exception is the tie-high and the tie-low nets. Most P&R tools use a dedicated router to route the power nets. All power routings created by the power router are considered pre-routes, and are not modified by the detailed router when the signal nets are routed.

The key consideration for power planning is:

- an acceptable IR-drop from the power pads to all power pins
- meeting electro-migration requirements
- does not result in routing congestion
- compact layout

A power plan consists of several types of power structure. A typical sequence to construct the power structures includes:

- core power rings are routed first
- core power pads are connected to the core power rings
- the power rings are added around the macros where necessary
- vertical stripes and horizontal stripes are added to reduce the IR-drop at the power rails of the standard cells and the macros
- the power pins of the hard macros are tapped to the core rings or the power stripes
- if tie-high and tie-low cells are not used, the tie-high and tie-low inputs to the hard macros and IO pads are tapped to the power structures

- the power rails for the standard cell are added to the power plan. The power rails can tap the power from the core power rings, the power stripes and the macro power rings.

2.5 Placement

In placement stage, standard cells are placed on the core area. Placement ensures that routing can be done for placed cells. Also special care about critical net is taken that net delay for such nets become minimum. Placement uses various partitioning algorithms and performs an optimized placement based on length of interconnect between partitions.

Placement is further divided in three parts.[2]

- Global Placement Decides placement of partitions and blocks on the die area
- Detail Placement Decides place of standard cells inside partition or block
- Legalization - Align placed cells to manufacturing grids

Placement may lead to an un-routable design, i.e., routing may not be possible in the space provided. In that case, another iteration of placement is necessary. To limit the number of iterations of the placement algorithm, an estimate of the required routing space is used during the placement phase. Good routing and circuit performance heavily depend on a good placement algorithm. This is due to the fact that once the position of each block is fixed, very little can be done to improve the routing and the overall circuit performance.

Before the start of placement optimization all Wire Load Models (WLM) are removed. Placement uses RC values from Virtual Route (VR) to calculate timing. VR is the shortest Manhattan distance between two pins. VR RC values are more accurate than WLM RC values.

2.6 Clock Tree Synthesis

Clock is the most critical net in digital circuits. All the functioning of digital circuits are based on clock only. Also clock is the reference for all timing measures of design. As every flip-flop of design needs to be clocked, clock is having a high fan-out. Due to all these reasons, Clock is routed first in the design. After Clock tree building, all other routings are done. Prime objectives of Clock Tree Routing are minimizing clock skew, Minimizing insertion delay, minimizing power consumption on clock tree as clock would have the most dynamic power consumption. Clock tree is synthesized by inserting buffers or inverters or both on clock path.

Clock is propagated after placement because the exact physical location of cells and modules are needed for the clocks propagation which in turn impacts in dealing with accurate delay and operating frequency and clock is propagated before routing because when compared to logical routes, clock routs are given more priority. This is because, clock is the only signal switches frequently which in acts as source for dynamic power dissipation. [11]

Though wide range of clock routing algorithms are available, EDA tool chooses the optimized algorithm automatically and it only shows the critical paths after propagating the tree. If a design results in negative slack, increasing the clock timing is an easy way but changing the clock period changes the operating frequency. Solving the negative slack without changing the clock period is possible by up-sizing or down-sizing the cell in critical paths.

2.7 Routing[5]

Routing is the process of creating physical connections based on logical connectivity. Signal pins are connected by routing metal interconnects. Routed metal paths must

meet timing, clock skew, max trans/cap requirements and also physical DRC requirements.

Routing is performed mainly in three major steps.

- **Global Routing** - assigning net to specific metal layer and global routing cells. Algorithm for global routing are designed such that they avoid routing congestion. It also respects routing blockages, macros, pre-routed nets like Power, Ground or clock.

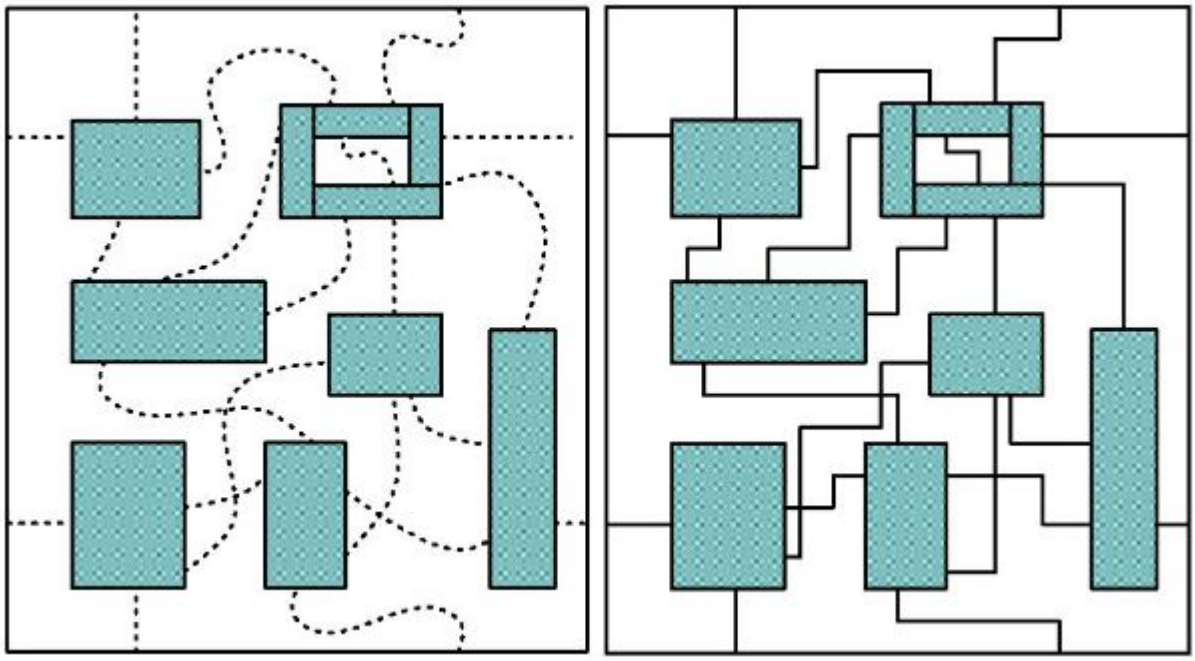


Figure 2.2: Global/Detail Routing[5]

- **Track Assignment** - In this step, each net is assigned a specific track. Actual metal trace are laid down. Algorithms try to make nets as long and straight as possible. It doesnt care about DRC in this step.
- **Detailed Routing** - This step tries to fix DRC violations after track assignment. At the end of this step, completely routed design netlist is prepared. Figure 2.2 shows difference of global routing and detail routing.

2.8 Engineering Change Order(ECO)[7]

ECO is the process of inserting a logic change directly into the netlist after it has already been processed by an automatic tool. ECOs are usually done to save time. It usually happens that some part of the design is not optimized by the automated tool due to some reasons. To implement the whole flow from synthesis , placement , CTS and routing takes a huge amount of time. Instead of that, some small manual changes can be made on the netlist of the routed design. This will affect just that portion of the design keeping whole design unchanged. Once the changes are made, only that part of the design is routed again. This takes very little time than performing whole physical design flow on the design RTL. However this method can not be used if huge modifications are to be made.

2.9 Static Timing Analysis Concepts

Timing analysis is one of the most important part of the flow. After each stage of the flow, timing needs to be checked and analyzed. As prime objective of implementation is to meet timing, it is critical to perform such check after every stage.

There are mainly two type of timing checks.

a. Setup Check

b. Hold Check

Setup Time is Minimum amount of time that data signal should be steady before clock event to be sampled reliably.

Where **Hold Time** is defined as, Minimum amount of time that data signal should be steady after clock event to be sampled reliably. Figure 2.3 provides better idea about setup and hold time.

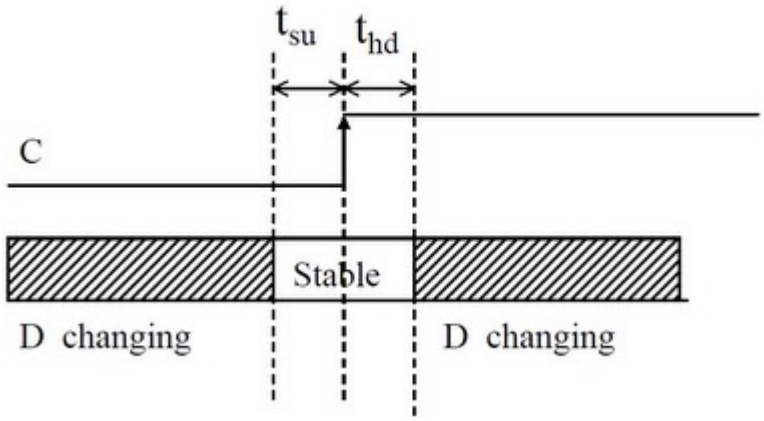


Figure 2.3: Setup and Hold[6]

Slack is defined as difference between required time of data and arrival time of data. Negative value of slack indicates that timing is violated.

The worst negative slack in design is called **WNS** whereas addition of all negative slacks in design is termed as **Total Negative Slack (TNS)**.

2.9.1 Setup Check[6]

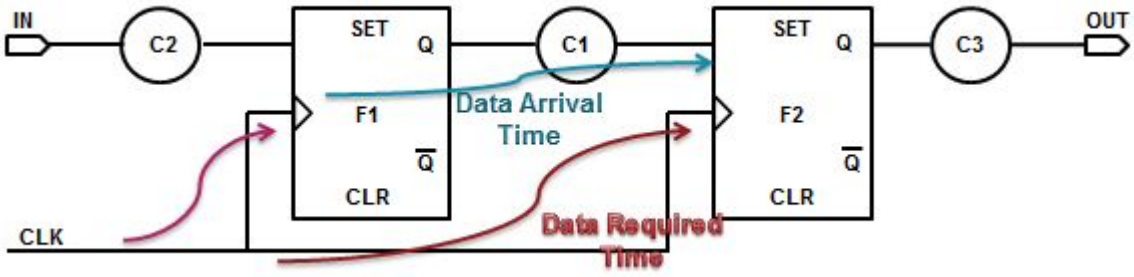


Figure 2.4: Setup and Hold Slack Timings

As shown in figure 2.4, note that data is launching from FF1 on the positive edge of clock. At FF2, data is coming from combinational logic C1. Data is captured at FF2 on positive edge of clock. So data path is FF1 ⇒C1 ⇒ FF2.

So, for setup analysis at FF2, data should be stable Ts time before positive edge of clock at FF2. Tsis setup time of FF2. So the setup slack can be counted by,

Setup slack = Data Required Time - (Data arrival Time + Ts)

Negative value of setup slack indicates timing violation in path.

2.9.2 Hold Check[6]

For the shown figure 2.4, data should be stable for Th time after positive edge of clock at FF2. Th is hold time of FF2. Hold slack can be counted by,

Hold Slack = Data Arrival Time - (Data Required Time + Th)

Negative value of hold slack indicates timing violation on the path.

Setup check is done on the next cycle of the clock where hold check is performed on the same clock cycle.

2.10 Parasitic Interconnect Corners

Parasitics can be extracted at many corners. These are mostly governed by the variations in the metal width and metal etch in the manufacturing process. Some of these are:

- **Typical**
  This refers to the nominal values for interconnect resistance and ca- pacitance.
- **Max C**
  This refers to the interconnect corner which results in maximum capacitance.

The interconnect resistance is smaller than at typical corner. This corner results in largest delay for paths with short nets and can be used for max path analysis.

- **Min C**

  This refers to the interconnect corner which results in minimum capacitance. The interconnect resistance is larger than at typical corner. This corner results in smallest delay for paths with short nets and can be used for min path analysis.

- **Max RC**

  This refers to the interconnect corner which maximizes the interconnect RC product. This typically corresponds to larger etch which reduces the trace width. This results in largest resistance but corresponds to smaller than typical capacitance. Overall, this corner has the largest delay for paths with long interconnects and can be used for max path analysis.

- **Min RC**

  This refers to the interconnect corner which minimizes the interconnect RC product. This typically corresponds to smaller etch which increases the trace width. This results in smallest resistance but corresponds to larger than typical capacitance. Overall, this corner has the smallest path delay for paths with long interconnects and can be used for min path analysis.

2.11 Design Rules[7]

Design Rules are a series of parameters provided by semiconductor manufacturers that enable the designer to verify the correctness of a mask set. Design rules are specific to a particular semiconductor manufacturing process. A design rule set specifies certain geometric and connectivity restrictions to ensure sufficient margins to account for variability in semiconductor manufacturing processes, so as to ensure that most of the parts work correctly.

Most basic rules are width and spacing rules.

A *width rule* specifies the minimum width of any shape in the design.

A *spacing rule* defines the minimum distance between two adjacent objects.

There are many other rules for multilayer designs. There are also some generalized rules which apply to both single layer and multi layer designs. Examples of such rules are, minimum area rule and antenna rules.

These design rules are provided by semiconductor manufacturers.

Main objective of Design rule checking is to achieve high overall yield and reliability for the design. If design rules are violating in the design, it may happen that design does not provide the functionality what it should.

2.12 PVT

PVTs are inter-chip variation which depend largely on external factors like: the ambient temperature, the supply voltage and the process of that particular chip at the time of manufacturing.

- **Variation in Process** : There are millions of devices (standard cells), and probably billions of transistors packed on the same chip. You can expect every single transistor to have the same process or the channel length. If manufactured chip is with worst process, it means that the channel length tends to deviate towards the higher side. This variation may be more for some transistors and less for some. It can be a ponderous task to quantify this variation between the transistors of the device, and is often modeled as a percentage deviation from the normal.[12]

- **Variation in Voltage** : : All the standard cells need voltage supply for their operation. And voltage is usually 'tapped' from the voltage rail via interconnects which have a finite resistance. In two parts of the chip, it is fairly probable for the interconnect length to be different, resulting in a finite difference in the resistance values and hence the voltage that the standard cells actually receive. So the voltage received by standard cell might be different.

- **Variation in Temperature** : Some parts of the chip can be more densely packed or might exhibit more active switching as compared to the other parts. In these regions, there is a high probability of the formation of localized 'HOT SPOTS' which would result in increased temperature in some localized areas of the chip. Again, this difference might not be order of a few degree centigrade, but can be significant.[12]

2.13 Summary

This chapter describes basic physical implementation flow and steps in it. Also, each step is explained in detail one by one. Some basic concepts used in physical implementation flow are also described. Chapter also defines some basic terms used in the flow. Overall, it provides almost all necessary information about the physical design flow.

Chapter 3

Implementation of Design

In this chapter, actual work that has been done on various designs with different platforms and different libraries is explained. This chapter focuses on the details of the design, its configurations and implementation of all configurations. Starting with basic information about the design itself and flow employed, chapter will be talking about various configurations of the design and their implementations.

3.1 Design Overview

The MMU-500 is a system-level Memory Management Unit (MMU),that translates an input address to an output address, by performing one or more translation table walks. It supports both ARM v7 and ARM v8 architectures.

Translation occurs in one or two stages. In first stage, input virtual address(VA) is translated to either output physical address(PA) or intermediate physical address(IPA). If in first stage, address is translated to intermediate physical address, than second stage translates intermediate physical address to physical address.

Design can majorly be divided in three main parts.

Translation Buffer Unit (TBU): TBU contains Translation look aside buffer that caches page tables. Each master has its own TBU so that it is local to the master rather than local to memory management unit.

Translation Control Unit (TCU): TCU is a control unit that controls the address translations. There is only one TCU in design.

Interconnect : This connects multiple TBUs in design to the TCU.

This design is a configurable design. User can configure the number of the TBUs, address bus bits and number of stages to use for translation depending upon the requirements.

3.2 Design Implementation Flow

For implementing the design with EDA tools, it is divided in various different stages. Each stage has its own significance. All the stages are listed and explained below with their significance. This flow is on the EDA tools provided by cadence. Similar flows are available with other EDA tool vendors also.

For the cadence EDA tools, there are two main tools used in the flow[1].

- RTL Compiler
- Encounter Design Implementation

RTL Compiler is a synthesis tool. This is used to compile and synthesis the design which is in RTL format to gate level netlist. This includes synthesis to technology independent representation, logical optimization and mapping to the target technology gates.

Encounter Design Implementation tool is implementation tool. This is used to implement the gate level netlist of the design to the routed netlist of the design. This includes many steps like placement, placement optimization, clock tree synthesis, routing and post route optimizations. To perform critical tasks of Clock tree synthesis and routing, internally EDI is using two different tools called ccopt and nanoroute.

The Flow with these two tools is as follow[8][9].

- ***syn_map* :** This step uses RTL Compiler tool. In this step, RTL is taken as input with constraints, technology libraries, technology lefs, design names, target frequency and many more parameters. In this step, RTL is converted to the technology independent representation first. After that, the logic is mapped to the available cells in the technology libraries. Output of the flow is synthesized mapped gate-level netlist of the design with constraints.

- ***syn_incr* :** This step uses RTL Compiler tool. The tool tries to optimize the logic synthesized and mapped with highest optimization level. This ensures that whatever change the tool makes, it will help the design in overall cost.

- ***syn_placed* :** In this step, both the tools, RTL Compiler and EDI, are used. This step is similar to placement aware synthesis. As it is placement aware, floorplan needs to be provided at this step. Tools does the initial placement of the cells and then optimize the placement. This step also includes power planning. Power and ground strips are laid on the floorplan in this step. Output of this step is gate-level netlist, constraints and DEF file of the design which has information about places of pins, macros and flip-flops in the designs.

- ***init/place* :** This are two small steps. Init is a very small step which is used to change the design information from RTL Compiler tool format to EDI supported information format. In place step, tool takes the information from previously placed database and optimize the placement based on timing requirements of

the design.

- ***preCTS*** *:* Performs timing optimizations on placed design before the clock tree is built. It checks and repairs the DRVs and setup violations.

- ***CTS*** *:* Clock tree is built in this step. As clock has to be balanced and properly skewed, it is built first and routed first. EDI uses a separate tool called ccopt for clock tree synthesis. ccopt is acroname of Concurrent Clock Optimizer. This tool builds the clock tree and give that to EDI tool. EDI tool then routes the clock tree based on non-default rules defined for clock tree routing. Also it routes with the shielding of clock net if required and specified.

- ***postCTS*** *:* Performs timing optimizations on design whose clock tree has been created. It checks and repairs design rule violations and setup timing violations. If there are any change due to addition of useful skew in the design, tool performs ECO route. Once setup optimization is done, a round of hold optimization is also executed.

- ***route*** *:* In this step, routing of the design is performed. All the nets are routed and all logical connections are made physical connections.

- ***postroute*** *:* performs timing optimization on design whose routing is done. It checks and repairs design rule violations, glitch violations and setup violations based on Signal Integrity(SI) and delays. Once setup optimization is done, a round of hold optimization is also executed.

As mentioned earlier, design is a configurable design. So, to prove that we are achieving the target, we need to implement it for several possible configurations and ensure that we are achieving our targets for them. To ensure that, smallest and largest configurations must be proved with several other configurations. Such configurations are listed and explained below with its descriptions and summary of implementations at last.

3.3 Configuration 1

Configuration1 was a startup configuration to the physical implementation. This configuration had five Translation Buffer Units and one Translation Control Unit. As the size of the configuration was small, it was implemented flat. Flat implementation is implementation where all the hierarchies and modules of the design are implemented in same implementation flow. This design also had 11 RAMs.

As physical implementation flow is combination of numbers of scripts and files, implementation of configuration 1 provided very good understanding about the flow, scripts and use of scripts.

Implementation of configuration 1 was on TSMC platform and 28 nm technology. This design was implemented targeting high performance of the design.

Implemented design needed some floorplan modifications to meet target frequency. Different trials made on floorplan were,

- Decreasing the floorplan by 10%
- Increasing clock frequency target by 8%

From static timing analysis of the results of all the trials, conclusion came out that for the given core area of the floorplan, given target frequency can be met but increased target frequency can not be achieved. To achieve higher clock frequency, core area has to be increased.

3.4 Configuration 2

Configuration 2 is configuration with total 32 Translation Buffer Units and one Translation Control Unit. Among these 32 Translation Buffer Units, 31 Translation Buffer

Units are asynchronous to the Control Unit where one TBU is synchronous to the TCU.

Here, all the 31 asynchronous TBUs are same instances but called multiple time in the design. As implementing all of them 31 times in design would increase the turn around time of design in terms of number of days, hierarchical implementation is used. In hierarchical implementation of the design, first a hierarchy of design is implemented. Once the hierarchy is implemented and provides timing, DRC and power clean implementation, LEF and LIBs of that are generated. In this configuration, asynchronous TBU is implemented as separate design first.

Once all the hierarchy are implemented, implementation of complete design starts. In this implementation, the hierarchy, which are implemented before, are treated as Macros. Macros are previously implemented blocks in the design. While implementing the whole design, you need not to implement those hierarchies again. However, LEF and LIBs of macro has to be provided to the tool to integrate it at the top level of design.

Implementation of configuration 2 was on TSMC platform and 28 nm technology. We have implemented the design with two different focuses. Once with High performance and second with low power.

3.5 Configuration 3 , 4 and 5

Configuration 3, 4 and 5 are also the configuration with 32 TBUs and one TCU. Difference is in the size of the design. This are bigger design and hence has high capacity of address translations. Here also one TBU is synchronous with TCU and other 31 TBUs are asynchronous to the TCU. Configuration 5 is bigger in size and in terms of logic than configurations 3 and 4. But as the implementation and its requirements

are similar, it is put under one section.

Similar to configuration 2, this design are also implemented hierarchically. Asynchronous TBUs are same instances called multiple time in the design. So, asynchronous TBU is implemented first. This implemented TBU is than plugged in as macro in implementation of top level design.

Configuration 3, 4 and 5 were implemented on TSMC platform and 28 nm technology. Implementation was done with high performance targets.

Floorplans of configurations 3, 4 and 5 which are having one TCU and one synchronous TBU with 31 asynchronous TBUs are similar. Here, all the TBUs are placed far away from the TCU. This is done to make sure that if it is used as a module in any system and system still works fine. TBUs are to be placed near to the memory accessing unit. This TBU will be talking to TCU asynchronously. So placing them at different distances in the floorplan makes sure that when it is integrated with the system, even though the TBUs are apart from the TCU and at different distances, it still meets timings and does not fail.

3.6 Configuration 6

Configuration 6 is the smallest configuration of all. This has only 1 TBU and one TCU. TBU here is synchronous to the TCU. As the design size is small, this configuration was implemented with flat implementation methodology. This configuration has 2 RAMs. As the RAM sizes are small, RAMs have not been used as macro in this configuration.

This configuration has been implemented on TSMC foundry platform with High Performance Target.

3.7 Configuration 7

Configuration 7 is almost similar to the Configuration 1. Only difference between these two configurations is that this has got some logic that is operated on half clock. Apart from that, number of TBUs are 4, among which 3 are asynchronous and 1 is synchronous.

This configuration is also implemented with flat implementation methodology as design size is small. Target foundry platform was TSMC and High Performance Target.

3.8 Configuration 8

Configuration 8 is smaller configuration. This has total 5 TBUs and one TCU. Among 5 TBUs, one TBU is synchronous to TCU and other four are asynchronous. Here, all the TBUs are different in logic. As they are not same, hierarchical method of implementation can not be used. So, configuration5 has been implemented flat. This configuration had 13 RAMs from which 11 RAMs are in TCU and TBU0 and TBU1 has 1 RAM.

This configuration is implemented on two different foundry platforms with total thee different targets. These targets are listed as,

- TSMC foundry platform, High Performance Target
- TSMC foundry platform, Low Power Target
- CP foundry, Lowe Power Target

To implement the configuration on CP foundry platform had number of challenges and issues. First of all, all the libraries needed to be changed. This includes following libraries.

- Technology File

- Library Exchange Format files
- Timing and Power Libraries
- Constraints

After making required changes also, implementation does not provide acceptable results. Issues and their solutions after making all those changes are discussed in next chapter.

Configuration	**No of TBUs**	**Target Platform**	**Implementation Methodology**
Configuration 1	5	TSMC 28 HPM	Flat
Configuration 2	32	TSMC 28 HPM	Hierarchical
	32	TSMC 28 LP	Hierarchical
Configuration 3	32	TSMC 28 HPM	Hierarchical
Configuration 4	32	TSMC 28 HPM	Hierarchical
Configuration 5	32	TSMC 28 HPM	Hierarchical
Configuration 6	2	TSMC 28 HPM	Flat
Configuration 7	5	TSMC 28 HPM	Flat
Configuration 8	5	TSMC 28 HPM	Flat
	5	TSMC 28 LP	Flat
	5	CP 28 LP	Flat

Table 3.1: Implemented Configurations

Complete summary of the implemented configurations is given in table 3.1

3.9 Summary

Chapter provides the description of physical implementation flow used for implementing the design with explaination of significance of each step and EDA tool used for that step. Also various configuration of the design which are implemented, are described with a brief information about design which is implemented.

Chapter 4

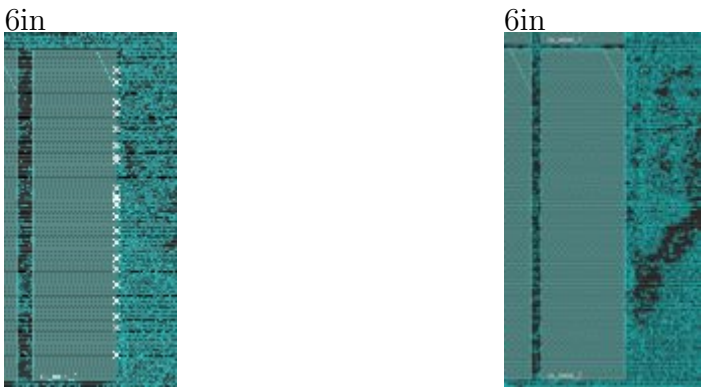
Flow Optimization

This chapter is focusing on the flow optimization trials and suggested recommendations. It is an iterative job to design a completely perfect flow for physical implementation of the design. There are always certain points in the flow where you can do various experiments and get the best results for the design. Some of the changes and experiments that I have tried are explained in this chapter with recommendations at last. However that should be noted that this results and recommendations are not always true for all type of designs. At the end of the chapter, some issues faced during implementation and their solutions are presented.

4.1 Power Grid Generation

Power planning is done in such a way in design that each of the standard cells gets power and ground supply. To ensure that, power and ground strips are generated in mesh structure. Power mesh is very dense in lower metal layers. It may be less dense in higher metal layers according to power supply structure used.

In the native flow, power strips were being generated before the placement of standard cells and unplaced macros. Due to this, power and ground strips were generated before we have exact locations of the macros. As tool is free to place the macros any-



(a) (b)

Figure 4.1: Effect of changing power planning place

where, it would place it at the best place for timings. Sometimes, the power routes are passing from the same locations. This results in DRC violations. Fig 4.1a shows such DRC violations.

To overcome this problem, I changed the flow. In the new flow, placement of unplaced macros is done first. Than power strip is generated. As now tool has the location and dimensions of the macro, power strips are not generated over that region and there are no DRCs generated due to power strips at the later stages of design.

Difference in the same design for this changes in flow is shown in figure 4.1 . Figure 4.1a is showing power strips of native flow whereas figure 4.1b shows power strips of modified flow.

4.2 CTS Experiments

As explained in chapter 2, Clock Tree Synthesis is one of the most essential and critical step in physical implementation flow. As we can achieve very good target frequency with proper clock tree synthesis, it is good to have a clock tree that is helpful to meet your timing requirements. An optimum clock tree can be synthesized with

the help of proper setting and control of clock skew, use of buffers and/or inverters, drive strength of buffers and inverters and various non-default routing methods for clock nets.

To get the most of the benefit from Clock Tree Synthesis, some experiments are done with respect to clock skew limits, usability of buffers or inverters in clock tree synthesis, drive strength of usable buffer and inverters, clock net shielding and Non-default routing rules.

Here are some basic definitions for the terms that are used normally while Clock Tree Synthesis.

Skew : Difference in arrival time of clock between two sequential registers.

Positive Skew : If capture clock comes late than launch clock, its positive skew.

Source Latency : delay from the clock origin point to the clock definition point in the design.

Insertion delay : delay from the clock definition point to the clock pin of the register.

4.2.1 Experiments

Four different experiments are done on Clock Tree Synthesis to get the optimum results for the design. These experiments are giving various constraints and limitations to EDA tool to build clock tree. It includes parameters like drive strength of cells, useful skew, usable cells to build clock tree, maximum wire length, clock shielding and non-default rules for clock net routings.

Experiment 1

- **Drive Strength :** upto 13x
- **Useful Skew Limit :** 20%
- **Usable cells for CTS :** Buffers
- **Wire Length Limit :** 180 um
- **Shielding :** No
- **NDR for Clock nets :** 2W_3S

Experiment 2

- **Drive Strength :** upto 13x
- **Useful Skew Limit :** 20%
- **Usable cells for CTS :** Buffers
- **Wire Length Limit :** 215 um
- **Shielding :** YES (VSS)
- **NDR for Clock nets :** 2W_2S

Experiment 3

- **Drive Strength :** upto 13x
- **Useful Skew Limit :** 20%
- **Usable cells for CTS :** Inverters
- **Wire Length Limit :** 215 um
- **Shielding :** YES (VSS)

- **NDR for Clock nets :** 2W_2S

Experiment 4

- **Drive Strength :** upto 9x
- **Useful Skew Limit :** 10%
- **Usable cells for CTS :** Buffers
- **Wire Length Limit :** 215 um
- **Shielding :** YES (VSS)
- **NDR for Clock nets :** 2W_2S

4.2.2 Results

In physical implementation, there is no straight forward result like pass and fail. So, to decide which experiment is giving better results, its better to have results in form of comparison table. Table 4.1 is enlisting all the parameters that affects the implementations.

Parameter	Exp 1	Exp 2	Exp 3	Exp 4
Area Related				
Clock Area	21045.99	22745.16	22830.46	23115.74
Clock Tree Cells	15305	15764	19344	16195
Insertion Delay & Skew				
Minimum Insertion Delay	0.961	1.029	0.989	1.105
Average Insertion Delay	1.269	1.35	1.331	1.403
Maximum Insertion Delay	1.37	1.445	1.403	1.44
Clock Skew	0.295	0.304	0.304	0.209
Timing Related				
WNS	-0.121	-0.187	-0.166	-0.089
TNS	-37.713	-94.438	-73.65	-22.293
Violating Paths	3982	3630	3377	2630

Table 4.1: CTS Experiments Results

4.2.3 Recommendation

From the results in 4.2.2 , it is clear that Experiment 1 seems to be better in terms of area, cell counts ans insertions delay. At the same time, Experiment 4 is having best timing numbers and clock skew among all at the cost of insertion delay, clock area and cell counts.

From the experiments, it is recommended to use buffers of drive strength upto 9X with skew limit of 10% to synthesize clock tree. Also should use clock shielding with Non-default rule of double width double spacing for clock tree routing to get the best timing numbers.

4.3 Issues and Solutions

Physical implementation flow is not a straight forward flow even with the matured flow for the design. There are modifications required in every implementation. The flow includes number of commands and each command may be having number of parameters to be set properly. If only one parameter is not set properly, implementation might end up with very bad timing or DRC violation numbers. Some of the issues might be related to EDA tools also. Here some of the issues are mentioned and also its solutions.

4.3.1 Placement Issues

While doing the placement, tool doesn't always puts the cells at the best location or it might happen that the logic cells which need to be placed nearby each other, are placed away. This kind of placement can lead to routing congestions or timing violations after routing stage in the flow.

To overcome such issue of placement, placement bounds are used. Placement bounds

are the bounds which guides the EDA tool that the logic coming under certain hierarchy has to be placed in some area. To create a placement bound, first a group of instances is created. This group has all the instances which needs to be placed nearby. Once the group is created, location of the group on the floorplan is decided. This ensures that the placement of the cells in the group takes place under the area that user has specified. This resolves the issue of spreading out of logic cells in placement.This can be done using following chain of commands[9] in cadence EDI tool.

```
createInstGroup <Name of groupe>
addInstToInstGroup <Name of Groupe> <name of instance/hierarchy>
createRegion <Name of Groupe> <x1 y1 x2 y2>
```

Here, x1 and x2 are starting and ending points of region in x direction and y1 and y2 are starting and ending point in floorplan in y direction respectively.

The figure 4.2 shows the effect of creating a placement bound on the placement of the cells.

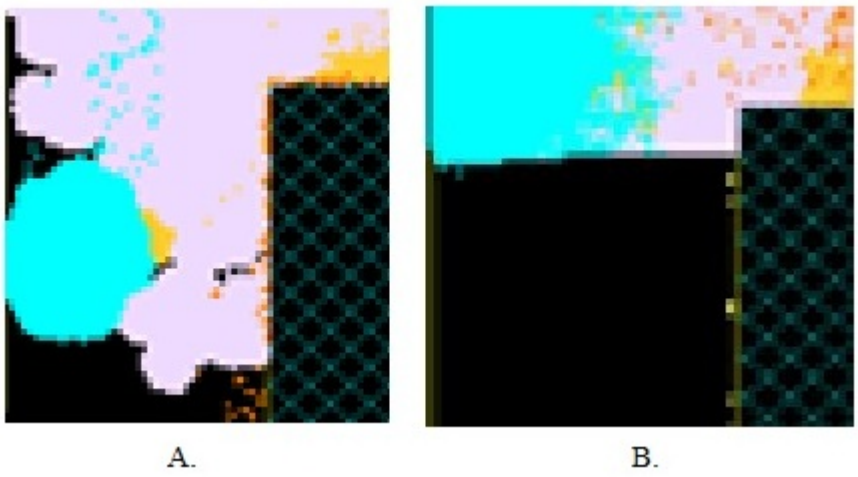


Figure 4.2: Placement Bound Effect (A. Placement without Bounds B. Placement with Bounds)

The other issue related to placement done by EDA tool is, tool usually tries to place cells such that design can work on maximum timing. While doing this, it might end up with the placement where number of logic cells in some area becomes high and some area are completely left unused. This kind of placement can lead to routing congestion in the design.

This issue can be resolved by using the EDA tool properly. All EDA tools provide various options for placement optimization. To optimize the placement such that the density of area doesn't exceed some amount, a parameter should be set. Different EDA tools have different commands but here for example, command for cadence EDA tool is,

```
setPlaceMode -maxDensity {value}
```

This command will optimize the placement of cells such that the density in the area doesn't go beyond specified value.

4.3.2 Cross Talk Issues

As nets in the chip are nothing but metal wires, they do have normal characteristics of metal wires like electric fields and magnetic fields. Dimensions of these nets are very small and so the effects will be very less but it will be significant. If two long nets are placed near to eachother on the same metal layer, one of them would be dominating the second. In this scenario, second net will be facing cross talk from first net.

As clock is the most critical net in design and it would be switching twice a clock period, cross talk due to clock net is very strong and effective. Also if there is some other net which is dominating clock net, clock will be disturbed. This in turns results in bad timing effects on all over the chip.

to prevent clock nets from cross talk and cross talk due to clock nets, two techniques are used.

- ***Shielding***[7]

  In shielding, similar to shielding on regular wire, clock net is shielded with Ground nets. This ground nets are laid on the same metal layer to clock and on both the side of clock net. Due to shielding, clock net is prevented from cross talk and also it can not produce cross talk to nearby nets.

  To command the tool to generate the shielding around the clock nets, it has to be defined at various stages. One, we need to specify that to configure ccopt tool to do shielding and second, we have to provide information to nanoroute tool to do shielding. This can be done by specifying following commands[9].

  ```
  setCTSMode -routeShielding <Ground Net Name>
  setCCOptMode-route_top_shielding_net <Ground Net Name>
  ```

  To guide nanoroute, we have to add following line to spec file provided to it.

  ```
  Shielding <Ground Net Name>
  ```

- ***Non-Default Rule Routing***

  Shielding need two extra nets per clock net. This consumes huge amount of routing resources as three nets have to be laid down on the same metal layer for one clock net. Due to that, design might see routing congestion. To remove need of shielding, NDRs are used. NDR are non default rules for routing. Normally net routing is done with one pitch width as net width and spacing of one pitch width. In non-default routing, clock net is made thicker i.e. twice or thrice the one pitch width and spacing is also increased to twice or thrice to

one pitch width. So the clock nets are thicker and more away from other nets. This reduces the cross talk effect.Figure 4.3 shows the NDR of 2w2s.

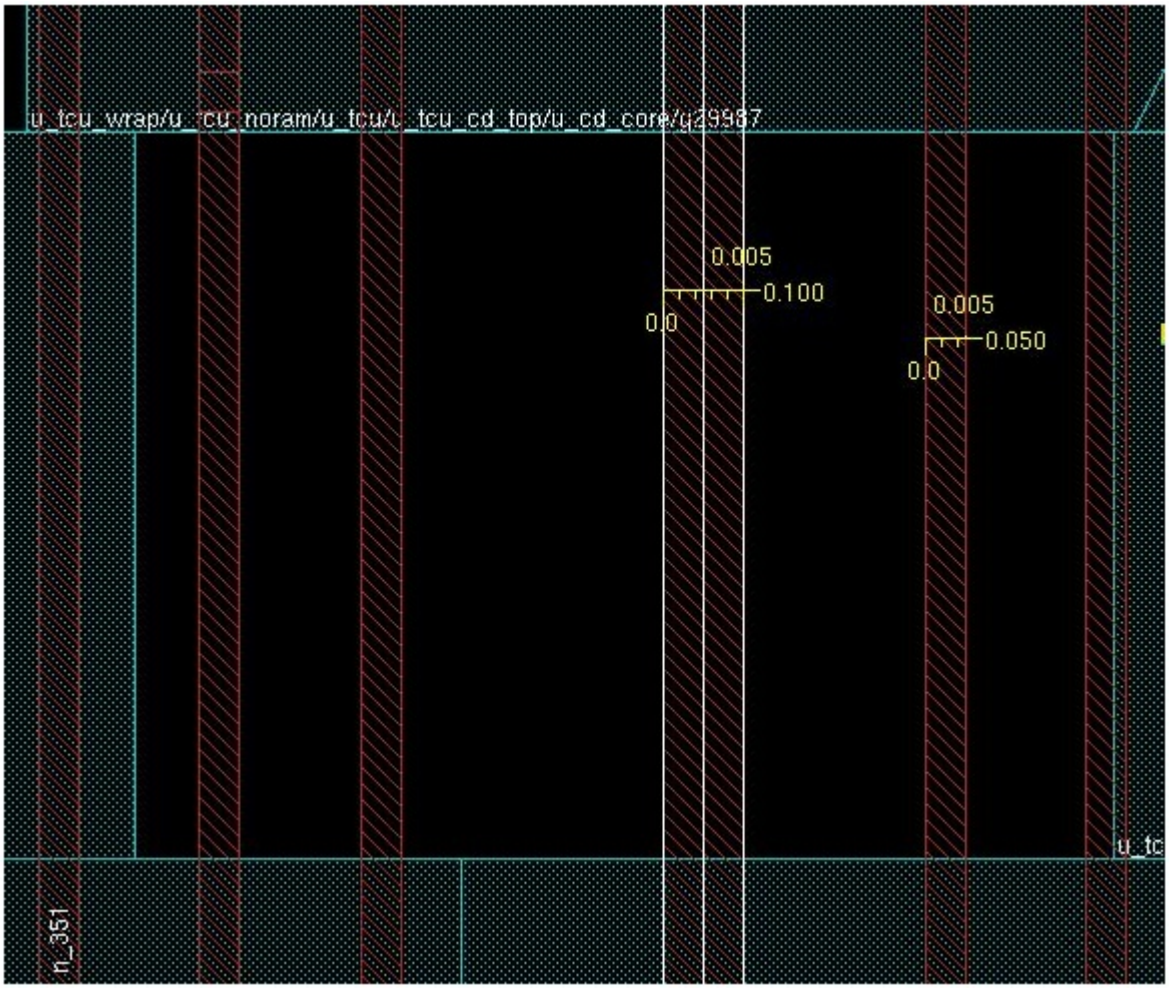


Figure 4.3: Non-Default Rule for Routing(Double width Double Spacing)

Various NDRs are used for different designs are, double width double spacing, double width triple spacing, triple width triple spacing.

To achieve Non-Default Rule routing, we have to provide .LEF of that routing rule to the tool. Also we have to provide information to the tool with following commands that it has to use which NDR to route clock nets.

```
setCTSMode -routeNonDefaultRule <Rule Name>
```

Also it has to be specified in the spec provided to nanoroute tool. It is done by following command.

```
NonDefaultRule<Rule Name>
```

Other reason of cross talk is placement of IO Pins on the chip. A large portion of the chip might be connected to the IO pins of the chip. As IO Pins are on the edges only, tool will try to put these logic near the edges. This makes the edges of the chip more dense. As density is high, number of cells in that area would be large and so, routing resources utilized in that area will be very high. This may lead to cross talk as many long nets would be placed near to each other by tool.

To make sure that cross talk doesn't come in design, IO Pins of the design should be kept as spread as possible. This will reduce the local area density near the edges and provide flexibility to the tool to route the nets from more routing resources. This will gradually solve the issue of cross talk near IO Pins.

4.3.3 Constraint Issues

Constraints are input to the physical design flow. If the constraints are given wrong, tool will be optimizing the design wrongly. Constraints are targets to the EDA tool that it has to achieve those targets and implement the design.

For example, if there are multicycle paths in the design and constraints for them are not defined, than tool will consider those paths as regular paths and will try to meet those paths in one clock cycle. As it spends more of the time in optimizing such wrong paths, run time of the design also increases and actually critical paths in design are not prioritize in optimizing.

To make sure that implementation doesn't miss out any constraint, its better to have all the constraints in single file and the same file being called in whole flow wherever constraints need to be read. If making a single constraint file is not possible than make sure that all the constraint files are read everywhere.

4.3.4 Issues Related to Platform Change

Changing the platform of foundry is not an easy task. Whole flow and optimization at each stage needs to be changed for different foundries and different technologies. As we have implemented configuration 5 on two different foundry platforms, we faces various problems.

First issue is related to the routing resources. As different foundries support different kind of technologies, the design implemented on one foundry platform may not be easily map to other. The CP foundry platforms have different technology and different routing rules like spacing, minimum width and pitch. This created a huge routing congestion in the design as routing tracks were reduced in CP foundry platforms. Later we changed the metal stack for CP foundry implementation. This also had routing congestion due to technology differences.

To solve the issue of routing congestion on CP foundry platform, the NDRs from clock route were removed. With that, the design could be implemented successfully. Thus, it might not be easy to resolve the issues raised because of changing the implementation platform as different foundries have their different standards. So Switching between foundries should be done with proper care and changes in the flow.

4.4 Summary

Chapter covers the flow optimizations that are done over the native flow of implementation. Apart from that, it also talks about some CTS experiments, results and conslusions. Issues faced in physical implementation of design are addressed and solution for them are also presented at last.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

This thesis presents details of physical implementation flow for ASIC designs over various platforms with some basic information about System Memory Management Unit whose different configurations have been implemented successfully with the same flow. We can conclude that for realistic targets, following the flow presented here would be leading to successful physical implementation of the ASIC design.

Some of the critical parts which can affect implementation results are floorplanning and Clock Tree Synthesis. Floorplanning has to be done with proper care considering nature of design and physical requirements. Optimum clock tree synthesis can be achieved by providing proper constraints and limitations to EDA tools. It is better to perform some experiments and decide parameters for clock tree synthesis on base of the results. That would lead to better implementation of design.

5.2 Future Scope

There might be some functional bug fixes coming in RTL. Those changes have to be incorporated in implementation with the help of Engineering Change Orders (ECOs)

or if that is not possible, a complete new implementation. Some more experiments on Clock Tree Synthesis should also be carried out to check if better clock tree can be synthesized or not.

Apart from that, currently implementation is being done on older version of EDA tools. This has to be switched to newer versions of tools as they are more reliable, accurate and having optimization capabilities. Moreover, EDA tool providers would not be providing support for older versions after some time.

References

[1] *www.support.cadence.com*

[2] *www.solvnet.synopsys.com*

[3] *www.en.wikipedia.org*

[4] *www.infocenter.arm.com*

[5] *www.asic − soc.blogspot.in*

[6] *www.vlsi − expert.com*

[7] *LouisScheffer, LucianoLavagno&GrantMartin*
"*ElectronicDesignAutomationforIntegratedCircuitsHandbook*"
Taylor&FrancisGroupPublications

[8] *CadenceRTLCompilerToolCommandReference*
*Version*12.2 − *June*2013

[9] *CadenceEDIToolCommandReference*
*Version*11.0*December*2011

[10] *CadenceEDISystemTimingClosureGuide*
*Version*11.1 − *April*2011

[11] *http* : *//usebackend.wordpress.com*/2012/12/18/*clock* − *tree* − *synthesis*/

[12] *http* : *//vlsi* − *soc.blogspot.in*/2013/03/*ocv* − *vs* − *pvt.html*