Physical Implementation of High Speed Serial Interface

Major Project Report

Submitted in the partial fulfillment of the requirements for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Bhatt Rahul S.
(14MECV03)

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2016

Physical Implementation of High Speed Serial Interface

Major Project Report

Submitted in the partial fulfillment of the requirements for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Bhatt Rahul S.
(14MECV03)

Under the guidance of

External Project Guide:

Mr. Sourav Banerjee
Structural Design Engineer,
Intel India Technology Pvt. Ltd.,
Bangalore.

Internal Project Guide:

Dr. N. P. Gajjar
PG Coordinator (Embedded System),
Institute of Technology,
Nirma University, Ahmedabad.

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2016

Certificate

This is to certify that the Major Project entitled **"Physical Implementation of High Speed Serial Interface"** submitted by **Bhatt Rahul S. (14MECV03)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, 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

Internal Guide

Dr. N. P. Gajjar
PG Coordinator,
Embedded System, EC

Program Co-ordinator

Dr. N. M. Devashrayee
PG Coordinator,
VLSI Design, EC

Director

Prof. P. N. Tekwani
Head of EE Dept.
Director, IT-NU

Certificate

Intel Technology India Pvt. Ltd.

This is to certify that the Major Project entitled **"Physical Implementation of High Speed Serial Interface"** submitted by **Bhatt Rahul S (14MECV03)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Bangalore

Mr. Sourav Banerjee
Structural Design Engineer,
IPG-MIG India,
Intel Technology India Pvt. Ltd.,
Bangalore.

Declaration

This is to certify that

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

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

- Bhatt Rahul S.
14MECV03

Acknowledgements

First and foremost, sincere gratitude to my managers Mr. Sourav Banerjee and Mr. Shivraj Thakare. Also I want to thank Intel Technology India Pvt Ltd, Bangalore for assigning me such project and guide me through.

I would also thank to my Project Guide, Dr. N. P. Gajjar, Professor & PG Coordinator, Embedded System, Institute of Technology, Nirma University, Ahmedabad for being a source of inspiration, giving valuable support and timely guidance for the project work.

I would like to express my gratitude and sincere thanks to our Director Dr. P. N. Tekwani, Head of Electrical Engineering Department for allowing me to undertake this thesis work and for his guidelines during the review process.

I would like to thank my Program Coordinator, Dr. N. M. Devashrayee, Professor, EC (VLSI Design), Institute of Technology, Nirma University, Ahmedabad for giving valuable support and motivation throughout the academic period.

I would like to express my gratitude and sincere thanks to my mentors Mrs. Neeharika Kavuri and Mr. Srivathsava NL at Intel Technology India Pvt Ltd, Bangalore for their valuable guidance throughout this period. They have given me valuable advices and support for my project work which I am very lucky to benefit from.

I take immense pleasure to thank my team members Mr. Santosh Kumar and Mr. Vinodh Cherukuri and other colleagues in the Intel, special thanks for helping me on this path and for making project at Intel more enjoyable.

I wish to thank my classmates for their delightful company which kept me in good humor throughout the journey.

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.

- Bhatt Rahul S.
14MECV03

Abstract

The processing speed of the single chip today has increased drastically since the beginning of the IC technology. The interconnection bandwidth is the factor which is degrading the performance of the digital system so our target is to achieve higher off chip data rate by applying different manipulation in placement and routing of the design. The high speed serial interface is implemented at the physical layer and our focus area is to implement the complete flow from RTL (Register Transfer Logic) to GDS (Graphic Database System) of this interface. The speed of this interface is in the range of few tens of Gbps. The synthesized gate level netlist is generated from the RTL which also has the influence of physical constraints and timing constraints. Formal Verification is carried out to ensure logical equivalency between different abstraction levels. Auto Place and Route (APR) is implemented on synthesized netlist with physical constraints and timing constraints keeping in mind. First placement is exercised with physical constraints, then Clock Tree Synthesis (CTS) and Routing is implemented bounded by timing constraints such that slack is balanced. After routing, the parasitic report is given for PV (Performance Verification) to close the timing violation. LVS (Layout vs Schematic) and DRC (Design Rule Check) cleanups are done to ensure manufacturability. RV (Reliability Verification) checks for the IR drops and Electromigration issues. After cleaning up all these physical verification flows the final GDS file is given for fabrication. This implementation is carried out at the submicron technology node below 50 nm.

Contents

List of Figures

Abbreviations

AER Advanced Error Reporting

APR Auto Place and Route

CTS Clock Tree Synthesis

DC Design Compiler

DEF Design Exchange Format

DRC Design Rule Check

ECO Engineering Change Order

EM Electromigration

FEV Formal Equivalency Verification

GDS Graphic Database System

HDL Hardware Description Language

HFN High Fanout Net

ICC IC Compiler

ICG Integrated Clock Gating

LVS Layout Vs. Schematic

MGT Multi-Gigabit Transceiver

PCI Peripheral Component Interconnect

RMS Root Mean Square

RTL Register Transfer Logic

RV Reliability Verification

SAIF Switching Activity Interchange Format

SDC Synopsys Design Constraint

SPEF Standard Parasitic Exchange Format

STA Static Timing Analysis

TTM Time To Market

TNS Total Negative Slack

VCD Value Change Dump

WLM Wire Load Model

WNS Worst Negative Slack

Chapter 1

Introduction

As we move further on the path shown by Gordon Moore, the individual chips are getting faster and denser. This results in higher chip performance. But in today's era, the computational complexity has increased and also the functionality requirements are increasing. To fulfill such requirements, we need different chips integrated with one another in such a way that the overall performance of SoC increases and functionality stays intact. But somehow the interconnection bandwidth between the chips is a major hurdle in a way of getting the efficient overall performance.

1.1 Problem Definition

The main objective here is to get the final layout from the RTL code with required speed and performance but without any architectural changes and PV violations to ensure that the intended functionality of the design holds good. This is achieved by manipulating the physical implementation flow of the design which includes synthesis and placement and routing. The timing and physical constraints are predefined for the design so all the modifications in APR will be bounded by these constraints.

After APR, there will be necessity to verify the correctness of the generated layout. These checks will be incorporated in the Physical Verification flows.

So in general, entire flow is divided in three major aspects:

1. Synthesis
2. Physical Implementation
   (a) Floorplanning
   (b) Placement

(c) Clock Tree Synthesis (CTS)

(d) Routing

3. Physical Verification

(a) Design Rule Check (DRC)

(b) Layout vs. Schematic (LVS)

(c) Antenna Rule Checking

(d) Reliability Verification (RV)

Figure 1.1 gives the overview of the physical design flow.

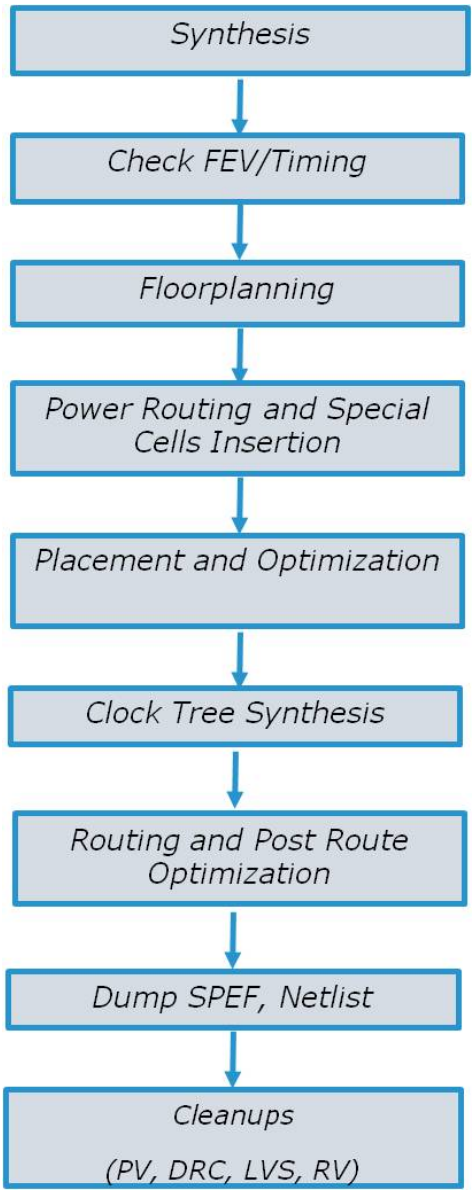

Figure 1.1: Physical Design Flow

Also after each stage, it is necessary to check the logical equivalency of the design to make sure that the changes made during the undergone stage have not altered the functionality or architecture of the design. And it is carried out by Formal Equivalence Verification (FEV).

The area of focus is to make improvements in following aspects of the design using the automated physical design flow.

1. Timing optimization.
2. Leakage power optimization.
3. Checking the logical consistency between the RTL and the netlist generated after synthesis and APR (Auto Place and Route) using mapping tools.
4. Perform STA (Static Timing Analysis) and verify whether clocks are properly defined and parasitic annotation has been done properly.
5. Check for the Electromigration and IR drop issues using standard analysis tools.
6. Utilized area optimization

Chapter 2

Literature Review

2.1 Multi-Gigabit Transceiver

A Multi-Gigabit Transceiver (MGT) is a SerDes fit for working at serial bit rates above 1 Gbps. MGTs are utilized progressively for information correspondences in light of the fact that they can keep running over longer separations, use less wires, and in this way have lower expenses than parallel interfaces with identical information throughput.

2.1.1 Function of MGT

Like different Serdes the MGT is capable to transmit parallel data as serial bit straem, and transform the received serial bits to parallel data. The most essential execution metric of a MGT is its serial bit rate which is the quantity of serial bits it can send or get every second. Despite the fact that there is no strict tenet, MGTs can commonly keep running at line rates of 1 Gbps or more. MGTs have turned into the 'data thruways' for data processing syatems that request a high in/out unprocessed data information and yield (e.g. video handling applications). They are turning out to be extremely basic on FPGAs-such programmable logic devices being particularly all around fitted for parallel data handling algorithms.

2.1.2 Signal Integrity and Jitter

Because of the high line rates of MGTs, signal integrity is basic for them. Bit Error Rate (BER) of the connection and jitter describe The nature of a given fast link. BER is the ratio of bits received in error to total bits received.

Jitter and BER are functions of the entire MGT connection which includes the power supplies, reference clocks, serial lines of MGTs and the digital systems that generate and consume their parallel information. Accordingly, the performance of the MGTs are measured by how little jitter they transmit and how much jitter they can endure before their BER

gets too high.

MGTs are utilized as a part of the usage of the below serial protocols:

- 10 Gigabit Ethernet
- Fibre Channel
- Gigabit Ethernet
- Infiniband
- PCI Express
- Serial ATA
- SerialLite
- Serial RapidIO

2.2 Introduction to PCI-Express (PCIe)

Peripheral Component Interconnect (PCI) is a fast parallel bus initially composed by Intel to associate I/O peripherals to a CPU. PCI-Express is a developmental form of PCI that keeps up the PCI programming utilization demonstrate and replaces the physical bus with a fast serial bus serving numerous lanes.

The PCI-Express (PCIe) bus engineering (once 3GIO or third Generation I/O) was presented by Intel, in association with other driving organizations, including IBM, Dell, Compaq, HP and Microsoft, with the goal that it will end up being the predominant standard for PC I/O in the years to come. PCIe has various upgrades over the more seasoned measures, including lower I/O pin count and lesser physical footprint, higher maximum system bus throughput, better execution scaling for bus device, local hot-plug usefulness and a more nitty-gritty error identification and reporting system (Advanced Error Reporting, AER). Later modifications of the PCIe standard give equipment backing to I/O virtualization.

The expressed objective of PCI-Express is to give:

- A neighborhood bus for chip-to-chip interconnects
- A method to upgrade PCI slot performance at lower expenses

Format specifications are kept up and created by the PCI-SIG (PCI Special Interest Group), a gathering of more than 900 organizations that additionally keep up the routine PCI details. PCIe 3.0 is the most recent standard for development cards that is underway and accessible on standard PCs.

2.3 Overview of PCIe

Theoretically, the PCIe bus is similar to a high speed serial substitution of the more seasoned PCI/PCI-X bus, an interconnect bus utilizing shared address/data lines.

A key distinction between PCIe bus and the more established PCI is the bus topology. PCI utilizes a mutual parallel bus design, where the PCI host and all gadgets share a typical arrangement of address/data/control lines. Interestingly, PCIe depends on point-to-point topology, with isolated serial connections associating each device to the root complex (host).

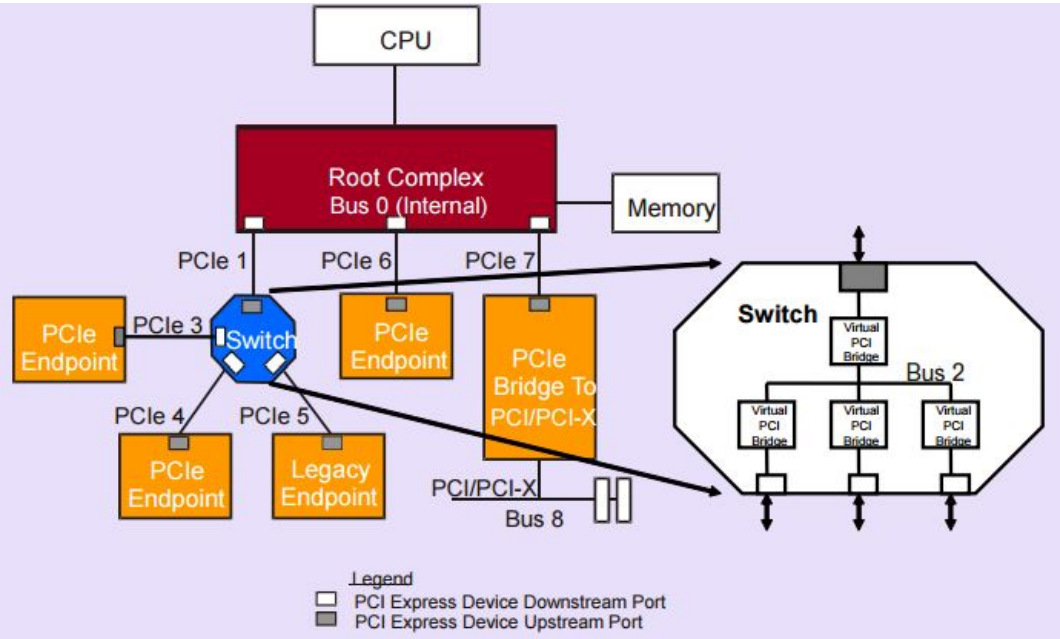

Figure 2.1: PCIe Topology

Because of its shared bus topology, access to the more seasoned PCI bus is arbitrated, and restricted in a single direction to one master at a time. Furthermore, the more seasoned PCI clocking scheme confines the bus clock to the slowest peripheral on the bus (paying little mind to the devices included in the bus exchange). Conversely, a PCIe bus link supports full-duplex correspondence between any two endpoints, with no innate restriction on simultaneous access over different endpoints.

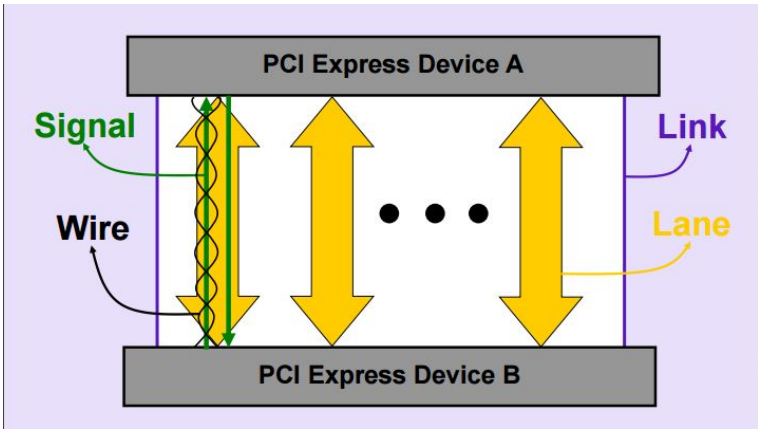

Figure 2.2: PCIe Terminology

2.3.1 Interconnect

PCIe devices transfer information by means of a logical connection called interconnect or link. A link is a point-to-point communication channel between two PCIe ports. It permits these channels to send and receive typical PCI requests (setup, I/O or memory read/write) and interrupts. A link may be made out of multiple lanes at the physical level. Low-speed peripherals, (for example, a 802.11 Wi-Fi card) utilize a solitary lane (×1) link, while a graphics adapter ordinarily utilizes a much more extensive and speedier 16-lane link.

2.3.2 Lane

A lane is made out of two differential signaling pairs. Out of these two pairs, one pair is used for accepting the data and the other for sending the data. In this manner, every lane is made out of four wires or signal. Theoretically, every lane is utilized as a full-duplex byte stream, transporting data packets in eight-bit "byte" format simultaneously in both directions between endpoints of a link. Physical PCI Express slots might contain from 1 to 32 lanes, all the more correctly 1, 2, 4, 8, 12, 16 or 32 lanes. Number of lanes are composed with a "×" prefix and for typical use, ×16 is the biggest size.

As far as bus protocol is concerned, PCIe communication is carried out in packets as appeared in Fig 2.3. The transaction layer of the PCIe port takes care of packetizing and de-packetizing data and status-message movement.
Radical differences in electrical signaling and bus protocol require the utilization of an alternate mechanical structure element and development connectors and accordingly, new motherboards and new connector sheets; PCI slots and PCIe slots are not exchangeable. At the product level, PCIe protects backward compatibility with PCI; legacy PCI framework software can recognize and arrange more up to date PCIe devices without express backing for the PCIe standard, however PCIe's new features are difficult to reach.

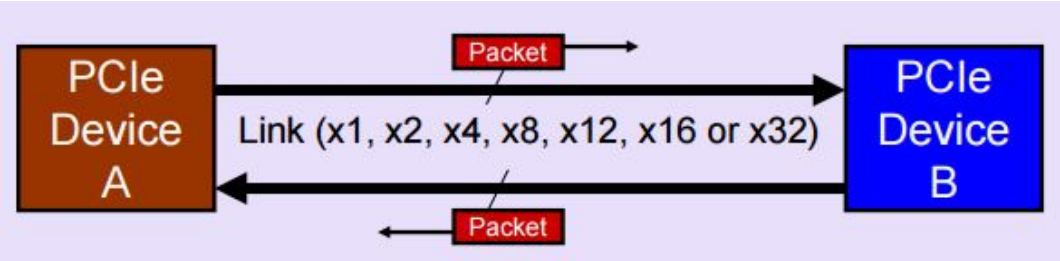

Figure 2.3: Packet based Transaction in PCIe

It takes anywhere between 1 to 32 lanes to create a PCIe link between two devices. In a multi-lane link, the packet data is striped across lanes, and peak data throughput scales with the general link width. The lane count is consequently arranged during initialization of the device, and can be limited by any of the endpoints. For instance, a single lane PCIe card can be embedded into a multi-lane slot, and the initialization cycle auto-arranges the most noteworthy mutually supported lane count. On the off chance that awful or untrustworthy lanes are available, then to give failure tolerance the link can powerfully down-design itself to utilize less number of lanes. The PCIe standard characterizes slots and connectors for various widths: ×1, ×4, ×8, ×12, ×16 and ×32. This kind of definition permits PCIe bus to serve both cost-touchy applications where we do not require high throughput, and in addition performance-critical applications, for example, enterprise storage (SAS or Fiber Channel), 3D graphics and networking (10-gigabit Ethernet).

As a perspective, PCIe 1.0 device consisting of four lanes (×4) and PCI-X (133 MHz 64-bit) device have generally the same peak single-direction exchange rate of 1064 MB/sec. The PCIe bus can possibly perform superior to the PCI-X bus in situations where numerous devices are exchanging data at the same time, or if communication with the PCIe peropheral is bidirectional.

2.3.3 Serial Bus

The reinforced serial bus architecture was picked over the customary parallel bus because of intrinsic confinements of the last mentioned, including half-duplex operation, abundance signal count, and characteristically bring down transfer speed because of timing skew. Timing skew results from isolated electrical signals inside of a parallel interface going through transmitters of various lengths, on conceivably diverse printed circuit board (PCB) layers, and at perhaps distinctive signal speeds. Regardless of being transmitted all the while as a single word, signals on a parallel interface experience diverse travel times and land at their destinations at various minutes. At the point when the clock rate of an interface is expanded to a point where the period of the clock is shorter than the longest conceivable time between signal arrivals, the signals does not arrive with sufficient coincidence and re-

covery of the transmitted word becomes difficult. Since timing skew over a parallel bus can sum to a couple of nanoseconds, the subsequent data bandwidth limitation is in the scope of hundreds of megahertz.

A serial interface does not display timing skew in light of the fact that there is one and only differential signal in every direction inside of every lane, and there is no outer clock signal since timing data is installed inside of the serial signal itself. In that capacity, ordinary bandwidth limitations on serial signals are in the multi-gigahertz range. PCIe is only one sample of the general pattern toward supplanting parallel busses with serial interconnects. Different samples incorporate Serial ATA, USB, SAS, FireWire (1394) and Rapid I/O.

Chapter 3

Methodology for Physical Design Implementation

3.1 Synthesis

Synthesis of a design is the first and foremost step for physical implementation of a design. We use Design Compiler for logic synthesis, which transforms a outline portrayal described in a Hardware Description Language (HDL), for example, VHDL or Verilog, into an optimized gate netlist which is mapped to a particular logic library. When the synthesized design meets power, timing, functionality and other design goals, we can go ahead with IC Compiler for physical implementation.

The synthesis flow for the Design Compiler is detailed below:

1. It takes HDL code of the design as an input.

2. Design Compiler makes use of different libraries as shown in the figure 3.1. Design Compiler makes an interpretation of the HDL to appropriate cells which are inferred from GTECH library and Design Ware Library. All the fundamental logic gates and FFs are mapped into the GTECH library whereas more complex cells, for example, substractors and multipliers etc. are included in DW library. To create the schematic view of the design, Symbol library is used by Design Compiler.

3. Target library is the specific technology library to which DC maps the optimized gate level design within the restriction imposed by the constraints. Synthesis process is carried out in such a manner that it optimizes design power, area, timing and functionality by taking into account the combination of best possible library cells suited for the design.

4. Now the design is at the stage where it is suited for the test synthesis. Test synthesis

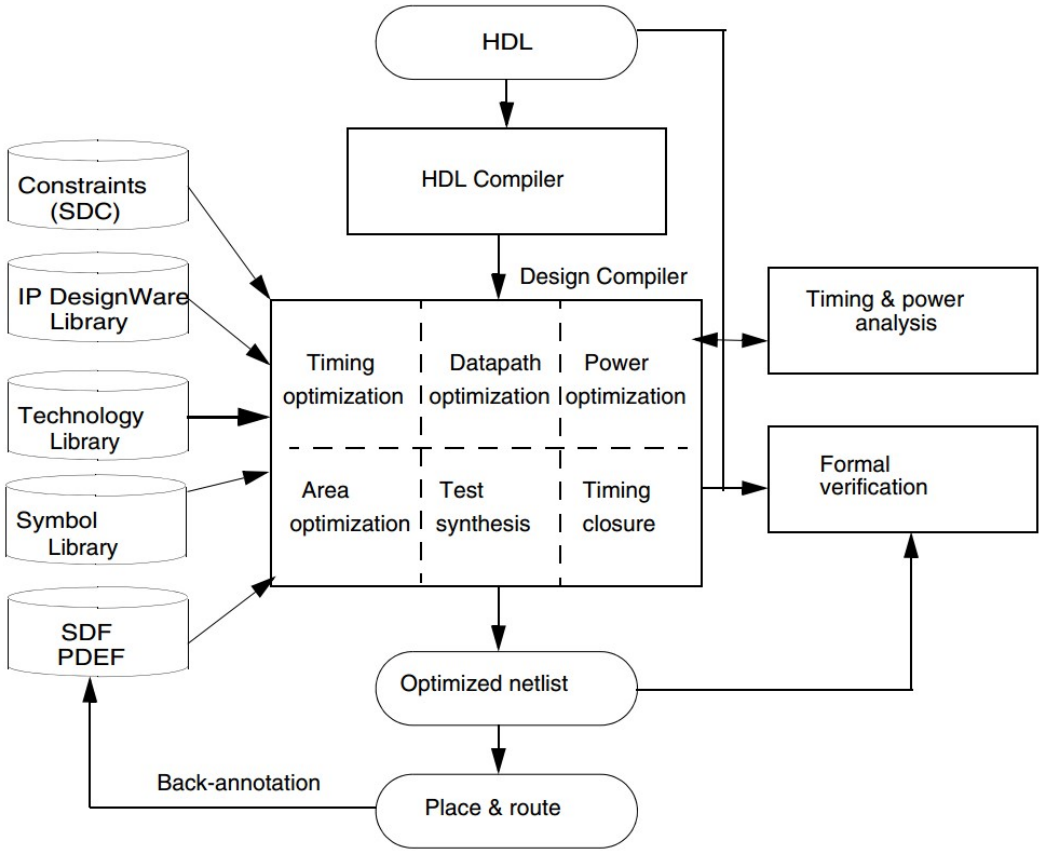

Figure 3.1: Design Compiler Flow

is the process by which designers can integrate test logic into a design during logic synthesis [3]. By test synthesis, we can determine whether our design is testable or not and if not then we can debug the problem at the early stage to meet the time to market deadline. After this we will get optimized gate level netlist that also includes all the cells and connection information.

5. Design is now ready to be placed and routed. APR tools are used to place all these cells and route their interconnects. With this actual routing, we have the practical delays for the interconnects. This information is back annotated into synthesis to get accurate timing results that converge with APR database.

3.1.1 Basic Synthesis Flow

Below are the steps for basic synthesis flow.

1. Generate HDL Files
   HDL file written in VHDL or Verilog is given as an input to the synthesis tool. To get the best possible synthesis outcome, our design has to be described using these HDLs very carefully.

2. Specify Libraries

   We need to give the different libraries to the synthesis tool by means of the proper commands shown in the figure 3.2

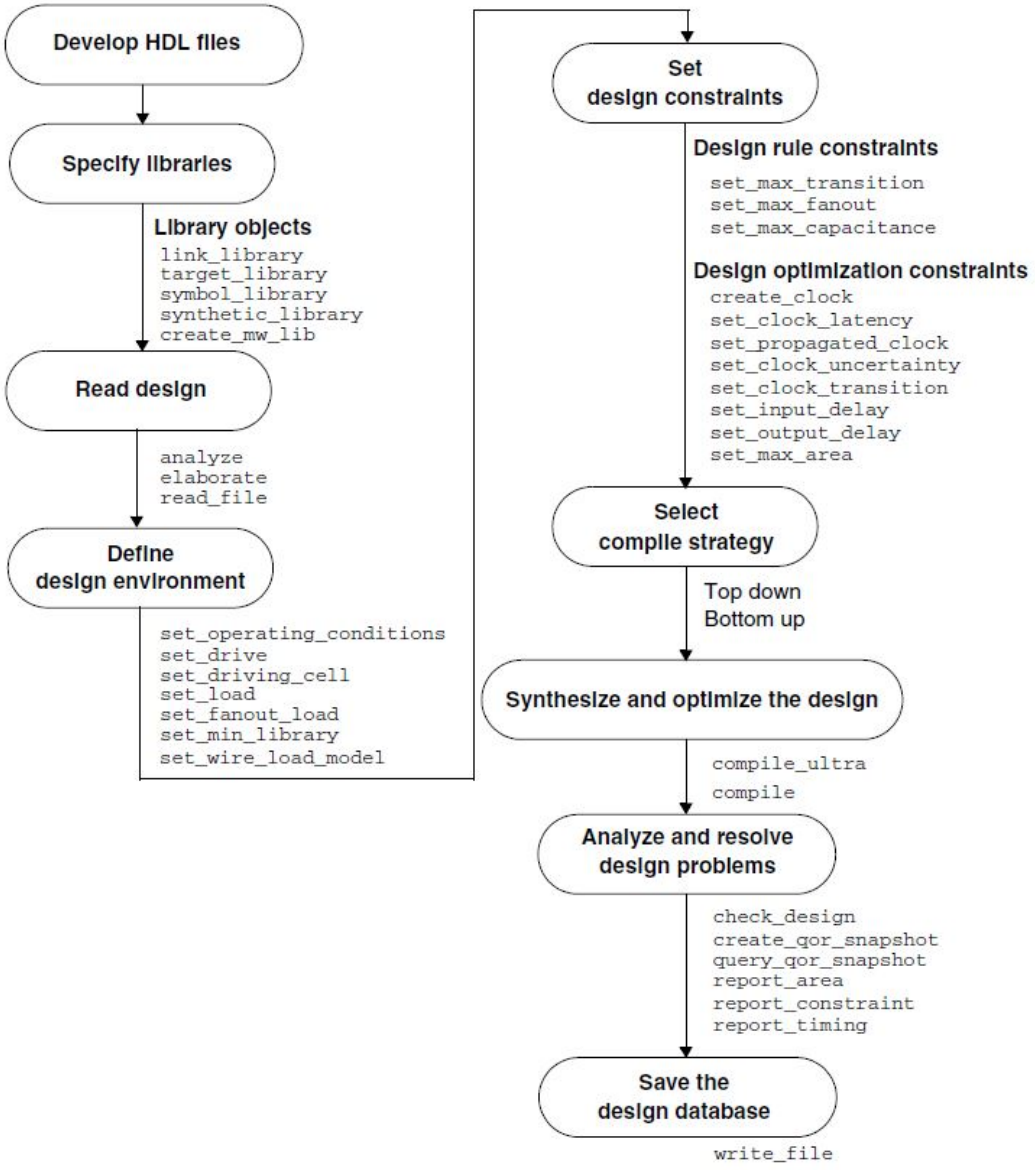

Figure 3.2: Basic Synthesis Flow

3. Read Design

   Synthesis tool can read the design using RTL as well as gate level netlist. It also analyze and elaborate the design according to its environment. HDL Compiler is used to read design.

4. Define Design Environment

   Synthesis tool also requires the environmental conditions in which the design will

be synthesized. These information is also modelled by specifying the appropriate operating condition (PVT information), drivers, loads, fanouts etc.

5. Set Design Constraints
   To restrict the synthesis of the design, the synthesis tool uses the constraints and design rules defined by the designer. Constraints constitutes of the different goals of the design for area and timing. Synthesis tool tries to converge to these goals without violating the design constraints.

6. Select Compile Strategy
   Top down and bottom up are the two fundamental compile strategies to optimize the design.

7. Optimize the Design
   Commands used to start the actual optimization process are shown in the figure 3.2. Also different options can be linked with these commands to get the required outcome.

8. Analyze and Resolve Design Problems
   After the completion of the optimization process, various reports are dumped by the synthesis tool. We need to analyze all the reports carefully to check whether the requirements are met or not.

9. Save the Design Database
   If we get the desired results after the optimization process, we need to save the design changes in the database explicitly.

3.2 Auto Place and Route (APR) Flow

After the synthesis is done, the generated optimized gate-level netlist is used to generate GDS. For that we use IC Compiler, which is one of the industry standard Place and Route tool for designing the automated digital layout. There are many stages involved in physical design.

The inputs to IC compiler, after synthesis has been done, are the gate level netlist generated after Design compiler and the constraints file which are being applied while synthesizing the RTL logic. Also we need the pin location DEF file for accurate floor planning.

During the process, IC Compiler goes through following stages.

1. Initial Design

2. Placement

3. Clock Tree Synthesis
4. Routing and Optimization
5. Base fill
6. Metal fill
7. DRC Optimization
8. Final GDS extraction and Reports dumping

3.2.1 Initial Design

In this stage we create milkyway design library with proper technology file, bus naming style, and reference control file with full paths. After this we read the verilog netlist which has been the output of the RTL synthesized using Design compiler. Then the Design constraints are read. Then we apply timing derates for both Cell and Net in the design. So in this step, all the initial design requirements are defined for the tool to optimize design in upcoming stages.

3.2.2 Floorplanning

Floorplanning is the process of defining the placement region. During floorplanning, the ports and hard macros are assigned a physical location. We use floorplan to precisely map the cell placements and to converge the timing after the APR stage. The floorplan information can be given to the tool by different methods i.e. it can extracted from the tool, can be extracted from DEF file or manual insertion. Floorplan is given to get the information about shape and area of the design, location of macro and ports, placement blockages and voltage area.

To generate the floorplan we need to read the DEF (Design Exchange Format) file which also contains the Power and Ground ports. Also in this stage we have to ensure that all the power/ground pins of the cells are properly connected to power/ground ports.

Points to consider during floorplanning:

1. Avoid narrow channels between hard macros. This leads to routing congestion.
2. Understand the high level dataflow diagram to aid in the placement of ports/hard macros.
3. Avoid placing hard macros/blockages at the center of the floorplan.

4. Take care of any special routing/spacing requirements for hard macros. These are more critical for the analog macros.

Keeping above points in mind we need to create the floorplan with proper geometries from all the four corners to the die core. Tool can itself create floorplan which might not be as robust as the floorplan created by the designer. Also the height and width of the die should be integral multiples of the tile. For well proximity effect we need to add both horizontal and vertical end cap cells. Also there needs to be corner cells which are specially created for both vertical and horizontal diffusion to take place during the fabrication. Tap cells needs to be added otherwise there will be no substrate connection and there might be latch up issues coming due to this.

3.2.3 Placement Optimization

During placement all the logic gates created during synthesis are assigned a physical placement site. To get the estimate of timing placement makes the use of parasitic values of resistance and capacitance from virtual route. Virtual route is defined as the minimum Manhattan distance between two pins. Prior to placement we need to remove Wire Load Model as Virtual Route produces more accurate results.

It is an automated flow in IC Compiler tool. IC Compiler considers the following for the placement:

- Logic connectivity
- Timing
- Congestion/Density

Placement is performed in four stages of optimization:

1. Optimization Before Placement: The pre placement netlist is enhanced at this stage. Nets having the higher fanouts (HFN) are gave way. It can likewise downsize the cells.
2. Optimization During Placement: With respect to Virtual Route, the logic is re-improved. It is capable of cell estimating, cell moving, cell bypassing, net part, duplication of gate, addition of buffer, recover the area. Improvement performs multiple cycles of setup settling, congestion driven placement and incremental timing.
3. Optimization After Placement and Before CTS: At this stage, netlist is optimized by taking into consideration the ideal clock. Also global routing is taken into account during the optimization. Setup, Hold, Max Cap, Max Trans violations are expected to be cleaned during this stage.

4. Optimization After Placement and After CTS: Timing is optimized with actual propagated clock and also measures are made to safeguard the clock skew.

3.2.4 Clock Tree Synthesis

After the standard cells are placed inside the die core area we need to insert clock tree. Clock Tree Synthesis (CTS) is the process of building a clock tree structure in the design. The goal of CTS is to achieve a uniform delay from the clock source to the endpoints and to minimize skew. Flop/Latch clock pins, hard macro clock pins are considered default clock balance points.

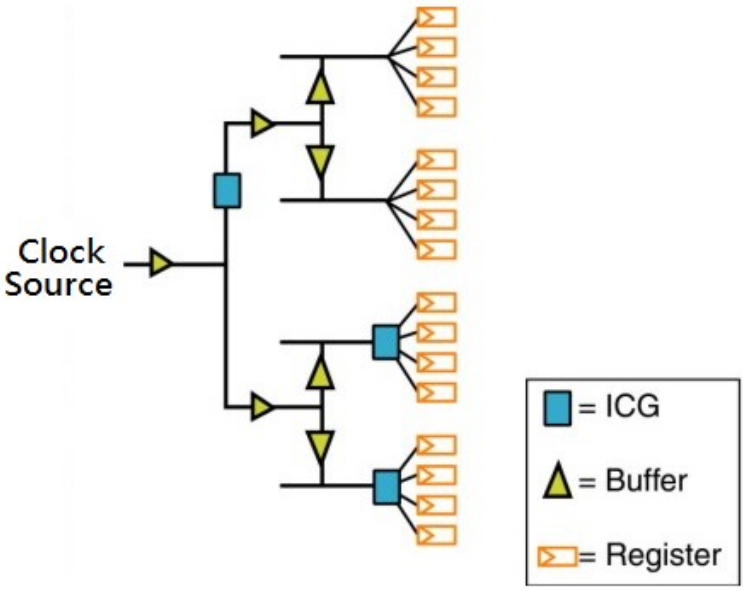

Figure 3.3: Clock Tree Synthesis

We have used the Z router for optimizing the clock skew so that clock reaches all the pins of standard cells at expected time. In clock tree synthesis we use special cells which have equal rise and fall transition values defined in the libraries given by Foundries. All the clock nets are given double width by creating a net routing rule. Also library cell spacing rules are defined in this stage. After this a sanity check is made by checking the physical design before proceeding forward.

When we want to do interclock delay balancing, we use clock_opt command with interclock as switch. When we want to update the IO latency before we start the post CTS optimization, we use clock_opt command with update_clock_latency as switch. Then we connect all the power and ground pins with the corresponding power and ground nets.

To reduce the dynamic power consumption of the clock tree network, we can use the advanced clock-gating technique known as XOR self-gating. The XOR self-gating technique

turns off the clock signal during the clock cycles when the data in the register remains unchanged. Figure 3.4 shows an example of a register gated by an XOR clock-gating cell. The clock gate is enabled only when switching activity occurs at the register.

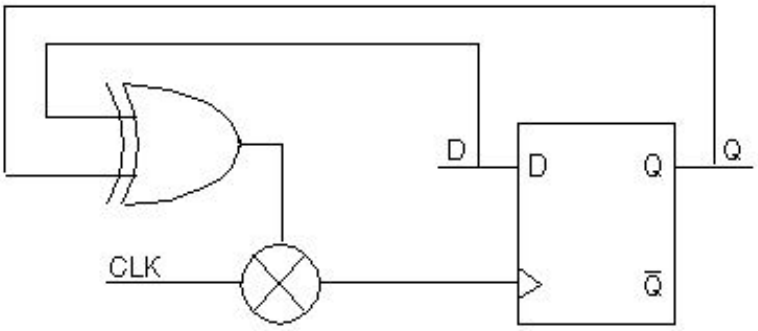

Figure 3.4: Example of XOR Self-Gating

After CTS, all the clocks in the design are propagated. Then we allow the tool to fix the timing violations for all the clocks in the design. Then we extract the parasitic resistance and capacitance by using the command extract_rc. When the design has congestion issues post CTS, we use refine_placement with proper congestion effort as medium or high depending on the requirement and we can perform layer optimization to improve existing buffer tree. Optionally we can insert a diode before routing to avoid antenna violations on the ports of the block.

While routing the clock nets for better design, always enable crosstalk aware routing conditions for signal integrity. If we enable this, tool itself creates the shield wires, if required, to prevent the cross talk. After this we have to dump two reports; one is global skew report and local skew report for analysis purpose. Also reports on capacitance and transition violations can be dumped.

3.2.5 Routing and Optimization

Routing takes care of physically assigning layers to each of the nets. The routing flow also takes care of the manufacturing rules like spacing, minimum area etc. For this it has to read the techfile provided by the foundry.

There are multiple steps in routing:

1. Global Route
   The goal of the global routing is to elaborate a routing plan in such a way that each

net is assigned to particular routing regions and also a given objective function, which is usually an estimate of a total wire length, is optimized.

2. Detail Route
   In detailed routing, each routing region is taken into account and each net is assigned a particular track within that region. It also performs hookup of the nets to the pins and fixes any routing violations.

First we need to load the route and signal integrity settings. We check for high fanout condition and possibly add buffer trees. Then we go for redundant via insertion and enable automatic redundant via insertion after each detail route change.

We do the initial routing using route_opt command. Then we perform derive_pg_connections to check whether all the power and grounds pins are properly connected. For further optimization we use route_opt effort as high and cross talk reduction as an option. Then run verify_zrt_route and verify_lvs commands.

After routing is completed, there may be some timing degradation. This will be caused by the real routes delay post route compared to estimated delay pre-route. To recover such timing degradation Incremental optimization is carried out.

3.2.6 Chip Finish (Base Fill) and Metal Fill

In this stage we insert the standard cell fillers for Device for Manufacturability. After routing optimization, filler cells are inserted to fill the "Gaps" in the design. This takes care of a continuous "n-well". Also critical area reduction can be done by spreading and widening the zrt wires. We can also insert diodes for antenna fixing. Then incremental routing optimization is performed again for register to register paths. Then the metal fill extraction options are set accordingly. Dummy metals (fills) are added to satisfy the metal density requirement provided by the foundry. Then signoff DRC checks are performed.

3.2.7 Outputs Stage

This is the last stage in physical implementation. In this stage, all the design collaterals are dumped out. Parasitic extraction is performed in form of SPEF. We also dump out the verilog netlist and SDC collaterals. Also the final layout is dumped out in GDS format. We also dump out both, setup and hold, timing violations. Also transition and capacitance violations are dumped along with setup and hold violations. Most of these collaterals are used in iterative procedure for the purpose of any further optimization.

3.3 Formal Equivalency Verification (FEV)

FEV Checks logic functionality independent of timing or actual hardware implementation. It finishes a lot faster than simulation. It can be used at different stages in a design flow to guarantee that functionality did not change across the flow. It is fastest way to verify a change/edit in netlist.

In FEV we formally verify actual design netlist with design which is extracted from RTL. We verify equivalence based on mathematical proof and not rely on simulation. We can also verify quick changes to a model.

Any mismatch is reported as "non-equivalents" (NEQs). The NEQs can be caused by:

- Comparing wrong RTL with netlist.
- Some changes done in the netlist which changes the functionality. E.g. buffer replaced by inverter.

Practical FEV is performed by giving the golden file and revised files separately. Here the golden file corresponds to the netlist extracted from the RTL design and revised file corresponds to the netlist extracted after synthesis or APR flow. Both the netlists are sourced from verification file by giving proper libraries, search files and mapping commands used in respective FEV tool.

3.4 Layout vs Schematic (LVS)

At the physical verification stage, Layout vs Schematic is one of the major checks. In this check we have to verify that the layout we have created is functionally the same as the schematic/netlist of the design. Also we check that while creating the design, we have correctly transferred the schematic into proper geometries. So there should be proper connections and we also have to ensure that no connection is missed.

By extracting the geometries from the layout, the LVS tool creates a layout netlist. After that the schematic netlist is compared with layout netlist. LVS result will be clean if these netlists match othrwise the tool will report the components causing the mismatch and also point their location. LVS check verifies the correctness of the design layout with respect to intended functionality of the design.

Below are some of the common LVS errors:

1. Opens: Tool will show opens if there are incomplete connections for certain nets.

2. Shorts: If the nets,which should not be connected, are overlaping with each othe then tool will point shorts at that location.

3. Unbound Pins: Unbound pin is reported if the pins don't have a geometry, but all the connection to the net are made.

4. Parameter Mismatch: LVS also checks for parameter mismatches. For example, the resistor is matched in both layout and schematic, but their value is different. Then a parameter mismatch is reported.

3.5 Design Rule Check (DRC)

To ensure manufacturability, there are certain layout rules enforced by the FAB which are coded in the techfile. Design Rule Checks determine if the layout rules are satisfied in the layout or not. Typical design rules inclue the spacing between metals, via rules, minimum metal width rules etc. There will also be specific rules pertaining to our technology. These rules are coded in the techfile and are taken care during placement and routing by ICC. There can be still some violations left after ICC because not all rules can be coded in the techfile. Specialized DRC analysis is done using the GDS from ICC to report all DRC violations. IC Validator is used for DRC analysis. The DRCs can be fixed either manually or through automated fixing process from ICC.

A 'design rule file', commonly known as runset by Synopsys ICV, is given as an input to the design rule tool. The design rules ensure sufficient margins to correctly define the geometries without any connectivity issues due to proximity in the semiconductor manufacturing processes, so as to ensure that most of the parts work correctly. For all mask layers, the minimum width rules exists. Also the spacing between these layers are specified. Depending on the width of one or both of the layers, these spacing rules may also alter. There can also be specific via density rules, rules between two different layers etc. The chip may not be functional if these design rules are violated.

3.6 Antenna Check

3.6.1 Antenna Violation

Process antenna effect (also known as "plasma prompted gate oxide harm") is an impact taking place during manufacturing i.e. this type of fault can happen at manufacturing stage explicitly. In order to avoid this issue from taking place, foundries typically supply antenna rules, which must be obeyed. If these rules are not obeyed then they will cause Antenna

Violations. As the charge accumulation on metals and discharge to a gate is through gate oxide, gate is at high risk and can also get damaged.

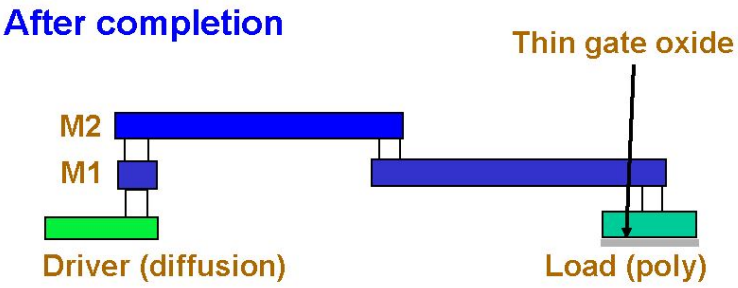

Figure 3.5: Illustration of Cause of Antenna Effect (a)

Figure 3.5 demonstrates a side perspective of normal net in an IC. Every net will incorporate no less than one driver, that should have a drain or source diffusion (implantation for modern technology), and no less than one receiver, that should be connected to a gate terminal over a thin gate oxide dielectric layer. As the gate dielectric is so thin, just a couple of molecules thick, it is likely to breakdown if a major stress is imposed. This can occur if the net, by one means or another, obtains a voltage to some degree higher than the ordinary working voltage of the chip.

This can't happen once the chip is fabricated, as each net has in any event some drain/source implant associated with it. A diode is framed by the drain/source implant. Breakdown of this diode takes place at a lower voltage than the oxide (either reverse breakdown or forward diode conduction), and does as such non-dangerously. This ensures the safety of gate oxide.

During the manufacturing process, metal layers are assembled layer by layer. i.e. metal1 is saved to begin with, then with plasma etching all undesirable segments are etched away. The metal geometries can gather charge from it when they are presented to plasma . When metal1 is finished, next step is to fabricate via1, then metal2 etc. So with every passing stage, static electricity appeared by glimmer in figure 3.6 gets developed on the metal geometries.

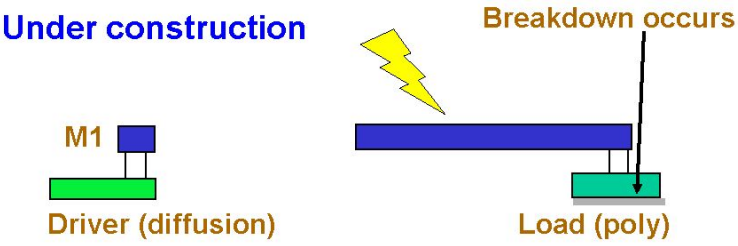

Figure 3.6: Illustration of Cause of Antenna Effect (b)

The bigger the metal territory that is presented to the plasma, the more charge they can gather. In the event that the charge gathered by the metal is large enough to make current stream to the gate, the gate oxide can get damaged. This occurs on the grounds that since the layers are assembled one-by-one, a drain/source implant may not be accessible at that time for discharge.

3.6.2 Antenna Rule

The Antenna ratio is characterized as the ratio between the physical region of the conductors making up the antenna to the aggregate gate oxide region to which the antenna is electrically associated. A most extreme allowable antenna ratio is set by every foundry for its procedures. The physical verification tool signals an error if the addition of the ratios of all lower layer interconnects, in addition to the layer under check, is more than the suitable range. For instance, suppose maximum reasonable antenna ratio for metal1 is 300. If the metal area interfacing with the gate is 400 sq.um and the area of gate is 1 sq.um then there will be an antenna violation.

3.6.3 Fixes for Antenna Violations

All in all, the router must be able to fix these antenna violations. Following are conceivable fixes to settle these violations:

- Change in the order of the routing layers. Instead of connecting the gate to the lower layer if we interface it with the most elevated metal layer, antenna violation will not happen regularly. This arrangement is appeared in figure 3.7.

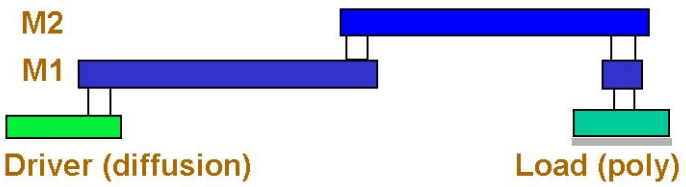

Figure 3.7: Antenna Violation Fix by Routing

- As appeared in figure 3.8, include diode to the net. A diode can be framed far from a drain/source of a MOSFET, for instance, with a p+ implant in a n-well or with a n+ implant in a p-substrate. The gate oxide can be protected if the diode is connected to the metal close to the gate. This should be possible just on nets with violation or on each gate. The “each cell” arrangement can settle all antenna issues with no requirement for activity by whatever other tools.

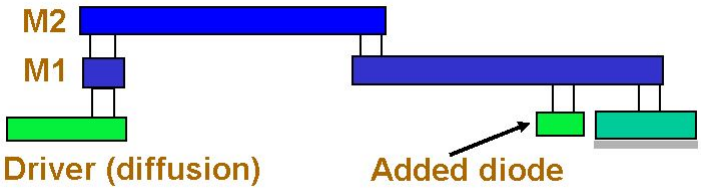

Figure 3.8: Antenna Violation Fix by Adding a Diode

3.7 Power Estimation

Power consumption is one of the major concerns for designers nowadays. Design needs a proper balance of many complex issues. Timing stays critical, but power has become important toward achieving design closure. Power affects requirements of cooling, performance of design, battery life of device, reliability of chip etc. That is the reason more accurate power analysis is required. So it is essential for design architects to consider the effect of their design techniques on power in confluence with area and timing. An inability to estimate power can prompt chip failure.

At 90 nm and below, signal integrity effects, power and timing are related to each other. An accurate timing engine is required to perform accurate timing and slew calculations to get the precise power analysis. Since timing parameters affect power dissipation, designers require a solution that takes advantage of these interdependencies [7]. To take advantage of these interdependencies, we need to carry out power estimation in single unified environment. For that we use PrimeTime PX tool. There are diverse components which add to Dynamic and Static Power. For establishing the flow, we have to take them into account to estimate the power consumption of our design. These are represented in figure 3.9.
The things which are required to accurately estimate the power consumption of a design are listed below:

- Netlist Data
  The netlist of the design is needed to acquire the connectivity information of the design as well as cell types which are used in the design. In addition to that it is also required to precisely compute the capacitance on the drivers.

- Design Constraints
  Design constraints are required to provide the information about the different restrictions applied to different I/O pins and paths so as to get accurate power information.

- Power Models
  Vendors of the library have to provide model of the cells that has the information of

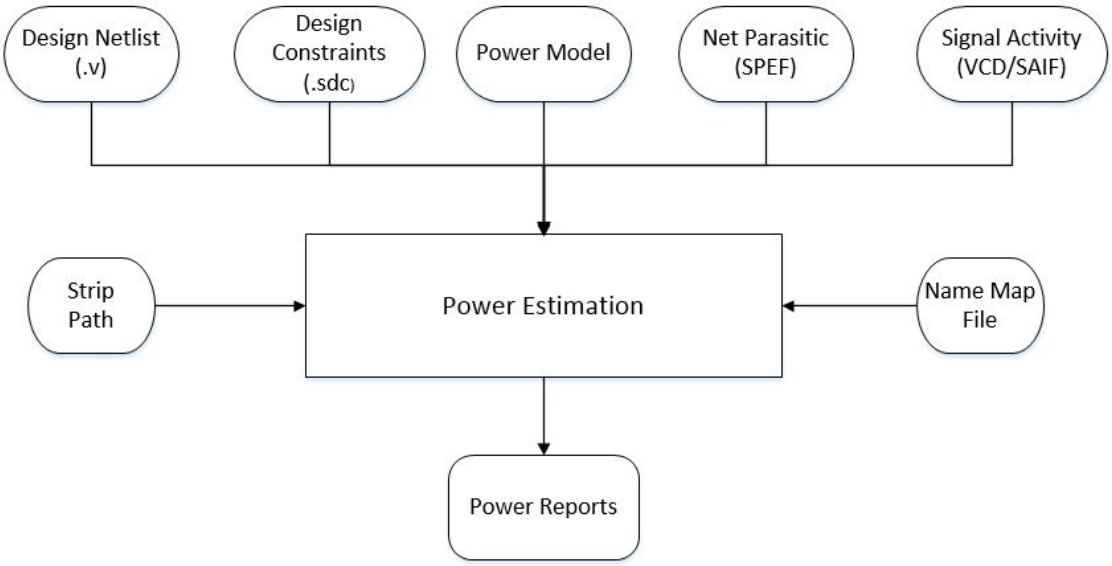

Figure 3.9: Power Estimation Requirements and Flow

both the static and dynamic power consumption internal to the cell for computing the internal power of the CMOS cells.

- Parasitic Information (SPEF)
  Parasitic information of the nets are provided as the transition time is extracted from that. Dynamic power is affected by parasitics of these nets and is proportional to the net capacitance. It also depends on the signal transition time and output load. To have more clear vision, see the figure 3.10.

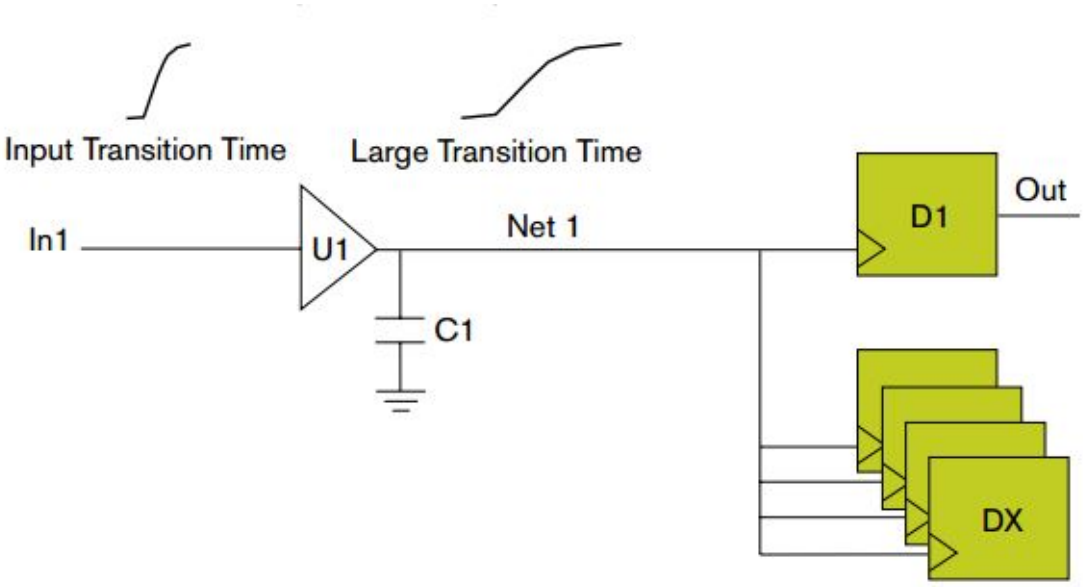

Figure 3.10: Effect of Transition Time on Dynamic Power

- Signal Activity
  The signal activity contains the information about the switching activity of of the nets based on event-based simulation or gate-level simulation. Both, static and dynamic, power are having influence of signal activity. The static power depends on the state of the logic and the dynamic power is proportional to the activity factor. Signal activity can be provided in two forms.

  1. VCD (Value Change Dump)
     For peak power analysis, the tool must be provided a timing logic simulation that dumps VCD file as an output which has all the information about the signal transition trace of each net. It comprises of all the important factors that can have impact on power consumption and hence is more accurate form.
  2. SAIF (Switching Activity Interchange Format)
     For average power analysis, the gate level toggle rate are provided in absence of event based rate. This is generated in SAIF format which has different information like time the signal stays at particular state and number of times the signal toggles per net. This will allow to estimate dynamic and leakage power.

- Strip Path
  VCD file is dumped at the top level using RTL. That will have the information for all the sub blocks in the design. Strip path is provided to point to the sub-block as most of the time the analysis is carried out at the block level.

- Name Map File
  Map file is also provided in order to enable the tool to properly map the correct registers as after synthesis, the RTL register names will be changed.

3.8 Reliability Verification

Our designs must perform as desired but also for a reasonable span of time. It is good to make the world's fastest processor, but what if it fails to operate after3-4 weeks! Typically semiconductor designs should work upto 7-10 years. That is why reliability of the design is as important as the performance. As the scaling is progressing we are having millions and millions of transistors on a single die. So there is a high chances that few transistors may fail. This will cause entire design to malfunction. So the reliability should keep up with design complexity.

In semiconductor industry, ICs should work in almost every environmental conditions which may cause the change in current of the device or change in temperature. Regular use of

device will wear it out after some time period. Electromigration is one of the reasons behind such wearout.

3.8.1 Electromigration

Electromigration corresponds to the unwanted mass transport of materials in a semiconductor. Interconnect wires consist of the metal ions which construct the lattice structure of the metal used for interconnects. Under the influence of high current density, the conducting electrons can transfer their momentum to diffusing metal ions. The metal ions will get drifted in the direction of the electron flow. As an outcome of this metal ions displacement, opens or shorts can be formed as shown in the figure 3.12 which can cause the wrong functionality or can lead to total circuit failure in extreme cases.

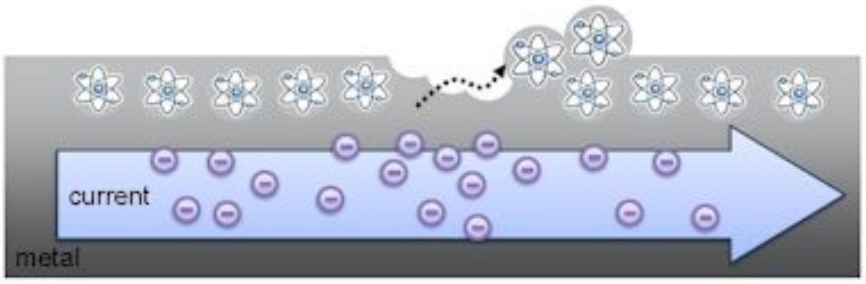

Figure 3.11: Electromigration in a Conductor

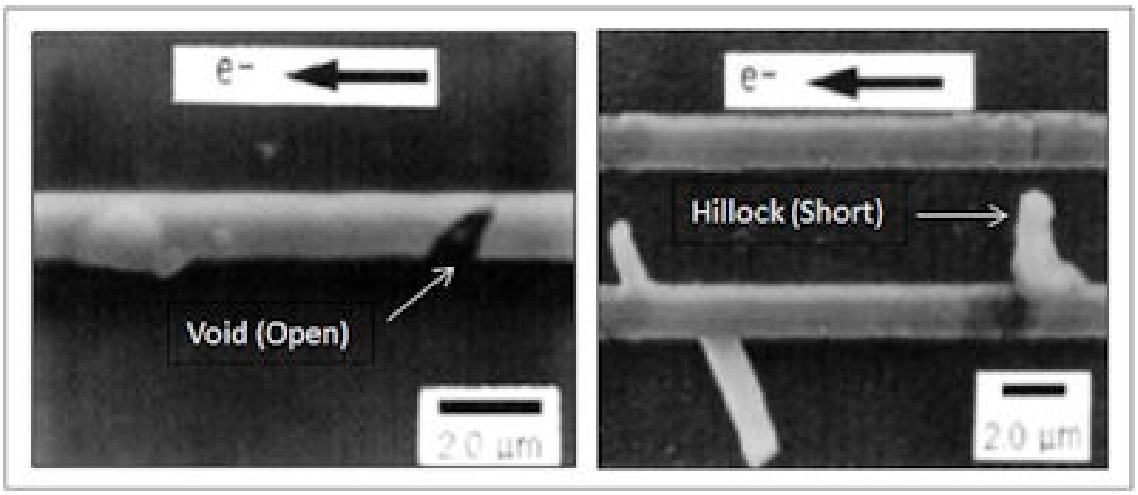

Figure 3.12: EM Failures - Void (Open) and Hillock (Short)

Electromigration gets worse as current density rises. Also the transfer of momentum gets more severe with increase in temperature. Beyond 28 nm, designs are having thinner interconnects leading to higher current density and also interconnects are changing at very high speed in GHz range. All such parameter contributes in electromigration.

3.8.2 Reliability Check

Reliability check verifies the ability of a design to perform the specific function under specified conditions for a specific period of time. Tool used for checking reliability is Ansys Totem. For this analysis we need to provide different data to the Reliability tool as shown in the figure 3.13.

Total analysis is done in two phase.

1. Effective Current Calculation:
   In this phase peak, average and RMS (Root Mean Square) current values are calculated and based on that total electromigration effect on a metal interconnect is evaluated. At the driver cell current values are high and decrease slowly as distance from the driver increases. Tool reads in the signal net routing and geometry information along with signal net parasitic data to determine the current values in each net, from its driver to its receiver. For every net and via in the design, these current values are calculated by the tool.

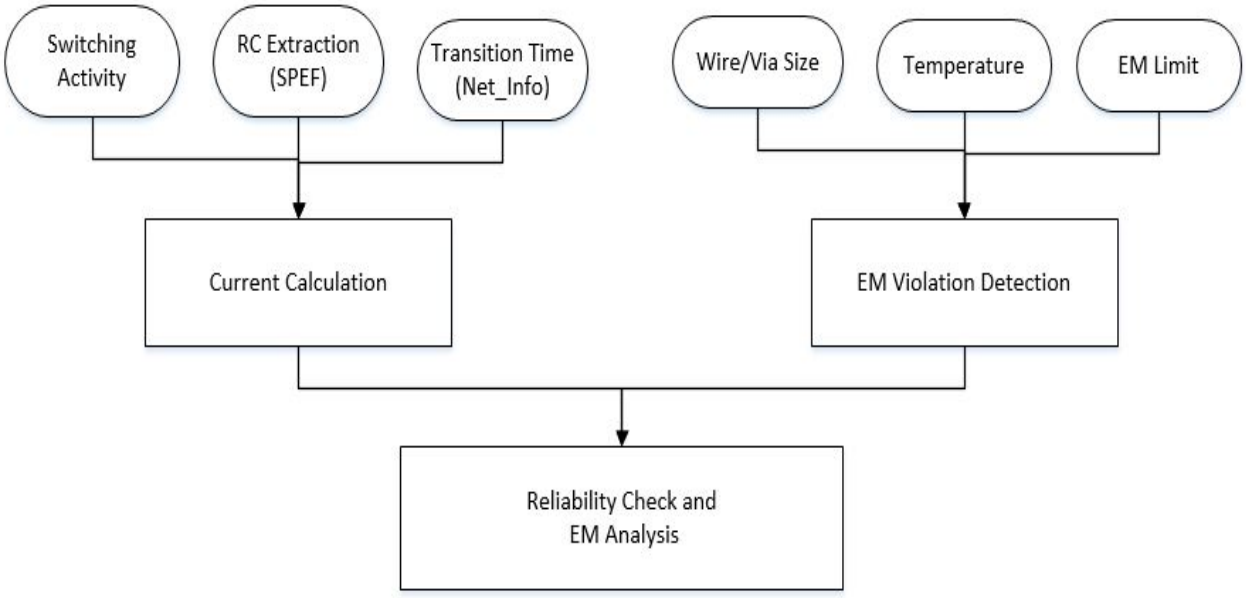

Figure 3.13: Components Required for Reliability Check and Electromigration

2. EM Violation Detection:
   After determining these current values, the current density is estimated at each wire segment and via and compared to the relevant EM limits. The EM limits can be specified in the foundry provided technology file as dependent on physical parameters such as the width of the wire, size of via, and temperature of the die.

3.8.3 Fixes for Reliability Violations

The reliability check gives the number of violations and worst violation. It returns the total list of signal as well as power nets and vias on which any of the three current component is exceeding the maximum limit value. Also their location is given in generated reports. The basic fix is to increase the current carrying capacity of the affected net or via. This is done by following two methods [8].

- Widening the wire: By increasing the wire width we can increase the allowable current limit to resolve the electromigration issue because increasing the wire width will reduce the resistance of that wire. Similarly we can increase the via size if the violation is on via.

- Sizing of cell: Use of the weaker driver cell will cause the minimal transition which in turn will reduce the current density and bring down the electromigration impact to a manageable level. But it will also cause the timing change due to the weaker drive strength so this approach is suitable only if we have adequate timing margins for that net.

3.9 Engineering Change Order (ECO)

Methodologies for synthesis and APR from industry leading organizations depends heavily on ECO flow to accommodate the timing closure at the last stage. An engineering change order (ECO) is an incremental change made to a complete or nearly complete design [4]. ECOs can be used to fix different types of violations like timing violation, functional errors, disturbance due to noise etc. without doing the synthesis, placement and routing on the total design. If any changes arrives in the design at the very last moments then also ECO can be used to incorporate them without altering the performance of the design.

Some redundant standard cells, commonly known as spare cells, are placed in the core area of the design during the placement stage [9]. When ECO changes arrive, instead of developing an entire new layout, ECO flow utilizes already existing spare cells so that it will have minimal effect on original chip layout. Traditionally, the ratio of spare cells to total number of cells is about 5% to 10%. These spare cells are quite useful when there is a need of a signoff timing closure or when we need to accommodate above mentioned incremental changes [9]. This utility comes at cost of extra area and leakage overhead but we need to fancy our chances against the design cycle time.

ECO can be categorized in two types.

1. **Unconstrained ECO**: It gives us liberty to insert new cells, remove cells and modify the position of already existing cells. This type of ECO preferred when our design is not taped out.
2. **Freeze Silicon ECO**: As it's name suggests, we can only modify the via mask patterns and metal keeping cell placement intact. It is implemented when we have taped out our design.

The typical Unconstrained ECO flow is shown in the figure 3.14 below. After the routing update we need to again check for the LVS. If any violation found then again we have to take the necessary steps to get it clean and for that we may need to go to a stage where the issue can be resolved. So this may not need to be a single pass flow.

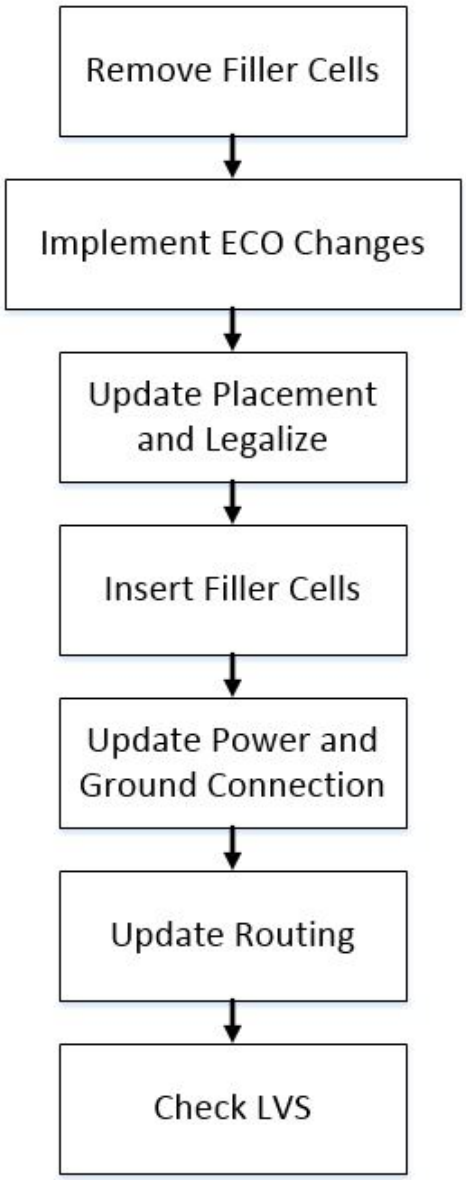

Figure 3.14: Typical ECO Flow

3.9.1 ECO Flow Optimization

There are various stages in APR flow. Figure 3.15 shows the APR stages which are affected when ECO is implemented. We have final APR database ready, but whenever ECO arrives we have to go back to the previous stages as mentioned in the figure 3.15 all the way back to route_opt database. Then after implementing the ECO, again we have to do base fill and metal fill which can cause the additional violations in the layout. Most of the violations stays same prior to ECO because they are introduced each time when we do metal fill. And then we have to check for various verification flows for the design which is very time consuming as we won't get all the violations in a single pass.

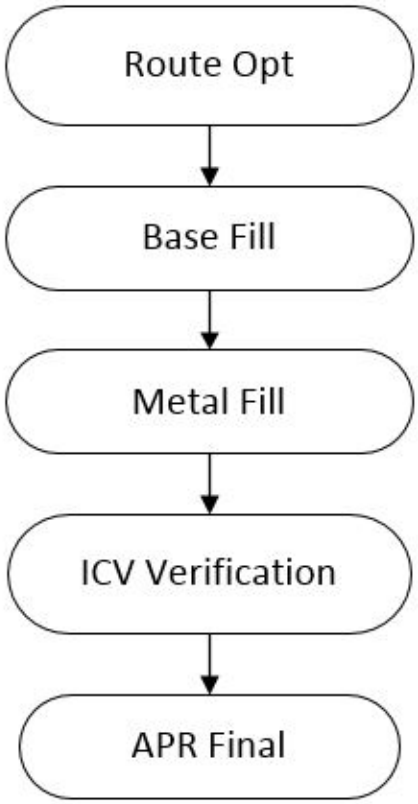

Figure 3.15: APR Stages Affected by ECO

The approach that we used here is we already have taken a backup of most common layout violations prior to ECO implementation and we have corrected those violations in route_opt database only. Now when the next ECO comes, we can directly implement it in route_opt database. By doing this we don't have to fix the same old violations which occurs after every metal fill. We will be left with only some handful of new violations that may be caused because of ECO. This will significantly reduce the ECO cycle.

Chapter 4

Results and Observations

The results of one of the digital blocks are shown here. Results show the comparison in terms of timing between its initial state and latest state. Different violations for different physical implementation stages are compared.

Figure 4.1 shows the comparison of setup violations between different stages of physical implementation after block level optimization. It shows setup Worst Negative Slack (WNS), Total Negative Slack (TNS) and number of violating paths. For this particular block there is no change in the number of violating paths as all these violations are at the interblock paths. So these are taken care when all the blocks are integrated at the top level.

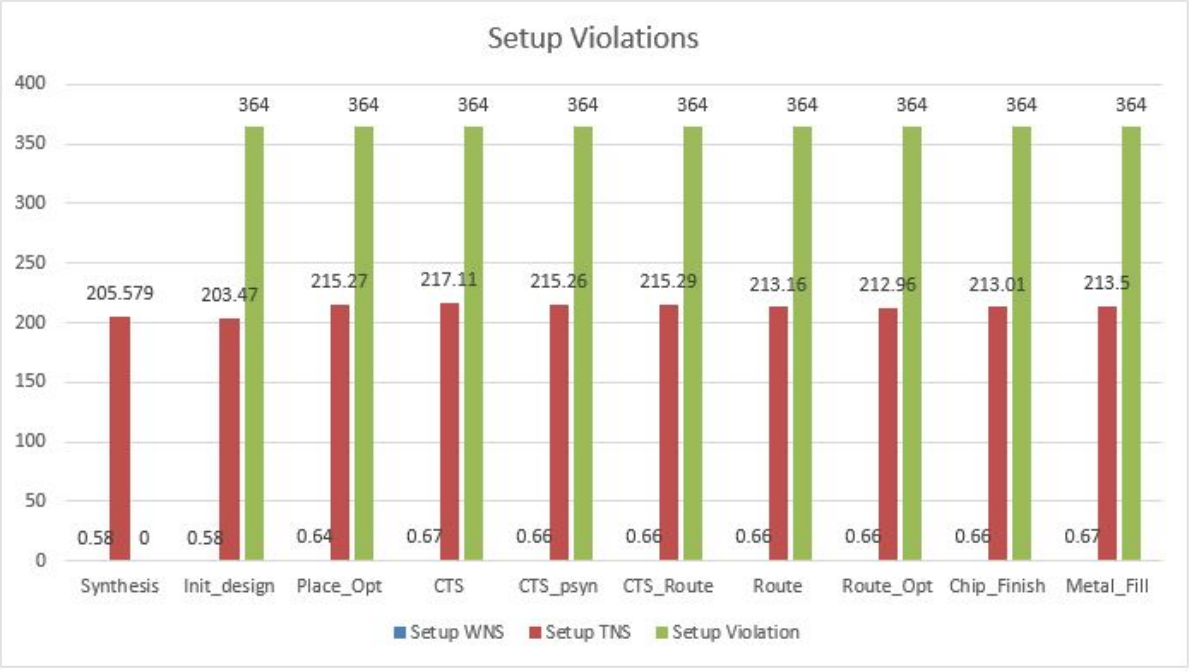

Figure 4.1: Setup WNS, TNS and Number of Violating Paths for Different Stages

Figure 4.2 shows the comparison of hold violations between different stages of physical implementation after block level optimization. It shows hold WNS, TNS and number of violating paths. As we can see, Hold violations vanishes after Clock Tree Synthesis because of the insertion of CTS buffers as a part of CTS build. Result shows only upto the CTS stages as after that also there is no hold violation at block level.

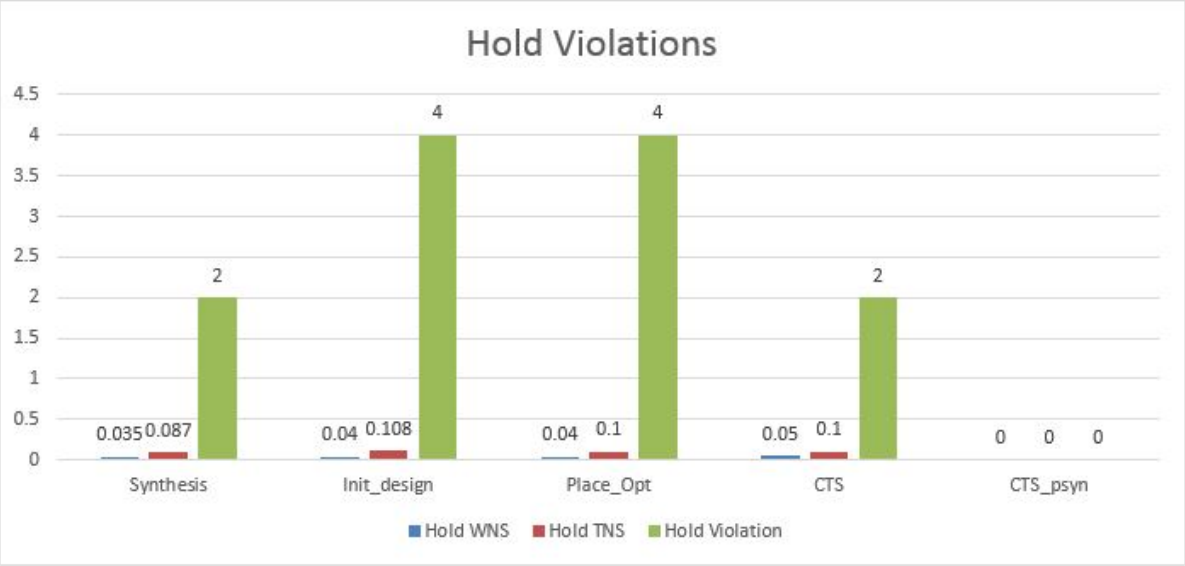

Figure 4.2: Hold WNS, TNS and Number of Violating Paths for Different Stages

Figure 4.3 shows the comparison of standard cell utilization between different stages of physical implementation after optimization at block level. A standard cell utilization is a

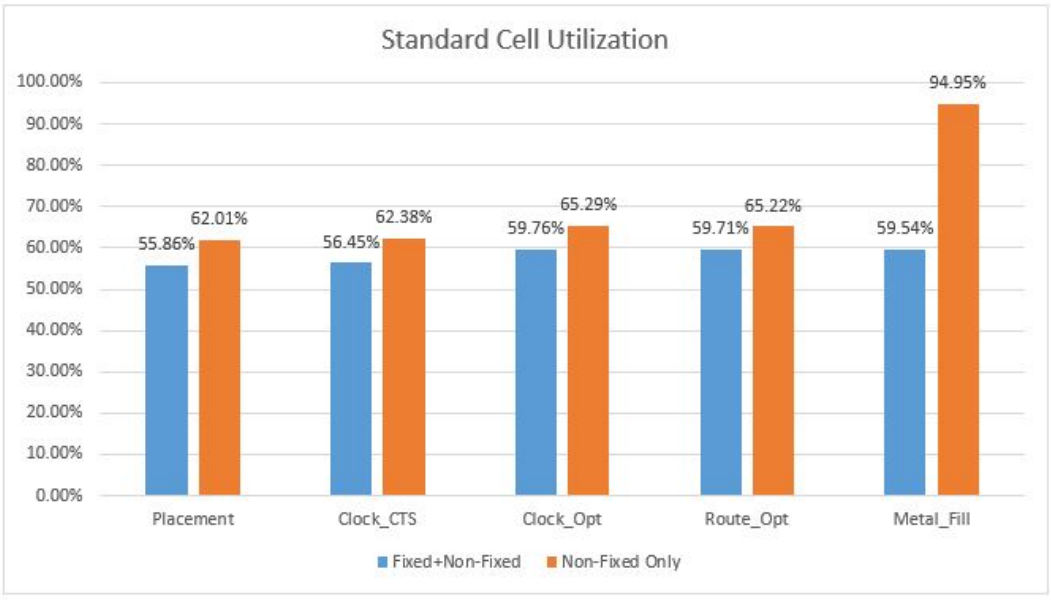

Figure 4.3: Comparison of Utilization for Different Stages

percentage of total area occupied by the cells from the total available area. For example, 60% utilization means 60% area is occupied by the placement of the standard cells and 40% area is still left. This remaining area can be used for placement of additional cells, during legalization of cells and to prevent routing congestion.

Utilization report is generated for "Fixed + Non-Fixed" cells and "Non-Fixed Only" cells. Figure 4.4 will give the basic idea about Fixed and Non-fixed cell utilization. The utilization for figure 4.4 is also calculated.

Figure 4.4: Fixed and Non-Fixed Cells

Default utilization (fixed and unfixed standard cells): 3/4 = 75%
Non-fixed-only standard cell utilization: 1/2 = 50%

Figure 4.5 compares the different violations before and after modifying the constraints.

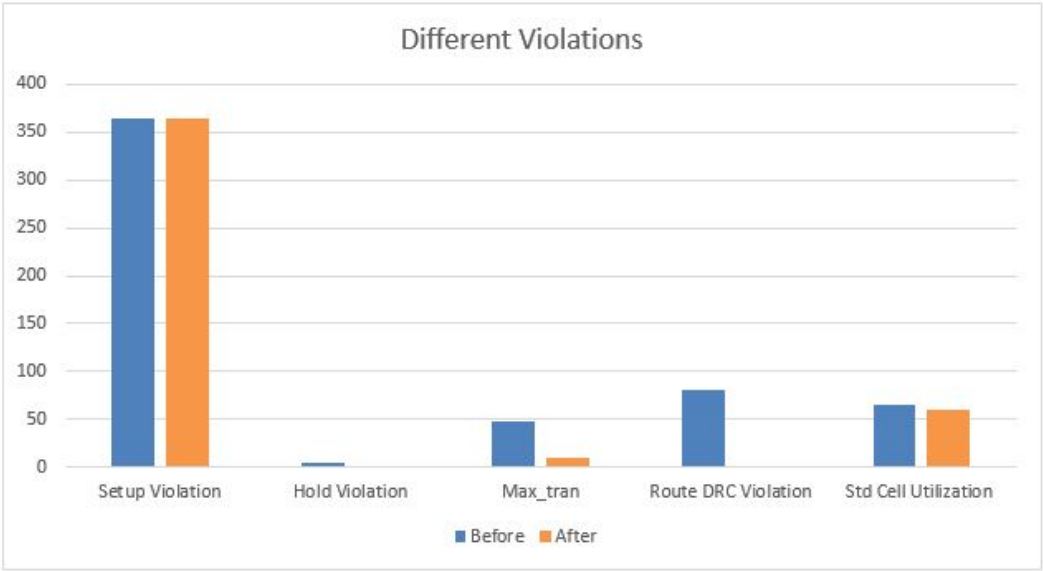

Figure 4.5: Comparison of Different Types of Violations

Setup violations are at the boundary so there is not any change in the count. There are no hold violations. Maximum transition violations are also reduced by the tool as we have

changed the transition constrains and remaining transition violations are taken care after some ECO iterations. Also standard cell utilization is reduced after the RTL team removed the redundant logic for this particular block.

Figure 4.6 shows the progress in cleaning up the reliability violations for the block. As this is not a single pass flow it took quite a few iterations to get complete RV clean result.

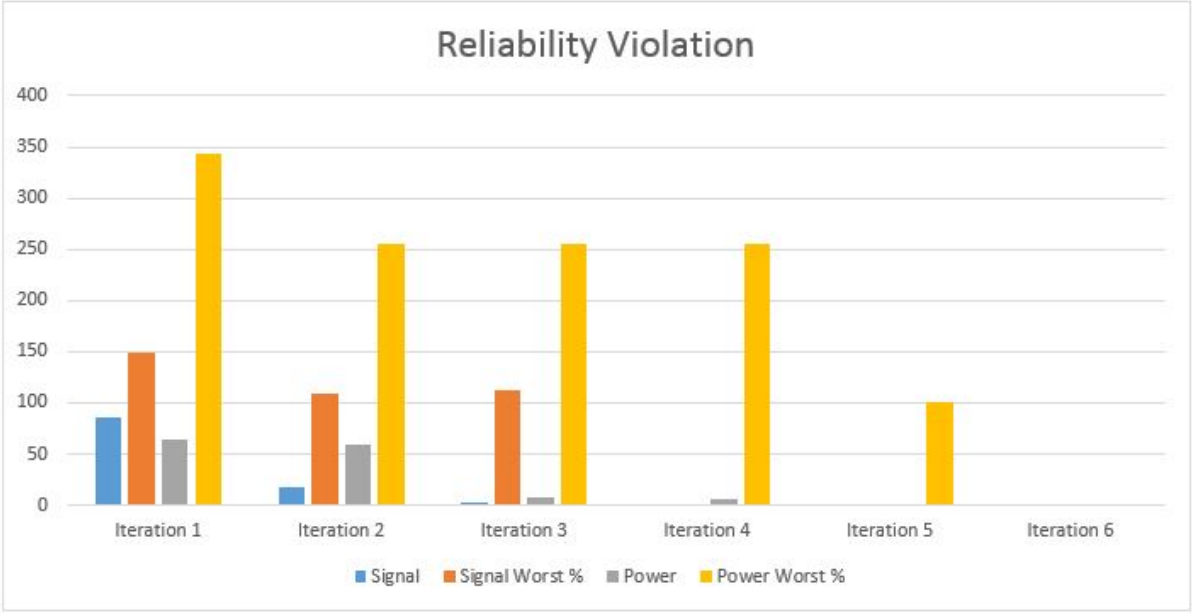

Figure 4.6: Reliability Violation Cleanup Progress

4.1 Challenges Faced

- After the APR flow, there were some physical only cells which were placed in the keepout region. This should not happen as the keepout regions are the regions around the boundary of fixed macros in our design in which no other cells are placed. So these physical only cells were causing opens during LVS in the design. To get rid of these opens we need to remove these cells manually. This was a fault in the APR flow and later the script was modified to resolve this issue.

- There were some nets in the block which were causing the shorts at the top level. These shorts were not observed at the block level. We need to add the routing blockages in the area creating these shorts. These routing blockages will not allow the routing through those areas and shorts were resolved at the top level.

- In the post APR netlist, there were some assign statements. As we know that the assign statements means zero delay which is not practical after the APR. So after

getting these ports with the assign statements, we determined their connections. Then tie low or tie high cells were added to those ports based on their connections. After that in the second pass, buffers were added for the remaining assign statements.

- There were many maximum transition violation in my block at the top level but at the block level no violations were found. Finally we found that there were no CTS cells in the design due to a missing library. So we had to update the library path and run the fresh APR flow. Then also CTS cells were not added in the design. By digging deep in the scripts we found that the flow was routing the design before the CTS stage and that was preventing the CTS cells from being placed in the design. After changing the script, got the desired results.

- There were non clock cells used in the clock path at routed database, we are supposed to use special clock buffers in clock paths which has equal rise and fall delays to maintain 50% duty cycle. We had to find all the clock cells which had been using non clock buffers and the cells were immediately swapped to respective cells.

- There were minimum area violations in the blocks as part of DRC run results. A script was written to add a dummy metal layer from the port terminals to meet the area requirements.

- There were some inout ports in the RTL. While dumping the RV collaterals, some of the nets connected to these inout ports were recognized to have two drivers i.e. one is the output of the cell and other is inout port. This caused the issue in the transition calculation for the tool. So we had to change the direction of these ports in the netlist. Consequently, FEV was failing as these changes were not reflected in the RTL, so worked with RTL team to get this issue resolved.

- As a part of RV fixes, we needed to increase the width of the violating nets and also increase the size of the vias. Increasing via array size was causing additional LVS issue in the design. We had to play with the orientation of these vias in such a way that they won't cause the LVS issue and RV also gets fixed.

Chapter 5

Conclusion

As the technology is progressing with Moore's law, device speed is getting faster which introduces critical challenges in implementing designs physically. Data communications blocks have become very critical in today's fast growing VLSI field. Most of the design in the modern days are mixed signal designs so it is very important to know the signal transfer between analog and digital blocks. Also such designs have different power domain and it is critical that all the cells are hooked to proper power domain. By properly understanding the functionality of such design, we can manipulate the placement of standard cells and routing of the signal and clock nets within our design in such a way that all the design requirements are adequately met without any major changes in the design functionality or violating design performance. All the different types of violations faced during this high speed interface implementations were resolved and fixed at the different hierarchy of the design, some at the block level and some at the top level.

As we move towards faster speed of the interface, these violations gets much worse and problem of crosstalk increases. The congestion problem has also some effects on solving these violations as less space will be available to accommodate the extra buffers to fix these violations. These will point some challenges in implementing these interfaces physically.

References

[1] A. Kahng, J. Lienig, I. Markov, J. Hu, *"VLSI Physical Design: From Graph Partitioning to Timing Closure"*, Springer (2011), ISBN 978-90-481-9590-9

[2] Mehrotra, Alok, Van Ginneken, Lukas P P P, Trivedi, Yatin, *"Design flow and methodology for 50M gate ASIC"*, IEEE Conference Publications, ISBN 0-7803-7659-5

[3] Synopsys Inc. *Design Compiler® User Guide, Version K-2015.06, June 2015.*

[4] Synopsys Inc. *IC Compiler™ Implementation User Guide, Version K-2015.06-SP4, December 2015.*

[5] Richard Solomon, Synopsys, *PCI Express® Basics & Background* [Online]. Available: `https://pcisig.com/sites/default/files/files/PCI_Express_Basics_Background.pdf`

[6] Sadiq M. Sait, Habib Youssef, *"VLSI Physical Design Automation: Theory and Practice"*, World Scientific Publication.

[7] Gordon Yip, Synopsys, *"Expanding the Synopsys PrimeTime® Solution with Power Analysis"*, White Paper, June 2006

[8] Geetha Rangarajan – Synopsys, James Deng – Altera *"Addressing signal electromigration (EM) in today's complex digital designs"*, White Paper, January 28, 2013

[9] Geetha Rangarajan – Synopsys, James Deng – Altera *"New spare cell design for IR drop minimization in Engineering Change Order"*, DAC '09 Proceedings of the 46th Annual Design Automation Conference, ISBN 978-1-60558-497-3, July 27, 2009