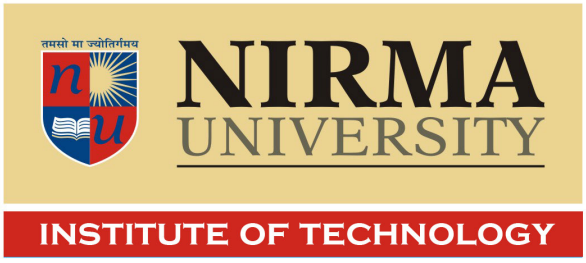
RTL to GDS-II Implementation of VLSI Design in 'nm' Technology

Major Project

Submitted in Partial Fulfilment of the Requirements
for the Degree of
Master of Technology(M.Tech.)
in
VLSI-Design
by

Nishant N. Modi
12MECV17



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

RTL to GDS-II Implementation of VLSI Design in 'nm' Technology

Major Project

Submitted in Partial Fulfilment of the Requirements
for the Degree of
Master of Technology(M.Tech.)
in
VLSI-Design
by

Nishant N. Modi
12MECV17

Dr. N. M. Devashrayee
Internal Guide

Mrs. Roma Rudra
External Guide

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

Declaration

This is to certify that

1. I, Nishant Modi, a student of Master of Technology in VLSI-Design, Nirma University, Ahmedabad hereby declare that the project work "RTL to GDS-II Implementation of VLSI Design in 'nm' Technology "has been independently carried out by me under the guidance of Mrs. Roma Rudra Principal Engineering, ARM Embedded Technologies Private Limited, Bangalore and Dr. N. M. Devashrayee, Program Co-ordinator, Department of VLSI-Design, Nirma University, Ahmedabad. This Project has been submitted in the partial 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.

Nishant Modi
12MECV17

Certificate

This is to certify that the Major Project entitled "RTL to GDS-II Implementation of VLSI Design in 'nm' Technology " submitted by Nishant Modi (12MECV17), towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI-Design of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. N. M. Devashrayee
Internal Project Guide,
PG Co-ordinator VLSI Design,
Institute of Technology,
Nirma University, Ahmedabad

Dr. D.K.Kothari,
Section Head of EC Dept
Institute of Technology,
Nirma University,
Ahmedabad

Mrs. Roma Rudra
External Project Guide
Principal Engineer, SYEG
ARM Embedded Technologies Pvt. Ltd.,
Bangalore

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

Date:

Place:Ahmedabad

Acknowledgement

This project work would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this project.

First and foremost, my utmost gratitude to **Mrs. Roma Rudra**, Principal Engineer, SYEG, ARM Embedded Technologies Pvt. Ltd., Bangalore. She has been my inspiration as I hurdle all the obstacles in the completion of this project. Her continuous guidance, support and resourcefulness in the field of physical implementation helped to conquer every big and small milestone. Without her constant support and help this project would not have been materialized. It is an honor for me to work under her guidance.

I would also like to thank **Mr. Vijay Kishan Narayanan**, my manager for giving me a live project to work on, and for his continuous help and support.

I would like to show my gratitude to **Dr. Niranjan M. Devashrayee** for being my mentor and for his continuous guidance during the full course of my postgraduate studies. I would like to thank **Dr. Nagendra P. Gajjar** for his valued attention, information and support.

I also owe my colleagues in the ARM, special thanks for helping me on this path and for making project more enjoyable.

Nishant Modi
12MECV17

Abstract

As the technology sizes of semiconductor devices continue to decrease, the effect like congestion, signal integrity, crosstalk etc. are becoming more significant on nanometer technologies. These all factors are affecting and forcing various technological methodologies throughout the design flow to constantly fight and keep updating the EDA tools to cop-up with these issues.

Physical design flow consists of producing a production-worthy layout from a gate-level netlist subject to a set of constraints. This report focuses on the steps involved in the physical design flow, problems imposed by shrinking process technologies, and the methods used to tackle these problems.The aim of this project is to successfully complete ASIC design flow from RTL to GDS-II, using the advance industry level tools. This project provides a solid base and practical hands-on experience of advanced tools like Cadence Encounter Digital Implementation System (Physical implementation for high-performance, giga-scale, low-power, and mixed-signal designs at advanced and mainstream process nodes),which gives complete solution for Logic Synthesis, Floor planning, Power Network Synthesis, Placement, Clock Tree Synthesis and Routing. Along with this, the analysis of various design factors affecting the performance of the final chip such as power, area and timing is also performed. Power analysis performed by Apache RedHawk tool & timing signoff is done by Synopsys Primetime.

Using these advanced tools the ultimate goal of the project is achieving timing closure means design free from Logical violations, Physical violations (Connectivity-opens, shorts), Design Rule violations (min spacing, min width, min via enclosure) & meeting the timing specifications (setup & hold timings).

Contents

i

ii

Declaration **iii**

Certificate **iv**

Acknowledgement **v**

Abstract **vi**

1 Introduction **1**
1.1 Physical Design: . . . 1
1.2 Motivation: . . . 2
1.3 RTL: . . . 3
1.4 GDS-II: . . . 3
1.5 Objective of Physical Implementation: . . . 3
1.6 Introduction to ASIC: . . . 4
1.7 Standard-CellBased ASIC: . . . 4
1.8 Need for Low Power ASIC: . . . 5

2 Physical Design Flow **6**
2.1 Introduction . . . 6
2.2 Data Preparation . . . 6
2.2.1 Data setup . . . 7
2.3 Synthesis . . . 8
2.4 Pre-Placement Optimization . . . 9
2.5 Floorplanning . . . 10
2.5.1 Size of the layout . . . 10
2.5.2 Partitioning . . . 10
2.5.3 Core Area . . . 11
2.5.4 Placements of IO Pads and IO Pins Geometries . . . 12
2.5.5 Placements of the Hard Macros . . . 12
2.5.6 Ensuring Routability . . . 12
2.5.7 Validating the Floorplan . . . 14
2.6 Power Planning . . . 14
2.7 Placement . . . 15
2.8 Clock Tree Synthesis . . . 17

2.8.1 Creating the Clock tree . . . 17
2.9 Post-CTS Optimization . . . 18
2.9.1 Hold Optimization . . . 18
2.10 Routing . . . 18
2.10.1 Improving Timing during Routing . . . 19
2.11 Post-Route Optimization . . . 19
2.12 Engineering Change Order (ECO) . . . 20
2.13 Timing Sign Off . . . 20

3 Static Timing Analysis Concepts 22
3.1 Introduction . . . 22
3.2 Other types of path: . . . 23
3.3 Setup Check: . . . 25
3.4 Hold Check: . . . 25
3.5 Ways to fix Setup Violation: . . . 26
3.6 Ways to fix Hold Violation: . . . 29

4 Design Problems & Solutions 30
4.1 Need of Placement Blockage . . . 30
4.1.1 Introduction . . . 30
4.1.2 Problem . . . 31
4.1.3 Solution . . . 31
4.2 Need of Routing Blockage . . . 31
4.2.1 Introduction . . . 31
4.2.2 Problem . . . 31
4.2.3 Solution . . . 32
4.3 Congestion . . . 32
4.3.1 Introduction . . . 32
4.3.2 Problem . . . 33
4.3.3 Solution . . . 33
4.4 Effect of Crosstalk and Need of Shielding . . . 34
4.4.1 Crosstalk Delay Effects . . . 34
4.4.2 Crosstalk Noise Effects . . . 35
4.4.3 Shielding . . . 35
4.4.4 Problem . . . 35
4.4.5 Solution . . . 36
4.5 Create Region/Bound . . . 36
4.5.1 Problem . . . 36
4.5.2 Solution . . . 36
4.6 Using standard cell library with increased channel length . . . 37
4.7 Need of Timing Exceptions . . . 38
4.7.1 False path . . . 38
4.7.2 Multi-cycle path . . . 39
4.8 Timing Improvement Tricks at Synthesis . . . 40
4.8.1 Path adjust . . . 40
4.8.2 Make PathGroup . . . 40
4.8.3 Ungrouping . . . 40
4.8.4 Low Vth standard cells . . . 40

5 Conclusion & Future scope **41**
5.1 Conclusion 41
5.2 Future scope 42

6 References **43**

List of Figures

1.1 A layout of a chip . . . 1
1.2 IC Design flow . . . 2
1.3 Cell based ASIC (CBIC) . . . 4
1.4 Dynamic and Leakage Power Comparison . . . 5

2.1 Block diagram of Timing Closure Flow . . . 6
2.2 The Basic Synthesis flow . . . 9
2.3 Partitioning . . . 11
2.4 A floor plan with one macro and I/O pads . . . 12
2.5 Steps in building a power plan . . . 15
2.6 Global routing . . . 19
2.7 Detail routing . . . 19

3.1 Types of pathgroups . . . 23
3.2 Types of pathgroups . . . 23
3.3 Faulse Path . . . 24
3.4 Setup and Hold Slack Timings . . . 25

4.1 Placement Halo . . . 30
4.2 Placement Halo . . . 31
4.3 Routing Halo . . . 32
4.4 Congestion . . . 33
4.5 Transition Slowdown or Speedup Caused by Crosstalk . . . 34
4.6 Delta Delay Includes the Fanout Stage Effect . . . 34
4.7 Noise bump due to crosstalk . . . 35
4.8 Clk pin and first buffer placement . . . 35
4.9 Shielding of Clk net . . . 35
4.10 no presence of bound . . . 36
4.11 placement after creating bound . . . 37
4.12 Comparison between different technology . . . 37
4.13 False path . . . 38
4.14 Multi-cycle path . . . 39

Chapter 1

Introduction

1.1 Physical Design:

The design-cycle of VLSI-chips consists of different consecutive steps from high-level synthesis (functional design) to production (packaging). The physical design is the process of transforming a circuit description into the physical layout, which describes the position of cells and routes for the interconnections between them. Once the circuit netlist has been captured and validated, each leaf of hierarchy has to be placed and routed. Figure 1.1 shows a schematic representation of a layout. The main concern in the physical design of VLSI-chips is to find a layout with minimal area, further the total wirelength has to be minimized. For some critical nets there are hard limitations for the maximal wirelength.

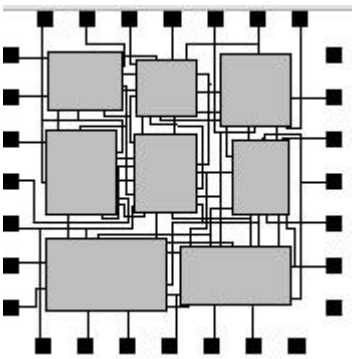


Figure 1.1: A layout of a chip

Due to its complexity, the physical design is normally broken in various sub-steps:

1. First the circuit has to be partitioned to generate some macro cells.
2. In the floorplanning phase the cells have to be placed on the layout surface.
3. After placement the global routing has to be done. In this step the 'loose' routes for the interconnections between the single modules (macro cells) are determined.
4. In the detailed routing the exact routes for the interconnection wires in the channels between the macro cells have to be computed.

This classical approach of the physical design is strongly serial with many interdependencies between the sub-steps. For example during floorplanning and global routing there

must be enough routing space reserved to complete the exact wiring in the detailed routing phase. Otherwise the placement has to be corrected and the global routing has to be computed again.

1.2 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.

RTL to GDSII flow includes many tasks like synthesis, floorplanning, routing. It is task or flow to convert 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.

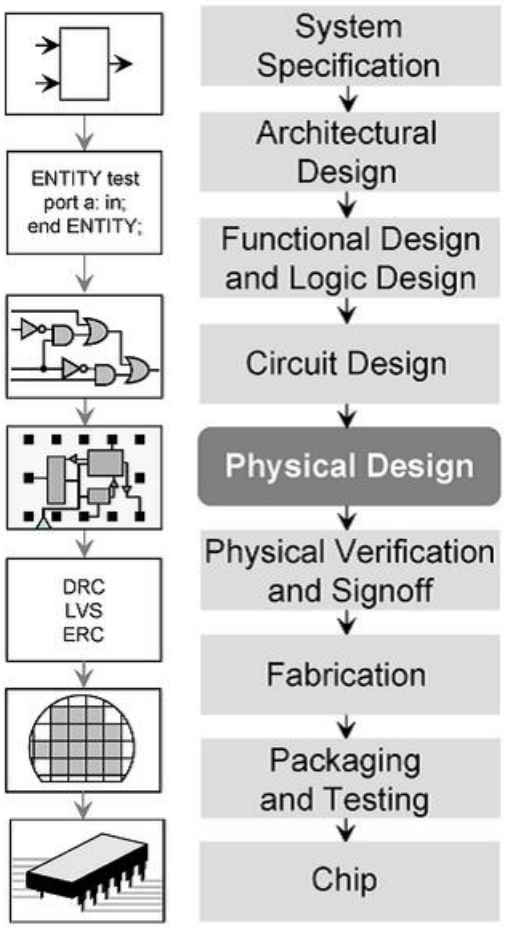


Figure 1.2: IC Design flow

1.3 RTL:

RTL is acronym of Register Transfer Level. 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 representation, lower level representations can be obtained. Verilog and VHDL are such well known HDLs.

RTL doesnt have gate level information. Logic synthesizer in EDA tool synthesis the logic in RTL and converts it to appropriate gate.

1.4 GDS-II:

GDS-II is a database format. It is a industry standard for data exchange especially for IC layout. Basically it is 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.

1.5 Objective of Physical Implementation:

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 PDCI- Bangalore team of ARM with the cadence Electronic Design Automation (EDA) tools,RTL compiler and Encounter Digital Implementation tools. Achieving timing closure is the main goal of physical implementation of VLSI Design, means:

- Design free from:
  - Logical violations
  - Physical violations (Connectivity-opens, shorts)
  - Design Rule violations (DRV-min spacing, min width, min via enclosure)
- Meeting the timing specifications (setup & hold timings).
- minimize area and power(static & dynamic).

1.6 Introduction to ASIC:

An application-specific integrated circuit (ASIC) is an integrated circuit(IC) customized for a particular use, rather than intended for general-purpose use. In today's world, ASICs.

Offer many advantages over off-the-shelf devices:

1. Smaller die size leads to board size reduction
2. Reduced power consumption, less heat dissipation
3. Lower costs under mass production
4. Improved performance
5. Better radiation tolerance
6. Improved testability
7. Enhanced reliability
8. Proprietary design implementation

1.7 Standard-CellBased ASIC:

A cell-based ASIC uses predefined logic cells like AND gates, OR gates, multiplexers, and flip-flops known as standard cells. The flexible blocks in a CBIC are built of rows of standard cells. Placement of the standard cells and the interconnect is defined by an ASIC designer in a CBIC. The advantage of CBICs is that they can be designed in less time with small amount of money compared to full-custom ASICs, and also the most important thing is it reduce the risk by using a predesigned, pretested, and pre-characterized standard-cell library which can be optimized individually. At the same time, the disadvantages are the time or expense of designing or buying the standard-cell library and the time needed to fabricate all layers of the ASIC for each new design. Figure shows a CBIC.

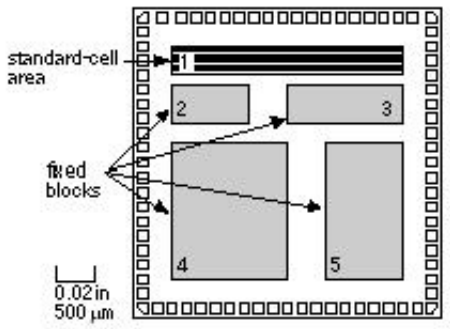


Figure 1.3: Cell based ASIC (CBIC)

Each standard cell in the library is constructed using full-custom design methods, but you can use these predesigned and pre-characterized circuits without having to do any

full-custom design yourself. This design style gives you the same performance and a flexibility advantage of a full-custom ASIC but reduces design time and reduces risk.[1]

1.8 Need for Low Power ASIC:

For early digital circuits, high speed and minimum area were the main design constraints. Most of the EDA tools were designed specifically to meet these criteria. Power consumption was never highly visible. Nowadays, the area reduction of digital circuits is no longer a big issue as with the latest sub-micron techniques, many millions of transistors can be fit in a single IC. Smaller chip size eventually leads to high demand for portable and handheld devices. More and more applications are battery powered, and low power ICs are the key to extend the usage time in between battery recharge, and in turn increase battery life and reliability of the product. Also in submicron technologies, there is a limitation on the proper functioning of circuits due to heat generated by power dissipation. Market forces are demanding low power for not only longer battery life but also reliability, portability, performance, cost and time to market. This is very true in the field of personal computing devices, wireless communications systems, home entertainment systems, which are becoming popular now-a-days. Implantable medical devices, such as pace maker, deep brain system for Parkinsons disease, and spinal cord stimulator for pain management, particularly need to dissipate less power for longer battery life and improved component reliability and safety.

As process technology reduces into 90nm and below, performance and density are taken to new levels, yet power loss in both switching and leakage makes designing with these devices a major challenge. Leakage power reduction is essential in sustaining the scaling of the CMOS process. Leakage power is now becoming proportional to dynamic orswitching power loss as shown in Figure below. While lowering of the threshold voltage leads to significant increase in sub-threshold leakage current, the increase in gate tunneling leakage current is caused by thinner gate oxides. While scaling improves transistor density, functionality, and higher performance on chip, it also results in power dissipation increase. Therefore, it has become necessary to use new techniques to manage energy at the system level.

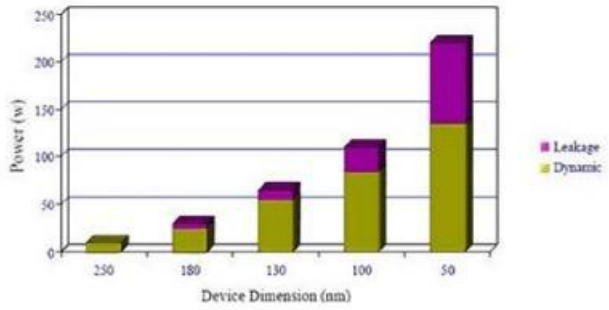


Figure 1.4: Dynamic and Leakage Power Comparison

Bottom line, low power budget has become one of the most important design parameters for VLSI (Very Large Scale Integration) systems.[2]

Chapter 2

Physical Design Flow

2.1 Introduction

Achieving timing closure on a design is the process of creating a design implementation that is free from logical, physical, and design rule violations and meets or exceeds the timing specifications for the design. For a production chip, all physical effects, such as metal fill and coupling, must be taken into account before you can confirm that timing closure has been achieved.

Timing Closure is not just about timing optimization. It is a complete flow that has to converge, including placement, timing optimization, clock tree synthesis (CTS), routing and SI fixing. Each step has to reach the expected targets or else timing closure will likely not be achieved.

Below is a high level diagram showing the steps in the timing closure flow

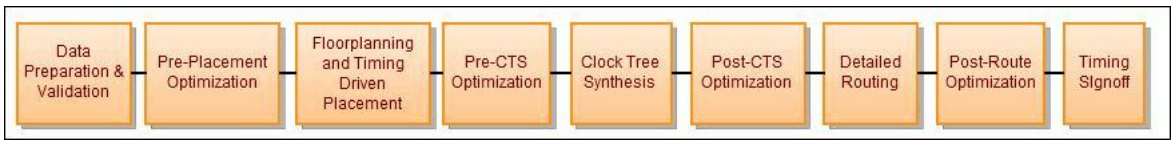


Figure 2.1: Block diagram of Timing Closure Flow

2.2 Data Preparation

This section outlines the data (libraries, constraints, netlist, etc.) required for implementing the timing closure flow.

The goals of data preparation include:

- Confirming that the tool has a complete and consistent set of design data (all library views and versions must be consistent).
- Ensuring that all tools in the flow interpret the timing constraints
- Making sure logically equivalent cells are defined properly.

- Creating a capacitance table file that matches the process technology.
- Correlating parasitics among the prototyping and sign-off extraction tools.

2.2.1 Data setup

This section lists the data required and data setup recommended for the timing closure flow.[3]

Standard Cell Libraries, I/O Cell Libraries, 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, and IBM etc. Standard cell height for 130 TSMC process is 3.65 m. 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.

Physical Libraries/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.

Verilog Netlist

The netlist should be unique.Can be of flattened or hierarchical.Contains connectivity info of all cells

Constraints

Timing constraints in the form of SDCs are required. Contain all design related constraints like Area, power, timing

Capacitance Table File

A capacitance table is used to accurately extract parasitics. A cap table is required for each RC corner to be analyzed. You may obtain the cap table(s) from your foundry. The Interconnect Technology (ICT) file describes the detailed process information for a given technology (layer thicknesses, materials, profiles, dielectric constants, and so forth). Any non-default rules for the design should be defined in the LEF file you are using.

The capacitance table file consists of two parts:

- Basic Capacitance Table Provides a table of spacing versus capacitance information for each layer
- Extended Capacitance Table Provides encoded extraction patterns that are derived using a 3D field solver which provides much greater accuracy during detailed extraction. A capacitance table file (basic or extended) is process-specific and not design dependent, so it only needs to be created once per technology. If a capacitance table already exists for your process, use it and do not recreate it.

QRC Tech File

A QRC technology file (ICT file) is required in order to run turbo-QRC (tQRC), integrated QRC (iQRC) or standalone QRC. A QRC tech file is required for each RC corner to be analyzed.

Signal Integrity (SI) Libraries

SI libraries in the form of cdB's are required for SI analysis and optimization. These can be generated from SPICE using the make_cdb command.

Multi-Mode Multi-Corner (MMMC) Setup

Multi-Mode Multi-Corner (MMMC) setup is required for optimizing designs over multiple operating conditions. MMMC setup is also required for accurate push-out/pull-in reporting during Signal Integrity (SI) analysis.

2.3 Synthesis

Synthesis is the process that generates a gate-level netlist for a design that has been defined using a Hardware Description Language (HDL). Synthesis includes reading the HDL source code and optimizing the design from that description. Using the technology library's cell logical view, the Logic Synthesis tool performs the process of mathematically transforming the ASIC's register-transfer level (RTL) description into a technology-dependent netlist. This process is similar to a software compiler converting a high-level

C-program listing into a processor-dependent assembly-language listing. The netlist is the standard-cell representation of the ASIC design, at the logical view level. It consists of instances of the standard-cell library gates, and port connectivity between gates. Proper synthesis techniques ensure mathematical equivalence between the synthesized netlist and original RTL description. The netlist contains no unmapped RTL statements and declarations.

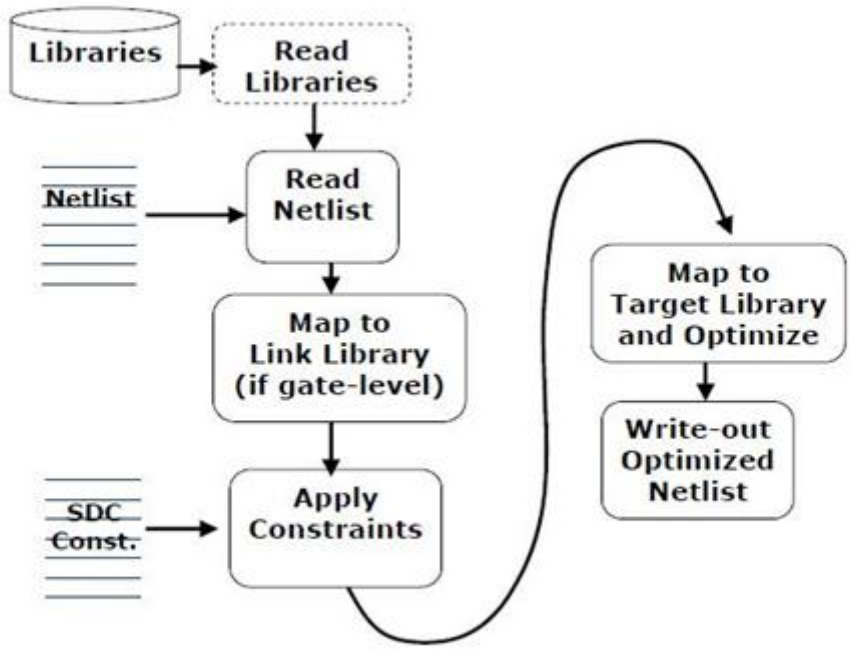


Figure 2.2: The Basic Synthesis flow

2.4 Pre-Placement Optimization

The goals of pre-placement optimization is to optimize the netlist to:

- Improve the logic structure
- Reduce congestion
- Reduce area
- Improve timing

In some situations, the input netlist (typically from a poor RTL synthesis) is not a good candidate for placement since it might contain buffer trees or logic that is poorly structured for timing closure. In most cases, high-fanout nets should be buffered after placement. It is more reliable to allow buffer insertion algorithms to build and place buffer trees rather than to rely on the placer to put previously inserted trees in optimal locations. Additionally, having buffer trees in the initial netlist can adversely affect the initial placement.

Because of these effects, it can be advantageous to run pre- placement optimization or simple buffer and double-inverter removal (area reclamation) prior to initial placement.For designs where the logical structure of the critical paths or high congestion are the sources of closure problems, restructuring or remapping the elements on the paths (or related portions of the design) can provide better timing results. This is done by running netlist-to-netlist optimization on the initial netlist prior to place and route. Additional restructuring can be performed later in the flow.

2.5 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.[4]

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 wastage of die area and routing congestion.

2.5.1 Size of the 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 co-ordinates 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.

2.5.2 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.

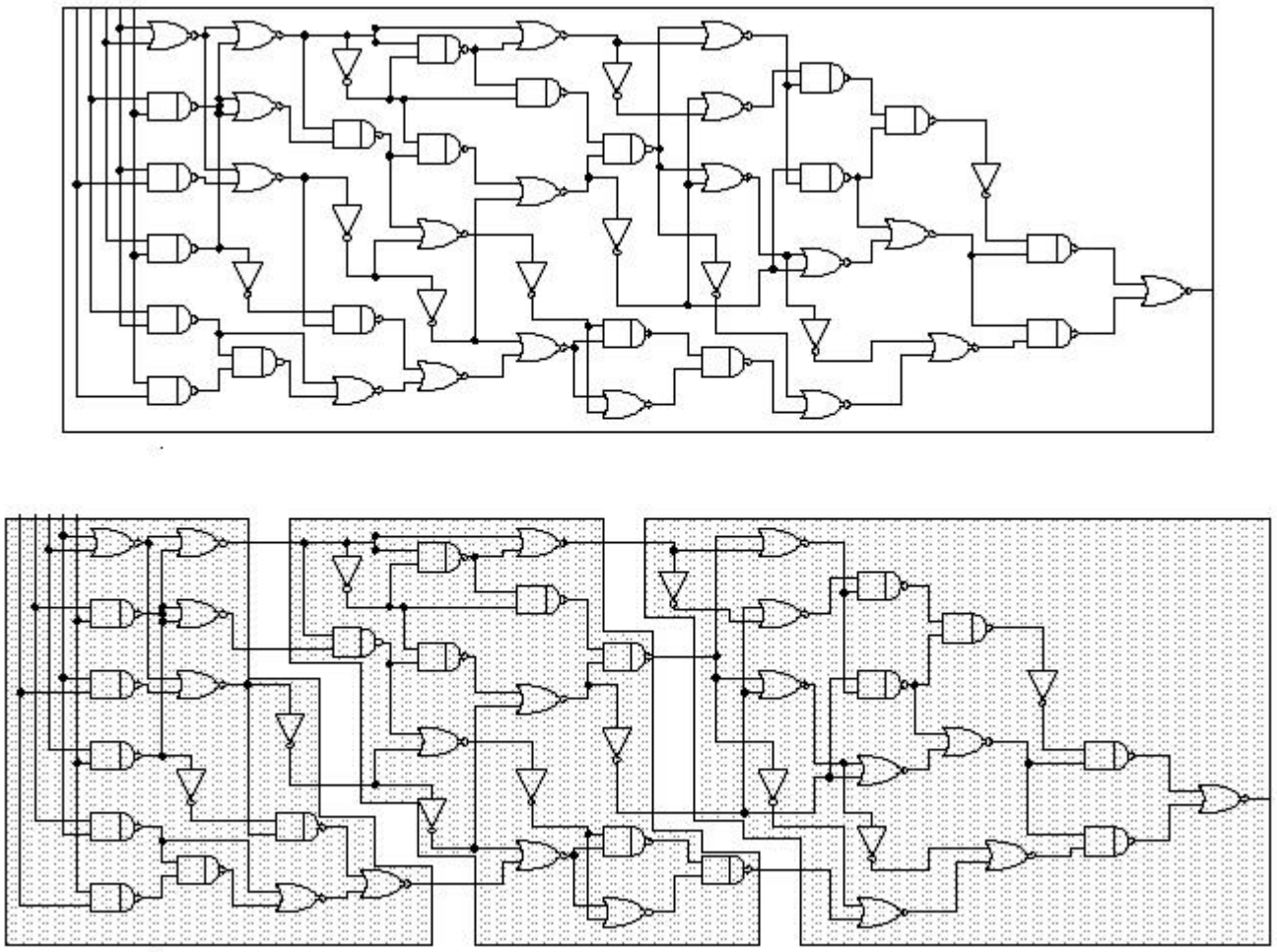


Figure 2.3: Partitioning

2.5.3 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.[5]

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.

2.5.4 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.

2.5.5 Placements of the Hard Macros

The floor plan 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 floor plan, which cannot be achieved without executing the rest of the flow. Thus, floor planning is an iterative and time consuming process. The trick in performing floor planning is to shorten the turn-around time of the iterations, and to reduce the number of iterations. The following figure depicts a simple floor plan of a chip-level layout with only one macro.

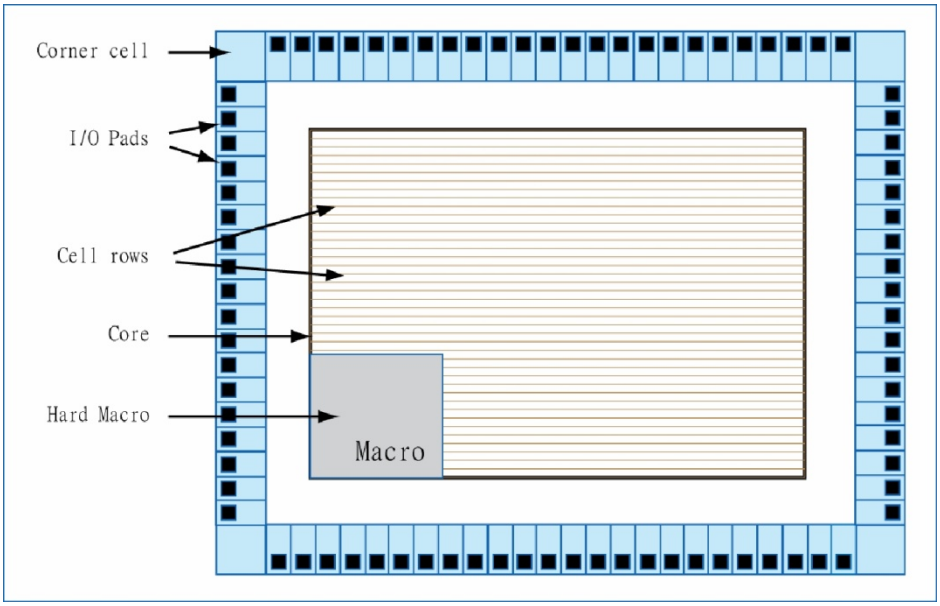


Figure 2.4: A floor plan with one macro and I/O pads

2.5.6 Ensuring Routability

Initial prototype iterations should focus on routability as the key to achieving predictable timing closure. You should attempt to resolve congestion before attempting timing closure. Designs which are congested are more likely to have timing jumps during timing and Signal Integrity (SI) closure. Tools such as module guides, block placement, block

halos, obstructions, and partial placement blockages (density screens) are used to control the efficient routing of the design.

Use the following guidelines during floorplanning and placement to avoid congestion.

1. Choose an appropriate floorplan style.

    - If possible, review a data flow diagram or a high-level design description from the chip designer to determine an appropriate floorplan style.
    - Assess different floorplan styles such as hard macro placement in periphery, island, or doughnut (periphery and island). Keep the macro depth at 1 to 2 for best CTS, optimization, and Design-for-test (DFT) results. If possible,consider different aspect ratios to accommodate a shallower macro depth. Consider using Automatic Floorplan Synthesis and relative floorplanning constraints to simplify floorplan iterations.

2. Preplace I/Os and macros

    - Review hard macro connectivity and placement based on the minimum distance from a hard macro to its target connectivity.
    - Preplace high-speed and analog cores based on their special requirements for noise isolation and power domains.

3. Review I/O placement to identify I/O anchors and associated logic

    - Verify that logic blocks and hard macros which communicate with I/O buffers are properly placed and have optimal orientation for routability. Push down into module guides to further assess the quality of the floorplan and resulting placement.

4. Allow enough space between preplaced blocks.

    - Allow space between I/Os and peripheral macros for critical logic such as JTAG, PCI, or Power management logic.
    - Use block halos, placement obstructions or fences around blocks prior to optimization or Clock Tree Synthesis (CTS). Placement generally does not do a good job of placing cells between macros. Reduce or remove halos and obstructions after placement to make sufficient space available around macros for optimization, CTS, DRV, or SI fixing to add buffers.
    - Place other cells such as endcaps, welltaps and decaps prior to placement as required.

5. Use module guides carefully

    - Place module guides, regions, or fences only when greater control is required. Be careful not to place too many module constraints early in the floorplanning process because it is time consuming and greatly constrains the placement. Module guides should be used for floorplan refinement or hierarchical partitioning.

- Review the placement of module guides related to datapath and control logic relative to the associated hard macros. Datapath logic can be a source of congestion problems due to poor aspect ratios, high fanout, and large amounts of shifting. Consider tuning the locations of these module guides and lowering the density to reduce congestion.
- Review the placement of module guides related to memories. These modules can typically have higher densities due to the inclusion of the memories.

6. Reorder scan chains

    - When evaluating congestion, make sure that scan chains are reordered to eliminate "false" hot spots in the design. Failing to reorder the scan chain can cause a routable floorplan to appear unroutable. If scan chain information is missing, no reordering is performed.

2.5.7 Validating the Floorplan

A congested or unroutable design at this stage will not get better during optimization.Make sure to consider the following when finalizing the floorplan:

- The power grid should be defined:
    - Global net connections properly defined.
    - Nets requiring optimization should be defined as signal (regular) nets.
    - PINS marked with + SPECIAL cannot be optimized.
    - Followpin routing should align to rows/cells with correct orientation (VDD pin to VDD followpin).
- Gaps between standard cells and blocks should be covered with soft or hard blockages. Placement cannot place cells in the area of soft blockages but optimization and CTS can. All blocks should be marked fixed.
- Tracks should match IO pins and placement grid (rows).

2.6 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.

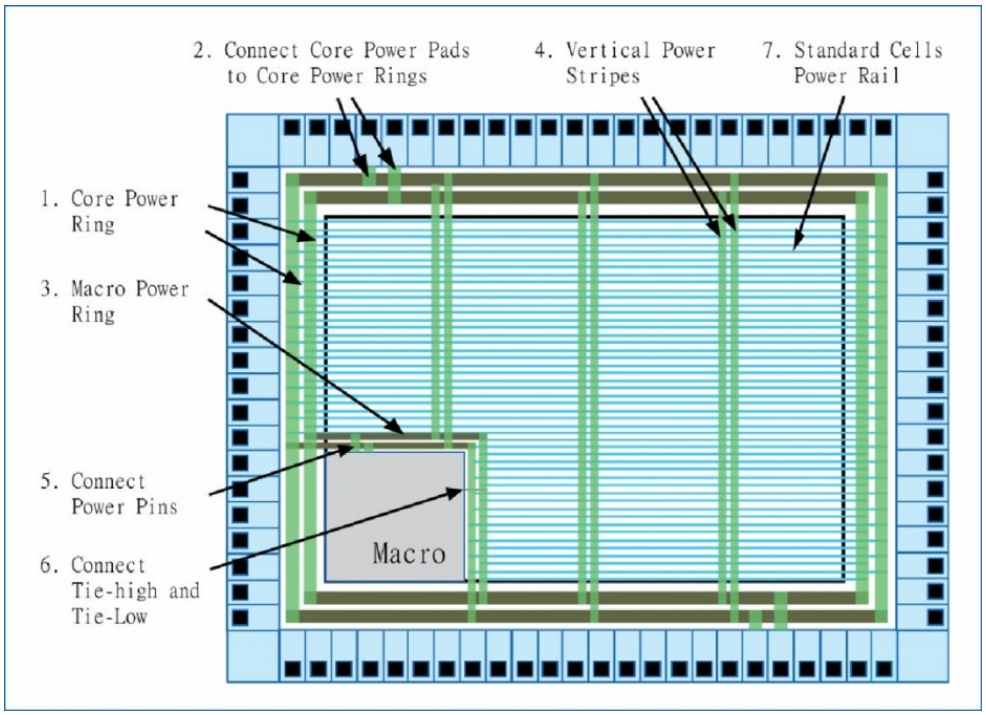


Figure 2.5: Steps in building a power plan

2.7 Placement

During placement, the blocks are exactly positioned on the chip. The goal of placement is to find a minimum area arrangement for the blocks that allows completion of interconnections between the blocks.[6]

Placement is typically done in two phases. In the first phase, an initial placement is created. In the second phase, the initial placement is evaluated and iterative improvements are made until the layout has minimum area and conforms to design specifications. Between adjacent cells, interconnection is as simple as ensuring that the signal in each cell goes all the way to the edge of the cell at the same location on the common edge. In this way, the mere placement of the cells adjacent to each other in the next level of layout hierarchy connects the two cells properly. This eliminates the need to manually connect adjacent cells on the top level and helps to guide the layout of the leaf cells.

However, it also requires that the leaf cell layouts meet design rule requirements when placed adjacent to each other. The quality of the placement will not be evident until the routing phase has been completed.

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 RCs are more accurate than WLM RCs.

Placement is performed in four optimization phases:

1. **Pre-placement optimization**
   Pre-placement Optimization optimizes the netlist before placement, HFNs are collapsed. It can also downsize the cells.

2. **In placement optimization**
   In-placement optimization re-optimizes the logic based on VR. This can perform cell sizing, cell moving, cell bypassing, net splitting, gate duplication, buffer insertion, area recovery. Optimization performs iteration of setup fixing, incremental timing and congestion driven placement.

3. **Post Placement Optimization (PPO) before clock tree synthesis (CTS)**
   Post placement optimization before CTS performs netlist optimization with ideal clocks. It can fix setup, hold, max trans/cap violations. It can do placement optimization based on global routing. It re does HFN synthesis.

4. **PPO after CTS**
   Post placement optimization after CTS optimizes timing with propagated clock. It tries to preserve clock skew.

2.8 Clock Tree Synthesis

The goal of clock tree synthesis (CTS) is to minimize skew and insertion delay. Clock is not propagated before CTS . After CTS hold slack should improve. Clock tree begins at .sdc defined clock source and ends at stop pins of flop. There are two types of stop pins known as ignore pins and sync pins. Dont touch circuits and pins in front end (logic synthesis) are treated as ignore circuits or pins at back end (physical synthesis). Ignore pins are ignored for timing analysis. If clock is divided then separate skew analysis is necessary:

- Global skew achieves zero skew between two synchronous pins without considering logic relationship.
- Local skew achieves zero skew between two synchronous pins while considering logic relationship.
- If clock is skewed intentionally to improve setup slack then it is known as useful skew.

In clock tree optimization (CTO) clock can be shielded so that noise is not coupled to other signals. But shielding increases area by 12 to 15%. Since the clock signal is global in nature the same metal layer used for power routing is used for clock also. CTO is achieved by buffer sizing, gate sizing, buffer relocation, level adjustment and HFN synthesis. We try to improve setup slack in pre-placement, in placement and post placement optimization before CTS stages while neglecting hold slack. In post placement optimization after CTS hold slack is improved. As a result of CTS lot of buffers are added. Generally for 100k gates around 650 buffers are added.[7]

2.8.1 Creating the Clock tree

Clock tree synthesis (CTS) is a series of procedures to build a buffer distribution network to meet the design's timing targets. The clock tree specification file is used to direct clock tree synthesis and includes:

- Design constraints including latency, skew
- Design rules, Buffer and routing type definitions
- Trace and synthesis controls like: MacroModel, ClkGroup, NoGating, LeafPin, ExcludedPin, PreservePin, ThroughPin, and GatingGroupInstances
- You can control the buffer types to build clock trees by listing the buffer types in theconfig file. You can also control the routing types for the clock nets.
- Deletes any existing buffers on the clock nets

2.9 Post-CTS Optimization

The goals of post-CTS optimization include:

- Fixing remaining design rule violations
- Optimizing remaining setup violations
- Correcting timing with propagated clocks
- Hold violation fixing

At this point in the design flow, the clocks are inserted and preferably routed.

2.9.1 Hold Optimization

Timing optimization to fix hold violations can be performed at this point. This is recommended if your design has a significant number of hold violations because they are easier to fix prior to routing. If the number of hold violation is low you can wait until after routing the signal nets.

2.10 Routing

Interconnect design and optimizations have received much attention recently in deep sub-micron VLSI layout design. The objective of the routing phase is to complete the interconnections between blocks according to the specified netlist. First, the space not occupied by the blocks (called the routing space) is partitioned into rectangular regions called channels and switchboxes.

The goal of a router is to complete all circuit connections using the shortest possible wire length and using only the channel and switch boxes. This is usually done in two phases, referred to as the global and detailed routing phases.

Global routing is often planned with the aid of these channels and tracks. Channels are areas of a die between functional units used exclusively for routing. They are often highly oblong to accommodate long buses. Because most metal layers in a channel are used for routing, it is usually not possible to put devices into the channels. Routing tracks are used to organize and simplify routing within and over layout cells. Generally, global routing layers (the higher metal layers) are assigned a preferred direction that alternates every layer. Usually this direction restriction does not apply to the lowest one or two metal layers because of their use in convoluted local routes. Picture the entire layout overlaid with minimum width metal lines running the length of the bit slice. The lines are at the minimum pitch which allows space for via landing pads to upper and lower metal layers. Each of these hypothetical metal lines is referred to as a track.[8]

Global routing is followed by detailed routing, which completes point-to-point connections between pins on the blocks. Loose routing is converted into exact routing by specifying geometric information such as width of wires and their layer assignments.

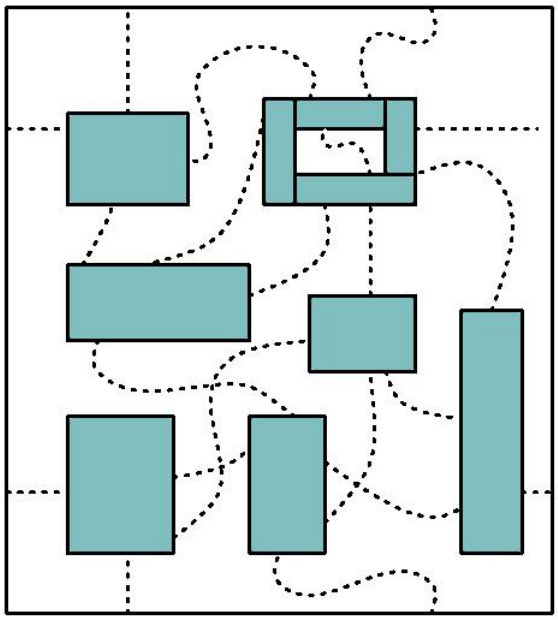


Figure 2.6: Global routing

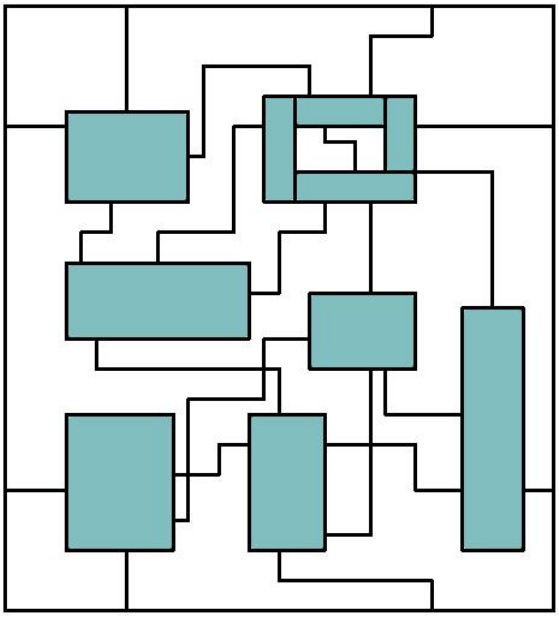


Figure 2.7: Detail routing

Compaction is simply the task of compressing the layout in all directions such that the total area is reduced. By making the chip smaller, wire lengths are reduced which in turn reduces the signal delay between components of the circuit. At the same time, this reduced area implies that more chips can be produced on a single wafer driving down the cost of manufacturing. Compaction must ensure that no rules regarding the design and fabrication process are violated during the process.

2.10.1 Improving Timing during Routing

The following tips can help achieve better timing results during the routing phase of the design.

- Make sure the LEF file contains a sufficient number and variety of vias
- Check the definition of tracks in the DEF file.
- If timing is way off and/or there is local or global congestion, return to post-CTS optimization and optimize further or run non-timing-driven routing.
- Unfix the clock nets before routing.
- Make sure the top max routing layer is set appropriately.
- Specify required NonDefaultRules (NDRs) and/or shield routing.

2.11 Post-Route Optimization

During post-route optimization, there should be very few violations that need correction. The primary sources of these timing violations include:

- Inaccurate prediction of the routing topology during pre-route optimization due to congestion-based detour routing
- Incremental delays due to parasitics coupling

Since the violations at this stage are due to inaccurate modeling of the final route topology and the attendant parasitics, it is critical at this point not to introduce any additional topology changes beyond those needed to fix the existing violations. Making unnecessary changes to the routing at this point can lead to a scenario where fixing one violation leads to the creation of others. This cascading effect creates a situation where it becomes impossible to close on a final timing solution with no design rule violations.

One of the strengths of post-route optimization is the ability to simultaneously cut a wire and insert buffers, create the new RC graph at the corresponding point, and modify the graph to estimate the new parasitics for the cut wire without re-doing extraction.

In addition to the timing violations caused by inaccurate route topology modeling, the parasitics cross-coupling of neighboring nets can cause the following problems that need to be addressed in high speed designs:

- An increase or decrease in incremental delay on a net due to the coupling of its neighbors and their switching activity.
- Glitches (voltage spikes) that can be caused in one signal route by the switching of a neighbor resulting in a logic malfunction.

These effects need to be analyzed and corrected before a design is completed. They are magnified in designs with small geometries and in designs with high clock speeds.

2.12 Engineering Change Order (ECO)

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.

2.13 Timing Sign Off

The goal of timing sign off is to verify that the design meets the specified timing constraints. This is accomplished by first using QRC to generate detailed extraction data and then using the specified timing analysis engine for a final analysis of setup and hold data.

At this point in the design process, final routing and post-route optimization is complete.

- Runs QRC to generate detailed parasitics.
- Uses the detailed parasitics and generates the setup timing reports.
- Generates the hold timing reports.

Chapter 3

Static Timing Analysis Concepts

3.1 Introduction

Static timing analysis is a method of validating the timing performance of a design by checking all possible paths for timing violations under worst-case conditions. Static timing analysis checks the design only for proper timing, not for correct logical functionality. In static timing analysis, the word static alludes to the fact that this timing analysis is carried out in an input-independent manner. It has proven efficient only for fully synchronous designs. Since the majority of chip design is synchronous, it has become a mainstay of chip design over the last few decades.

The Way STA is performed on a given Circuit:

To check a design for violations or say to perform STA there are 3 main steps:

- Design is broken down into sets of timing paths,
- Calculates the signal propagation delay along each path
- And checks for violations of timing constraints inside the design and at the input/output interface.

The STA tool analyzes ALL paths from each and every startpoint to each and every endpoint and compares it against the constraint that (should) exist for that path. All paths should be constrained, most paths are constrained by the definition of the period of the clock, and the timing characteristics of the primary inputs and outputs of the circuit.

Data Paths:

- Input pin/port to Register(flip-flop).
- Input pin/port to Output pin/port.
- Register (flip-flop) to Register (flip-flop)
- Register (flip-flop) to Output pin/port

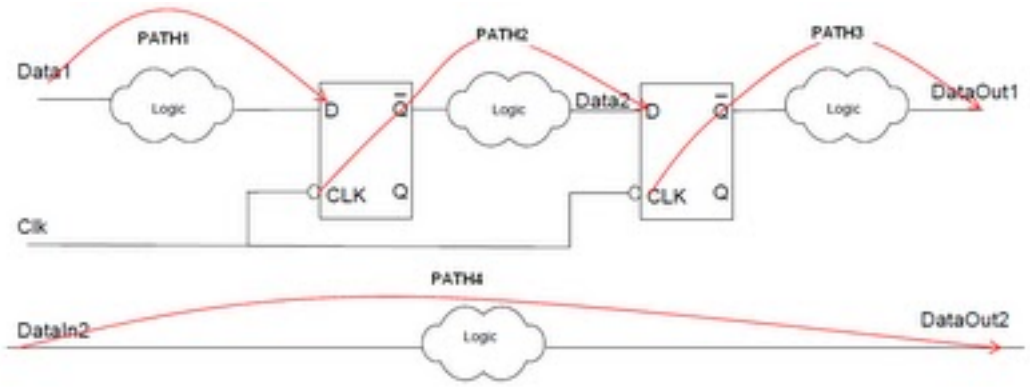


Figure 3.1: Types of pathgroups

PATH1- starts at an input port and ends at the data input of a sequential element. (Input port to Register)
PATH2- starts at the clock pin of a sequential element and ends at the data input of a sequential element. (Register to Register)
PATH3- starts at the clock pin of a sequential element and ends at an output port.(Register to Output port).
PATH4- starts at an input port and ends at an output port. (Input port to Output port)

Clock Path:

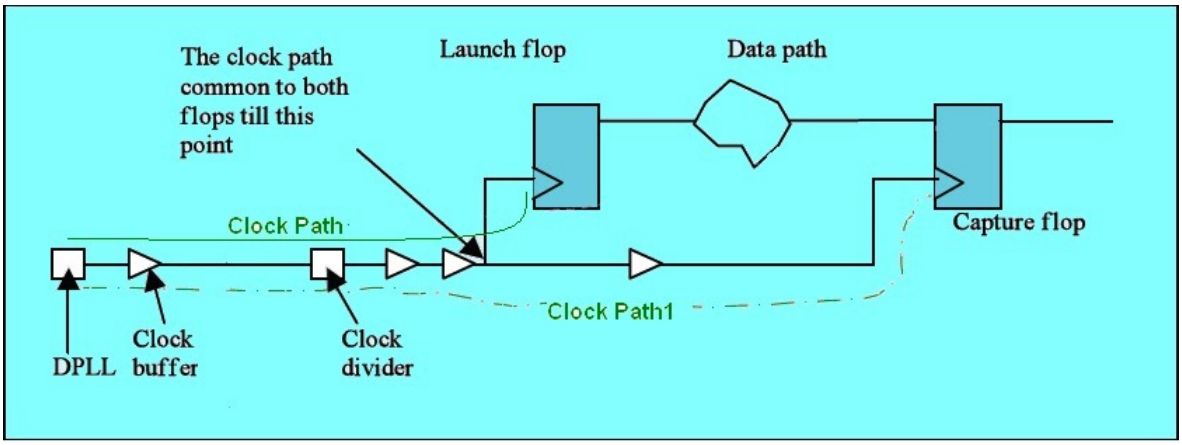


Figure 3.2: Types of pathgroups

In the above fig it is very clear that for clock path the starts from the input port/pin of the design which is specific for the Clock input and the end point is the clock pin of a sequential element. In between the Start point and the end point there may be lots of Buffers/Inverters/clock divider.

3.2 Other types of path:

There are few more types of path which we usually use during timing analysis reports. Those are subset of above mention paths with some specific characteristics. Since we are

discussing about the timing paths, so it will be good if we will discuss those here also. Few names are

- **Critical path**

  Timing critical path are those path that do not meet your timing. What normally happens is that after synthesis the tool will give you a number of paths which have a negative slag. The first thing you would do is to make sure those path are not false or multicycle since it that case you can just ignore them.

- **False Path**
  - Physically exist in the design but those are logically/functionally incorrect path. Means no data is transferred from Start Point to End Point. There may be several reasons of such path present in the design.
  - Some time we have to explicitly define/create few false path with in the design. E.g for setting a relationship between 2 Asynchronous Clocks.
  - The goal in static timing analysis is to do timing analysis on all ?true? timing paths, these paths are excluded from timing analysis.
  - Since false path are not exercised during normal circuit operation, they typically don't meet timing specification,considering false path during timing closure can result into timing violations and the procedure to fix would introduce unnecessary complexities in the design.
  - There may be few paths in your design which are not critical for timing or masking other paths which are important for timing optimization, or never occur with in normal situation. In such case , to increase the run time and improving the timing result , sometime we have to declare such path as a False path , so that Timing analysis tool ignore these paths and so the proper analysis with respect to other paths. Or During optimization don't concentrate over such paths. One example of this. e.g A path between two multiplexed blocks that are never enabled at the same time. You can see the following picture for this:

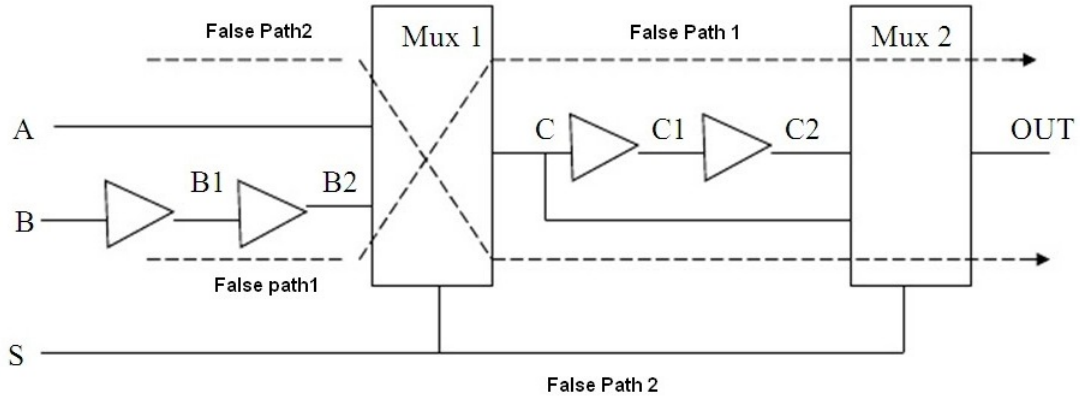


Figure 3.3: Faulse Path

Here you can see that false path 1 and False Path 2 can?t occur at the same time but during optimization it can affect the timing of another path. So in such scenario, we have to define one of the paths as false path.

- **Multi-cycle path:**
  A multicycle path is a timing path that is designed to take more than one clock cycle for the data to propagate from the startpoint to the endpoint.
  A multi-cycle path is a path that is allowed multiple clock cycles for propagation. Again, it is a path that starts at a timing startpoint and ends at a timing endpoint. However, for a multi-cycle path, the normal constraint on this path is overridden to allow for the propagation to take multiple clocks.

  - When you have different parts of the design that run at different, but related frequencies. Consider a circuit that has some stuff running at 60MHz and some running on a divided clock at 30MHz.
  - Instead of actually defining 2 clocks, you can use only the faster clock, and have a clock enable that prevents the clocks in the slower domain from updating every other clock,
  - Then all the paths from the "30MHz" flops to the "30MHz" flops can be MCP.
  - This is often done since it is usually a good idea to keep the number of different clock domains to a minimum.

3.3 Setup Check:

As shown in figure 3.4, 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.

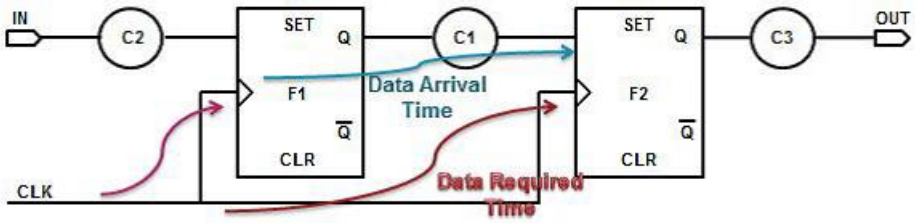


Figure 3.4: Setup and Hold Slack Timings

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.

3.4 Hold Check:

For the shown figure 3.4, 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 performedon the same clock cycle.

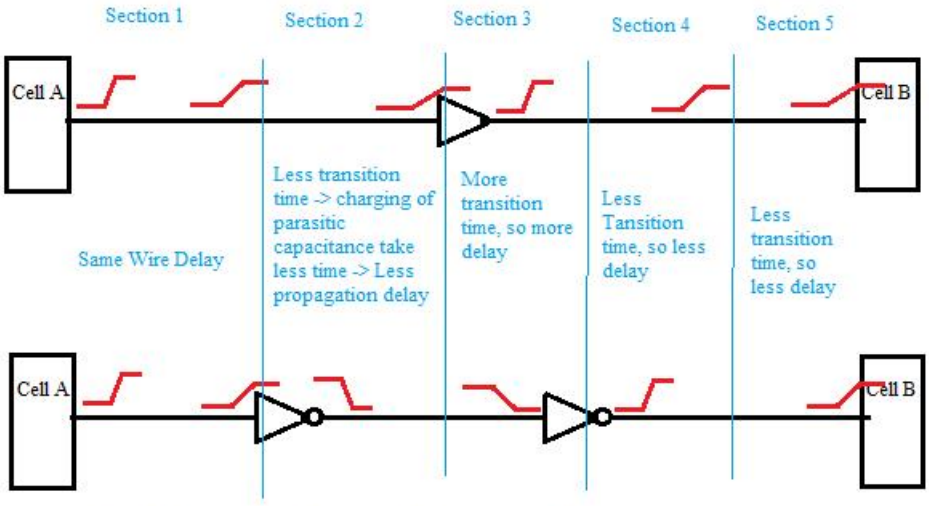
3.5 Ways to fix Setup Violation:

Setup violations are essentially where the data path is too slow compared to the clock speed at the capture flip-flop. With that in mind there are several things a designer can do to fix the setup violations.

Method 1: Reduce the amount of buffering in the path.
It will reduce the cell delay but increase the wire delay. So if we can reduce more cell delay in comparison to wire delay, the effective stage delay decreases.

Method 2: Replace buffers with 2 Inverters place farther apart
Adding 2 inverters in place of 1 buffer, reducing the overall stage delay.

- Adding inverter decreases the transition time 2 times then the existing buffer gate. Due to that, the RC delay of the wire (interconnect delay) decreases.
- As such cell delay of 1 buffer gate ? cell delay of 2 Inverter gate
- So stage delay (cell delay + wire delay) in case of single buffer ¡ stage delay in case of 2 inverter in the same path.
- You will get the clear understanding by following figure and you can refer the first post to understand how transition time varies across the wire.



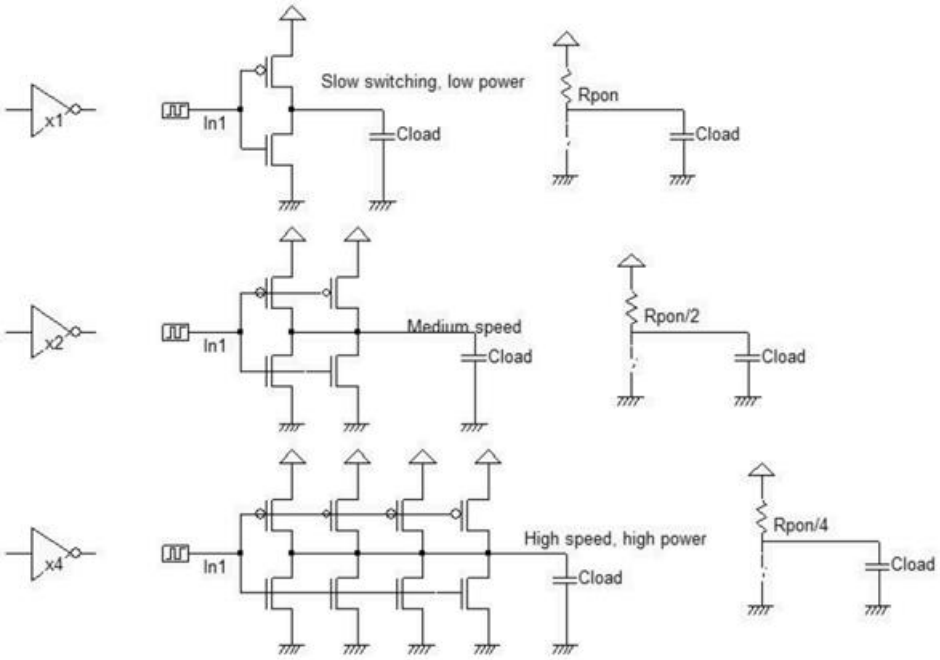
If, Cell delay of Buffer = 2*(Cell delay of NOT gate)
and Driving strength of Buffer = Driving Strength of NOT gate.
Stage Delay (In case of 1 Buffer) > Stage Delay (In case of 2 NOT gate)

Method 3: HVT swap. Means change HVT cells into SVT/RVT or into LVT.

- Low Vt decrease the transition time and so propagation delay decreases.
- HVT/NVT/LVT type cells have same size and pin position. In both leakage current and speed, LVT Greaterthan NVT greaterthan HVT. So replace HVT with NVT or LVT will speed up the timing without disturb layout.
- Negative effect: Leakage current/power also increases.

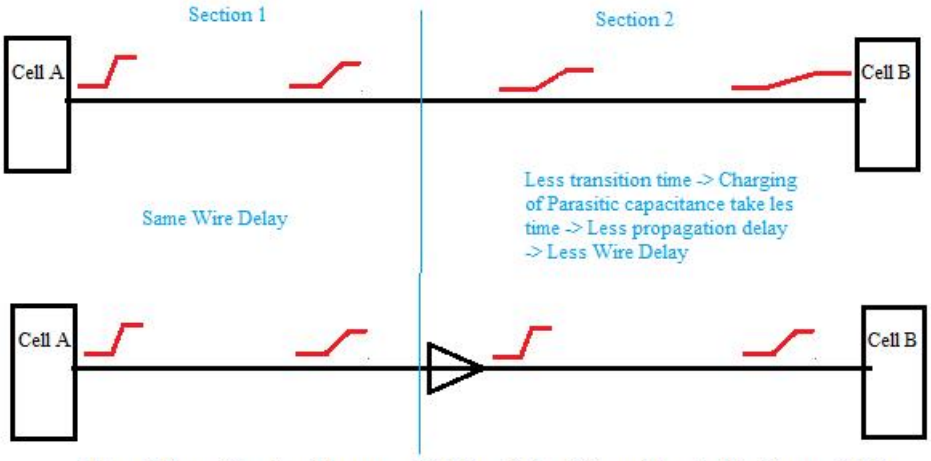
Method 4: Increase Driver Size or say increase Driver strength (upsize the cell)

- Normally larger cell has higher speed. Increasing driver is very commonly used in setup fix.
- **Negative effect:** Higher power consumption and more area used in the layout.



Method 5: Insert Buffers

- Some time we insert the buffer to decrease over all delay in case of long wire.
- Inserting buffer decreases the transition time, which decreases the wire delay.
- If, the amount of wire delay decreases due to decreasing of transition time, over all delay decreases.
- Negative Effect: Area will increase and increase in the power consumption.



Wire Delay of Section 2 in case of Buffer < Wire Delay of Section2 with out Buffer
If , Cell Delay of Buffer < Delta Wire Delay of Section 2
Then Adding Buffer Decrease the over all Stage Delay

Method 6 : Adjust cell position in layout

- Let?s assume there are 2 gate (GATE A and GATE B) separated by 1000um. There is another GATE C placed at the distance of 900um from GATE A.
- If we re-position the GATE C at 500um from GATE A (center of GATE A and B), overall delay between GATE A and B decreases.
- Note: The placement in layout may prevent such movement. Always use layout viewer to check if there are any spare space to move the critical cell to an optimal location.

If you will see closely then you come to know that transition time is very bad in section 2 (for original circuit) in comparison to the transition time in section **3** (rearranged circuit).
That mean,**Delta wire delay in section 2 > Delta wire delay in section 3**

Method 7 : Clock slew
By delaying the clock to the end point can relax the timing of the path, but you have to make sure the downstream paths are not critical paths.

3.6 Ways to fix Hold Violation:

Hold violation is the opposite of setup violation. Hold violation happen when data is too fast compared to the clock speed. For fixing the hold violation, delay should be increases in the data path.
Note: Hold violations is critical and on priority basis in comparison are not fixed before the chip is made, more there is nothing that can be done post fabrication to fix hold problems unlike setup violation where the clock speed can be reduced.
The designer needs to simply add more delay to the data path. This can be done by

Method 1 : By Adding delays

- Adding buffer / Inverter pairs /delay cells to the data path helps to fix the hold violation.
- **Note:** The hold violation path may have its start point or end point in other setup violation paths. So we have to take extra care before adding the buffer/delay.

Method 2 : Decreasing the size of certain cells in the data path

- It is better to reduce the cells closer to the capture flip flop because there is less likely hood of affecting other paths and causing new errors.

Chapter 4

Design Problems & Solutions

4.1 Need of Placement Blockage

4.1.1 Introduction

Placement blockages are used to control placement of cells in specified areas.

1. Placement halos are placement blockages around blocks which prevent cells from being placed inside the halo area.

2. A halo is associated with a block so if the block is moved, the halo moves with it.A placement blockage prevents cell placement in a specific area but unlike a halo it is not associated with any block.

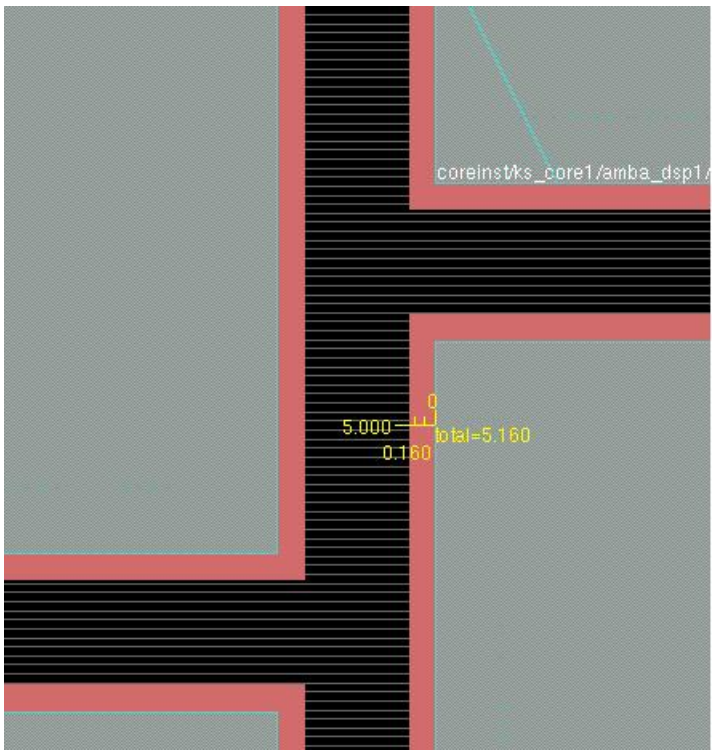


Figure 4.1: Placement Halo

The valid placement blockage types are:

- Hard - The area cannot be used to place blocks or cells. This is the default.
- Partial - Sets a percentage for the maximum cell utilization in this area.
- Soft - The area cannot be used to place blocks or cells during standard cellplacement, but can be used during in-place optimization, clock tree synthesis, ECO placement or placement legalization (refinePlace).

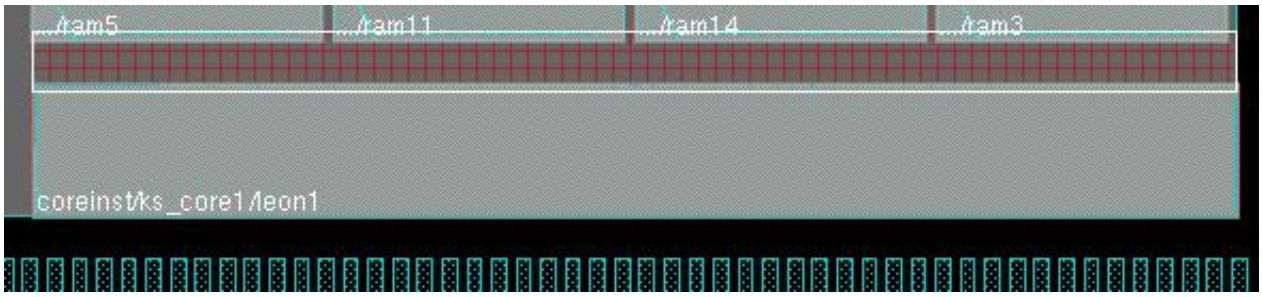


Figure 4.2: Placement Halo

A partial placement blockage can reduce routing congestion by setting a maximum placement density in a specified area. Type to Partial and specify Placement Percentage of 30%. This means the maximum cell density allowed in this area is 30%.

4.1.2 Problem

Placement of some cells is done to the space between two RAMs. Hence logic talking with these cells resulted in longer paths and suffers from setup violation.

4.1.3 Solution

Used hard placement blockage, so no cell is placed to the space between RAMs.

4.2 Need of Routing Blockage

4.2.1 Introduction

Routing blockages prevent routing in an area for specified layers. The router must also space all routing away from the routing blockages based on the spacing rules in the technology file.you can specify the layers prevented from routing in this area.

4.2.2 Problem

While creating power mesh routing, VDD and VSS strapes were generating over Hard Macros on M2, M3, M4 layer. Which results in shorts between metal layers (M2, M3, M4) used inside the Macro and metal layers (M2, M3, M4) used for power mesh routing.

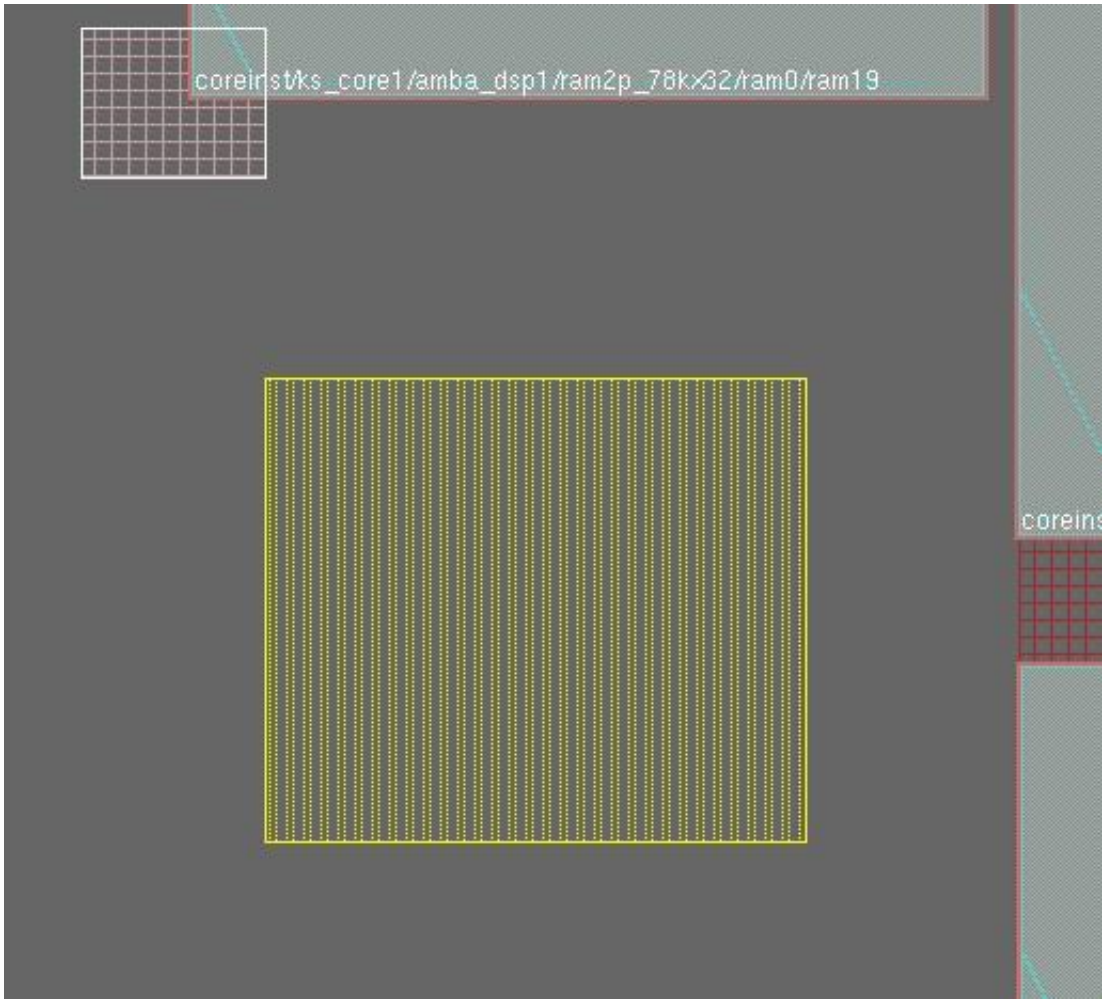


Figure 4.3: Routing Halo

4.2.3 Solution

Created routing blockage surrounding each Macro and specified the layers (M2, M3, M4) to prevent power mesh routing above Macros. Thus there are no shorts.

4.3 Congestion

4.3.1 Introduction

Congestion is a problem that you face when designing chips. Congestion means that more routing resources are needed than you actually have. Congestion can occur locally in a portion of a block or globally for the whole block. Sometimes your designs are congested at different locations with different severity.

Congestion= number of tracks required/actual number of tracks available

Example: Congestion map shows a diamond shape with the following numbers: V = 55/50 H = 5.5/12 The V and H represent vertical and horizontal routing tracks. The

first number indicates the required (or used) tracks , and the second number indicates the total of available tracks.

4.3.2 Problem

An acceptable placement result must be routable. High congestion can lead to longer routes which negatively impact

- Timing,
- increased routing runtime and
- shorts

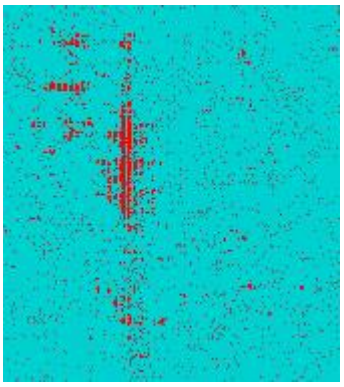


Figure 4.4: Congestion

4.3.3 Solution

For a module or submodule that has route congestion, complete one of the following actions to improve congestion, depending on the type and severity of the violation:

1. Change the block pin orientation. Route congestion usually occurs around blocks that have their pins facing incorrectly. You can identify these blocks by clicking on them in the design display area to see where the flight lines terminate. When the block pins are visible, you can rotate or flip the block(s) to alleviate the congestion.
2. Add density screens You can use density screens to control standard cell placement density in certain areas where there is high routing congestion. Use the Add Density Screens tool to create a screen over the highly congested area.
3. If I/O pins are not sparse enough and rather they are gathered near the corners then it results in congestion near corner area. So try to spread the pins with enough spacing along the edge to avoid congestion.
4. If there isnt case of local congestion but the whole design suffers from high floorplan utilization then solution is to increase floorplan area appropriately.
5. If crosstalk is not the problem in design then avoid shielding of clock nets and routing signals. Because shielding reduces routing resources and causes congestion.

4.4 Effect of Crosstalk and Need of Shielding

Reason for use of shielding is crosstalk. Crosstalk is the undesirable electrical interaction between two or more physically adjacent nets due to capacitive coupling. Crosstalk can lead to crosstalk-induced delay changes or static noise.

4.4.1 Crosstalk Delay Effects

Crosstalk can affect signal delays by changing the times at which signal transitions occur. For example, Figure 4.5 shows the signal waveforms on cross-coupled nets A, B, and C.

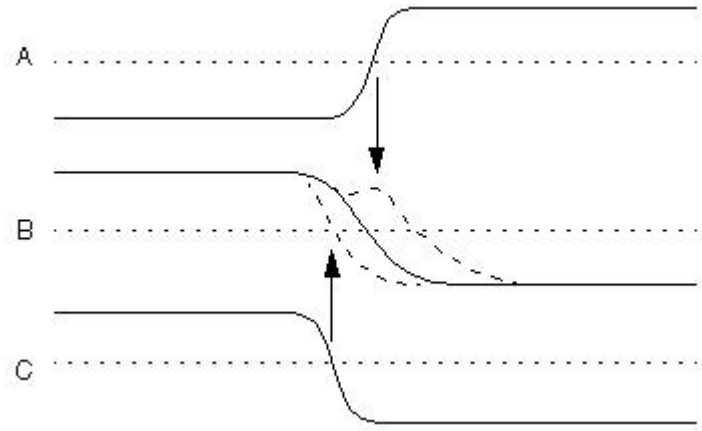


Figure 4.5: Transition Slowdown or Speedup Caused by Crosstalk

Because of capacitive cross-coupling, the transitions on net A and net C can affect the time at which the transition occurs on net B. A rising-edge transition on net A at the time shown in Figure3.5 can cause the transition to occur later on net B, possibly contributing to a setup violation for a path containing B. Similarly, a falling-edge transition on net C can cause the transition to occur earlier on net B, possibly contributing to a hold violation for a path containing B.

Delta Delay and Fanout Stage Effect

Crosstalk effects distort a switching waveform, which adds delay to the propagated waveforms of the fanout stages, as shown in Figure 4.6.

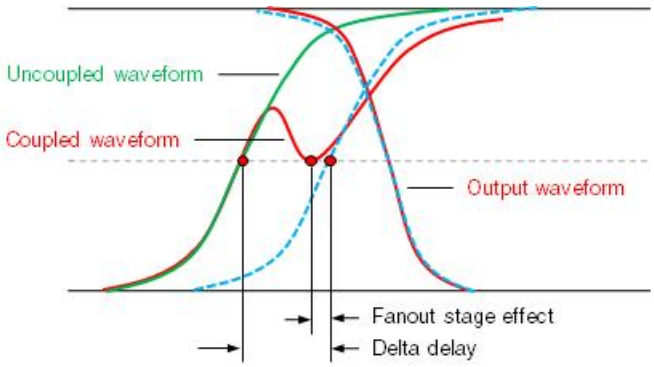


Figure 4.6: Delta Delay Includes the Fanout Stage Effect

4.4.2 Crosstalk Noise Effects

Figure 4.7 shows an example of a noise bump due to crosstalk on cross-coupled nets A and B.

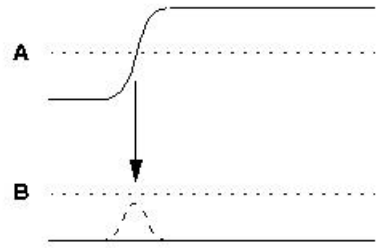


Figure 4.7: Noise bump due to crosstalk

Net B should be constant at logic 0, but the rising edge on net A causes a noise bump or glitch on net B. If the bump is sufficiently large and wide, it can cause an incorrect logic value to be propagated to the next gate in the path containing net B.

4.4.3 Shielding

The router shields routed nets by generating shielding wires that are based on the shielding widths and spacing defined in the shielding rules. In addition to shielding nets on the same layer, there is also an option to shield one layer above or one layer below or the layer above and the layer below. Shielding above or below the layer is called coaxial shielding. Coaxial shielding provides even better signal insulation than same-layer shielding, but it uses more routing resources. Shielding can be performed either before or after signal routing. Shielding before signal routing, which is referred to as preroute shielding, provides better shielding coverage but can result in congestion issues during signal routing. Preroute shielding is typically used to shield critical clock nets. Shielding after signal routing, which is referred to as post-route shielding, has a very minimal impact on routability, but provides less protection to the shielded nets.

4.4.4 Problem

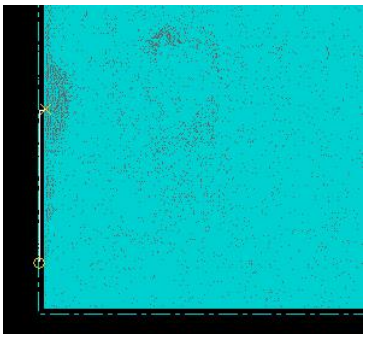


Figure 4.8: Clk pin and first buffer placement

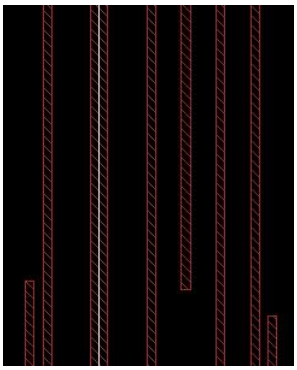


Figure 4.9: Shielding of Clk net

As shown in figure 4.8, long net between clk pin & driving buffer which is surrounded by routed signals.Which increases delay of buffer due to crosstalk effect.

4.4.5 Solution

As shown in figure 4.9, shielding applied to top level clock nets, which creates VSS power net running along with clk net based on the shielding widths and spacing defined in the shielding rules on the same layer. So now buffers in the clock paths dont suffer from undesired crosstalk delay and improves timing results.

4.5 Create Region/Bound

- Creates a region for a module or a group.
- Regions are rectangular or rectilinear regions within which the placement tool attempts to place the associated cells.

4.5.1 Problem

As shown in figure 4.10, logic module which was talking with ram, was placed away from the ram. Which results in longer path, more buffer counts and hence setup violation was coming to those Ram2Ram paths.

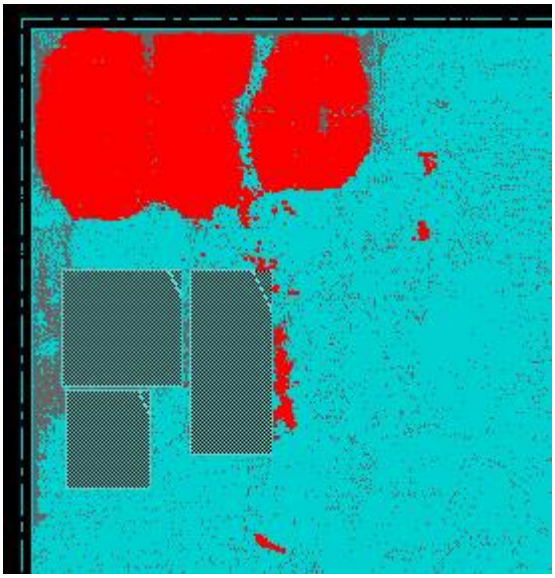


Figure 4.10: no presence of bound

4.5.2 Solution

To avoid long Ram2Ram paths I should place that logic module close to the ram as shown in figure 4.11. So I created a region to place the logic module at my desired place in the floorplan.

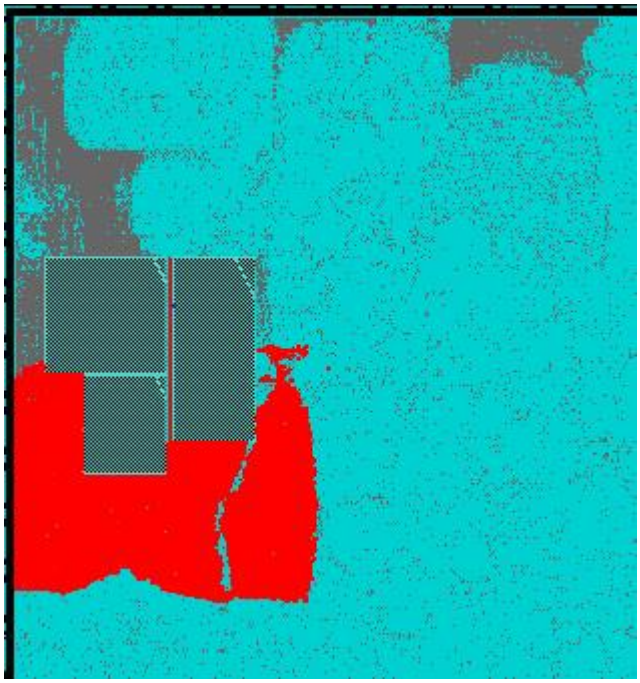


Figure 4.11: placement after creating bound

4.6 Using standard cell library with increased channel length

Implemented same design with standard cells having more channel length at the same target frequency and operating voltage.

Conclusion

	Smaller Chanel length	Larger channel length
Leakage Power	More	Less
Dynamic Power	Less	More
Area	Less	More
Buffer Count	Less	More
Delay	Less	More
Runtime	Less	More

Figure 4.12: Comparison between different technology

- As channel length decreases, we can achieve more and more frequency but at the cost of increased leakage power because of sub-threshold conduction. Because of which gate loses its control over channel and conduction occurs even though VGS is less than Vth.
- Dynamic Power= CV2f V= operating voltage, f= operating frequency, C= channel capacitance As voltage and frequency are same for both trials, the only affecting factor is capacitance. Capacitance increases as channel length increases.So ultimately dynamic power increases for larger channel length trial.
- Channel Area= W.L W= Width of the channel L= Length of the channel So, with increase in channel length, individual cell area increases. Hence floorplan utilization increases.
- Since cell with higher channel length is slow, setup violation will suffer with more negative slack and so tool has to do more buffering to meet the timings. Because of that buffer count and runtime both increases compared to cell with smaller channel length.

4.7 Need of Timing Exceptions

4.7.1 False path

In a false path, there is a logical connection from one point to another. Because of the way the logic is designed, this path can never control the timing. Because both selects can never be 0 concurrently this path doesn't need to be optimized to meet the clock cycle timing from the first to the second flip flop. By setting it false path, tool wont optimize it but still tool will flag it as a path.

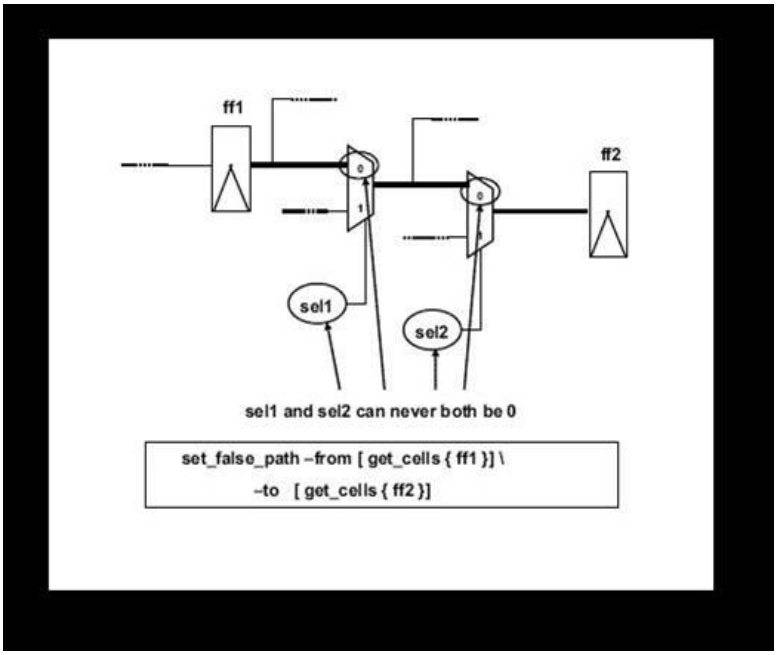


Figure 4.13: False path

4.7.1.1 Problem

- Had self loop (same launching & capture register) path having WNS
- During CTS, tool works on WNS path but since it is self loop path tool is not able to add useful skew.
- Moreover tool wont work on other critical paths also.

4.7.1.2 Solution

- Mark that self loop path as False path for CTS stage and release that path from false path for the later stages.

4.7.2 Multi-cycle path

- A Multi-cycle path in a design is a Register-to-Register path, through some combinational logic where the path will require N number of clock cycles (where N¿1) before the computation is propagated to the destination register.
- Figureshows path P1 that starts at flip-flop U1, goes through gates G1, G3, G5, and G6, and ends at flip-flop U5. This path has a total propagation delay longer than the clock period for CLK1.
- In synthesis, it is encouraged that the designer inform the synthesis tool of any multi-cycle paths. This would allow the synthesis tool to more efficiently optimize the other logic paths that are not meeting the setup requirements rather than to attempt to optimize this multi-cycle path.

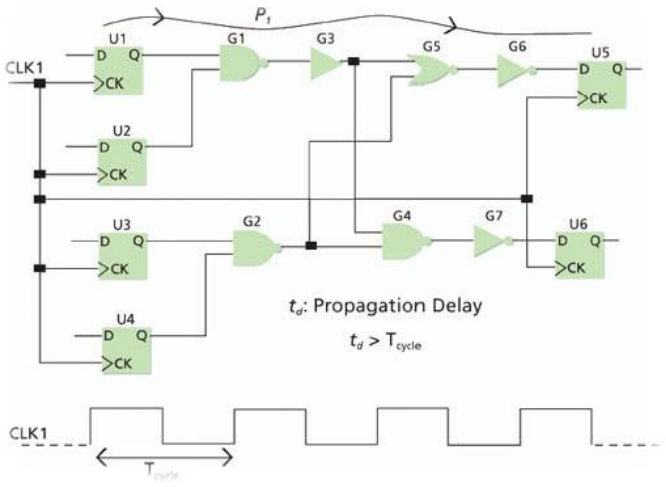


Figure 4.14: Multi-cycle path

- The design on which I am working have multi-cycle paths which are either starting or ending at “DFT” ports.
- As DFT paths are not timing critical, tool wont attempt to optimize these paths by putting them as multi-cycle path.

4.8 Timing Improvement Tricks at Synthesis

4.8.1 Path adjust

- Tightening or relaxing constraints on a selective paths.
- This trick can help closing timing for a small number of violating paths.

4.8.2 Make PathGroup

- If macro is coming in critical Reg2Reg paths which has huge delay, then make a separate path group for all the paths passing through that macro.
- This will greatly improve the optimization of all the other Reg2Reg paths not passing through macro.
- This macro paths can also be optimized by over constraining/path adjust.

4.8.3 Ungrouping

- Determines whether all instances of the specified module should be flattened/inlined during synthesis.
- Tool will work across the module boundaries and would have more room for optimization.

4.8.4 Low Vth standard cells

- Usage of these cells significantly improves timing. But usage of these cells significantly increases the power numbers as these have high leakage power.

Chapter 5

Conclusion & Future scope

5.1 Conclusion

This report provides a brief overview of physical design flow. A technology library, cell libraries and gate netlist are read into the P&R tool. Design constraints are then imported into the P&R tool. The first step in physical layout implementation starts with floor planning. This is followed by power planning, placement, clock tree synthesis and detailed routing. After detailed routing, the implementation of the layout is completed. Before the layout is taped-out, it has to be verified by the sign-off flow, which includes physical verification and post-layout verification.

Physical design flow consists of producing a production-worthy layout from a gate-level netlist subject to a set of constraints. This report focuses on the steps involved in the physical design flow, problems imposed by shrinking process technologies, and the methods used to tackle these problems.The aim of this project is to successfully complete ASIC design flow from RTL to GDS-II, using the advance industry level tools. This project provides a solid base and practical hands-on experience of advanced tools like Cadence Encounter Digital Implementation System (Physical implementation for high-performance, giga-scale, low-power, and mixed-signal designs at advanced and mainstream process nodes),which gives complete solution for Logic Synthesis, Floor planning, Power Network Synthesis, Placement, Clock Tree Synthesis and Routing. Along with this, the analysis of various design factors affecting the performance of the final chip such as power, area and timing is also performed. Power analysis performed by Apache RedHawk tool & timing signoff is done by Synopsys Primetime.

The physical design step is a very important step in manufacturing an IC and System on Chip. My current project involved the design of General Interrupt Controller block. This project has given me very good exposure regarding the physical design flow followed in companies. Also, I learnt the complete physical design flow, and learnt how to tackle various problems related to timing and other quality constraints in a design.

5.2 Future scope

The shrinking of the CMOS technology has been increased very aggressively with ultra-thin sizes. This shrinking of the design creates many significant challenges and reliability issues in design which causes augmented process variations, short channel effects, power densities and leakage currents etc. Continuous shrinking of channel length increases the high speed devices in very large scale circuits.The scaling of bulk CMOS, however, faces significant challenges in the future due to the fundamental material and the process technology limits.

FINFET based transistors are used as choice and solution for CMOS based technology with scaled device geometry. In these device structures, the effect of short-channel length can be controlled by limiting the off-state leakage. Moreover, FINFETs has advantages of suppressing short channel effects, gate-dielectric leakage currents etc.

So in future this design of General Interrupt Controller, which is implemented by me using 28 nm CMOS standard cell library can be implemented using FINFET standard cell library on less than 20 nm technology in order to achieve significant optimization of power, performance and area using the same RTL, as RTL is technology independent.

Chapter 6

References

1. Michael John Sebastian Smith [June 1997], Application-Specific Integrated Circuits, Addison-Wesley Publishing Company, VLSI Design Series.

2. Vishwani D. Agrawal, Srivaths Ravi, Low-Power Design and Test, July 2007.

3. Solvnet, Synopsys. http://www.solvnet.synopsys.com

4. Gordon E. Moore, Cramming More Components onto Integrated Circuits, Electronics.Vol.38, April 19, 1965

5. Michael John Sebastian and John Smith, Application-Specific Integrated Circuits, Wesley Publishing Company, 1997.

6. M. Keating and P. Bricaud, Reuse Methodology Manual for System-on-Chip designs, 2nd Edition, Kluwer Academic Publishers, Norwell 1999.

7. F. Balarinet. al, Hardware-Software Co-Design of Embedded Systems, The POLIS approach, Kluwer Academic Publishers, 1997.

8. D. E. Lackey, Applying Placement-based Synthesis for On-time System-on-a-Chip Design, IEEE Custom Integrated Circuits Conference, 2000, pp. 121-124.

9. Cadence Support http://support.cadence.com