Optimization In Timing Analysis Corners In 28nm

By

Chitrarth Patel

Roll No: 10MECV26

Department of Electronics And Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382481

May 2012

Optimization In Timing Analysis Corners In 28nm

Major Project Report

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology

In

Electronics And Communication Engineering

(VLSI Design)

By

Chitrarth Patel

(10MECV26)

Under the Guidance of

Mr. Sandeep Asija

STMicroelectronics India Pvt. Ltd.

Department of Electronics And Communication Engineering

Institute of Technology

Nirma University

Ahmedabad-382481

May 2012

Declaration

This is to certify that

i) 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.

ii) Due acknowledgement has been made in the text to all other material used.

Chitrarth Patel

Certificate

This is to certify that the Major Project entitled **"Optimization In Timing Analysis Corners In 28nm"** submitted by **Chitrarth Patel (10MECV26)**, 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.

Date: ____________________ | **Place: Ahmedabad**

External Guide | **HOD**

____________________ | ____________________

Mr. Sandeep Asija(Manager)
STMicroelectronics India Pvt. Ltd.,

Prof. A. S. Ranade
Professor, EC

Director | **Internal Guide**

____________________ | ____________________

Dr. K. Kotecha
Director, IT, NU,

Prof.N.P.Gajjar
Nirma University

Acknowledgements

Apart from the efforts of me, the success of this project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to those people who have been instrumental in the successful completion of this project. I would like to thank my Project manager **Mr.Sandeep Asija**, **Senior manager**, Chip Implementation Team, STMicroelectronics, Bangalore, for giving me this project and for his guidance at each and every stage of the project even in his busy schedule. I would like to thank Chip Implementation Team members at STMicroelectronics **Karthik Venkataraman**, **Ms. Sumithra**, **Ms. Gayatri**, **Mr. Ganesh Prabhu**, **Kathir Ramakrishnan** and **Swaroop A** for their excellent direction and guidance for my project work. In addition, a special thanks to Management of STMicroelectronics, Bangalore for providing me with all the required facilities.

I would like to acknowledge and thank **Dr. K. R. Kotecha**, the director of NIRMA University and **Dr. Ashok Ranade**, HOD, E and C Dept. for providing me with an opportunity to get an internship at ST Microelectronics, Bangalore. I would like to thank my internal guide **Prof.N.P.Gajjar** for guiding, constant encouragement, support and suggestions for improvement during the course of my project. His encouraging words brought out the best in me.

- Chitrarth Patel

10MECV26

Abstract

Static Timing Analysis (also referred as STA) is one of the many techniques available to verify the timing of a digital design. An alternate approach used to verify the timing is the timing simulation which can verify the functionality as well as the timing of the design. The term timing analysis is used to refer to either of these two methods - static timing analysis, or the timing simulation. Thus, timing analysis simply refers to the analysis of the design for timing issues.

The STA is static since the analysis of the design is carried out statically and does not depend upon the data values being applied at the input pins. This is in contrast to simulation based timing analysis where a stimulus is applied on input signals, resulting behavior is observed and verified, then time is advanced with new input stimulus applied, and the new behavior is observed and verified and so on. The purpose of static timing analysis is to validate if the design can operate at the rated speed. To simulate and verify all timing conditions of a design with 10-100 million gates is very slow and the timing cannot be verified completely. Thus, it is very difficult to do exhaustive verification through simulation.

In this project Timing Analysis is done to identify and diagnose the violation in complex design, learn to perform synthesis for the giver RTL using synopsys tool DC(DESIGN COMPILER), STA using synopsys tool PT(PRIMETIME) tool, which Setup and hold violations in complex design. Formal verification is done by the synopsys tool called formality.

Contents

Declaration iii

Certificate iv

Acknowledgements v

Abstract vi

List of Figures ix

1 Introduction 1

2 Background Theory 3

2.1 Background Theory . . . 3
2.1.1 Ad-hoc Method . . . 3
2.1.2 Structured Method . . . 4
2.2 Fault classification . . . 8
2.3 Traditional ASIC design flow . . . 9
2.4 Synthesis . . . 12
2.5 Formal Verification . . . 16
2.6 Timing Analysis . . . 18
2.6.1 Types of Checking performed . . . 18
2.6.2 Timing Paths . . . 18
2.6.3 Delay Calculation . . . 20
2.6.4 Constraints Checking . . . 20
2.6.5 Timing Exceptions . . . 22
2.7 Setup and Hold Checking for Latches . . . 22
2.7.1 Timing Exceptions . . . 25
2.8 Setup and Hold Checks . . . 26
2.8.1 Path Delay Tracing for Setup and Hold Checks . . . 26
2.8.2 Setup Timing Check for Worst-Case Conditions . . . 27
2.8.3 Hold Timing Check for Best-Case Conditions . . . 27
2.8.4 Simultaneous Best-Case/Worst-Case Conditions . . . 28
2.9 Clock Reconvergence Pessimism Removal . . . 29

2.9.1 Reconvergent Logic Example 30
2.9.2 Minimum Pulse Width Checking Example 30
2.10 DFT(scan) Insertion 32
2.11 DFT compression (codec insertion) 34
2.11.1 Scan Compression working 35
2.11.2 Hierarchical Adaptive Scan Synthesis (HASS) 36
2.12 Defining the Operating Conditions 37

3 Methodology 39
3.1 To convert RTL into Gate level netlist 39
3.2 To convert RTL into Gate level netlist 40
3.3 DFT (scan) insertion and stitching 41
3.4 Parasitic Interconnect Corners 42

4 Result Analysis 44

References 50

List of Figures

2.1 Ad-hoc method - Test point insertion 4
2.2 MUX based Scan cell 5
2.3 Clocked scan cell 6
2.4 Level sensitive scan design 7
2.5 Boundary Scan 8
2.6 Basic Synthesis flow 14
2.7 Formal verification 17
2.8 Timing paths 20
2.9 Setup and Hold checks 22
2.10 Latch-Based Paths 23
2.11 Time Borrowing in Latch-Based Paths 24
2.12 Hold Checks in Latch-Based Paths 25
2.13 Design Example 26
2.14 Setup Check Using Worst-Case Conditions 27
2.15 Hold Check Using Best-Case Conditions 28
2.16 Clock Reconvergence Pessimism Example 29
2.17 Reconvergent Logic in a Clock Network 30
2.18 Scan shift 32
2.19 Scan capture 33
2.20 DFT insertion flow chart 34
2.21 Compressor and decompressor logic added to scan chains 35
2.22 Adaptive scan compression 36
2.23 Hierarchical Adaptive Scan Synthesis Architecture 37

4.1 Worst Negative Slack(WNS)/Critical Path Path for Setup and Hold . 44
4.2 worst path/critical path for the setup 45
4.3 worst path/critical path for the hold 46
4.4 worst path/critical path for the setup and hold for CMAX-0C corner 47
4.5 worst path/critical path for the setup and hold for CMAX-125C corner 47

Chapter 1

Introduction

Static Timing Analysis (also referred as STA) is one of the many techniques available to verify the timing of a digital design. An alternate approach used to verify the timing is the timing simulation which can verify the functionality as well as the timing of the design. The term timing analysis is used to refer to either of these two methods - static timing analysis, or the timing simulation. Thus, timing analysis simply refers to the analysis of the design for timing issues. The STA is static since the analysis of the design is carried out statically and does not depend upon the data values being applied at the input pins. This is in contrast to simulation based timing analysis where a stimulus is applied on input signals, resulting behavior is observed and verified, then time is advanced with new input stimulus applied, and the new behavior is observed and verified and so on.

Given a design along with a set of input clock definitions and the definition of the external environment of the design, the purpose of static timing analysis is to validate if the design can operate at the rated speed. That is, the design can operate safely at the specified frequency of the clocks without any timing violations. The more important aspect of static timing analysis is that the entire design is analyzed once and the required timing checks are performed for all possible paths and scenarios of the design. Thus, STA is a complete and exhaustive method for verifying the timing of a design. Static timing analysis is a complete and exhaustive verification of all timing

checks of a design. Other timing analysis methods such as simulation can only verify the portions of the design that get exercised by stimulus. Verification through timing simulation is only as exhaustive as the test vectors used. To simulate and verify all timing conditions of a design with 10-100 million gates is very slow and the timing cannot be verified completely. Thus, it is very difficult to do exhaustive verification through simulation.

In this project Timing Analysis is done to identify and diagnose the violation in complex design, learn to perform synthesis for the giver RTL using synopsys tool DC(DESIGN COMPILER), STA using synopsys tool PT(PRIMETIME) tool, which Setup and hold violations in complex design. Formal verification is done by the synopsys tool called formality.

The objective of this project is to implement the DFT technique for Design Under Test (DUT) to increase the test coverage without increasing the test cost during testing of chip at gate level. This DFT technique includes scan insertion and compression for DUT. Further Static Timing Analysis (STA) is done for DFT inserted DUT to fix all timing violations. Formal verification is done between two versions of design (RTL-RTL, RTL-Netlist, Netlist-Netlist) for logical equivalence check. After the DFT check is done for DUT, we will get a design which is 100

Chapter 2

Background Theory

This chapter includes the Background theory of DFT which explains different types of DFT techniques in detail. Next contains different kinds of faults that may present in DUT. Next explains the traditional ASIC chip design flow and theory related to the methodology used in this project. Finally it concludes with some literature reviews.

2.1 Background Theory

DFT is an extra design effort implemented for Design Under Test (DUT) to locate and diagnose the faults present in DUT by increasing controllability and observability of the internal nodes of design without affecting the test cost. DFT refers to the design techniques that make the task of subsequent testing easier. There is definitely no single methodology that solves all embedded system-testing problems. There also is no single DFT technique, which is effective for all kinds of circuits. DFT techniques can classified into two categories

2.1.1 Ad-hoc Method

In this method large designs are partioned to small design to reduce the test cost and test points are added manually to the designs to increase testability and observability.

The controllable points (cp) are active points and observable points (op) are passive ones. Fig.2.1 shows controllability points (CP) and observability points (OP) which are added manually in ad-hoc method.

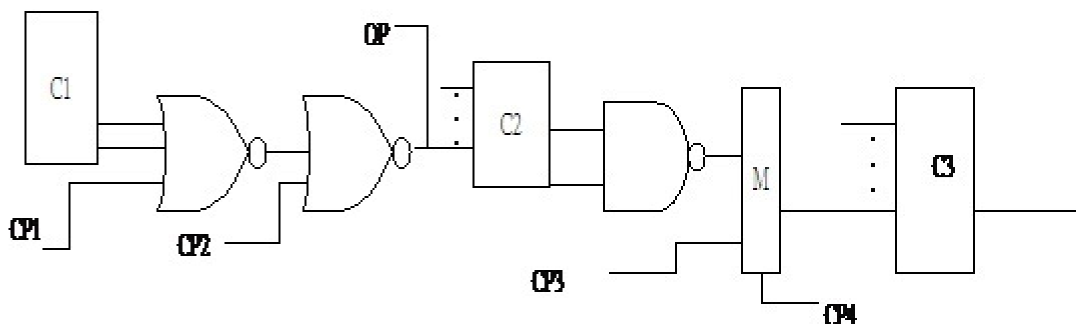


Figure 2.1: Ad-hoc method - Test point insertion

Disadvantages

- Experts are needed and test generation is often manual
- No guarantee of result (poor fault coverage)
- Increase design iterations, hence not suitable for large designs

2.1.2 Structured Method

Structured method is highly automated DFT technique which provides good controllability and observability of internal state variables for testing by serial shifting of data. This method includes

- Scan technique

  In this method all the normal flops in the design are replaced by scan flops

- Scan design requirements

  - One (more) test control ports are required at PI and PO, which are called as scan-in and scan-out port respectively

- Test structure (hardware) is added for DUT
- Normal Flip flops are converted to Scan Flip flops (Scan cells) and are connected so that they behave as a shift register in test mode. Scan Flip Flop is a flop with extra logic
- Combinational ATPG is used to obtain tests for all testable faults in the combinational logic
- Shift registers tests are applied and ATPG are converted into scan sequences for use in manufacturing test.

• Different types of Scan cells used for scan design

a. Mux based scan cells

In this approach MUX is inserted at the input side of flip flop. MUX's select line is connected to scan enable, which decides the operating mode of the design. This MUX is added to increase controllability at input side of each flip flop. Two types of inputs, Data and Scan input are connected to D0 and D1 pins of MUX respectively. Based on scan enable signal, corresponding input is fed to flop. Fig.2.2 shows a basic mux based scan cell.

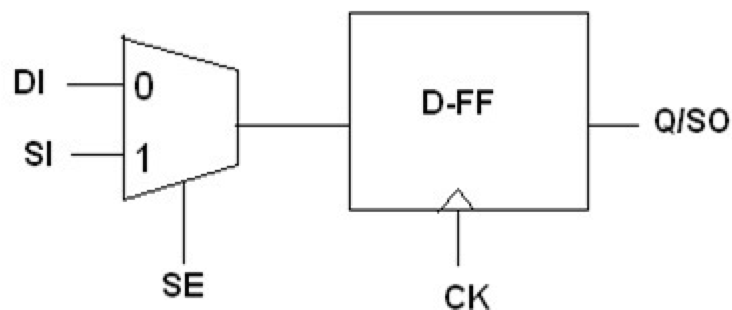


Figure 2.2: MUX based Scan cell

b. Clocked Scan cells

In clocked scan cells, input selection is conducted using two Independent clocks. In normal/capture mode, the data clock is used to capture the contents present on the data input into the clocked scan cell. In shift mode, the shift clock is used to shift the new data from scan input to clocked scan cell, while the content of clocked scan cell is being shifted out.Fig shows a basic Clocked scan cell.

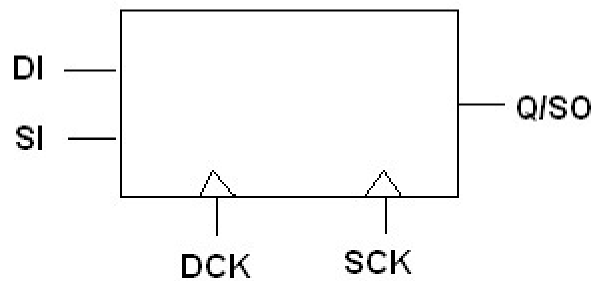


Figure 2.3: Clocked scan cell

- LSSD

  LSSD stands for level sensitive scan design. It is a latch based design which guarantees race-free and hazard-free system operation as well as testing. It is insensitive to component timing variations such as rise time, fall time and delay. It uses two latches (one for normal operation and another for scan) and three clocks. LSSD requires that the circuit to be level sensitive. Fig. 2.4 shows LSSD.

  - Normal mode : A-clk = B-clk = 0, sys-clk =0 —- 1
  - Test (scan) mode: sys-clk =0, A-clk,B-clk = 10 —- 01 to shift scan data through Latch

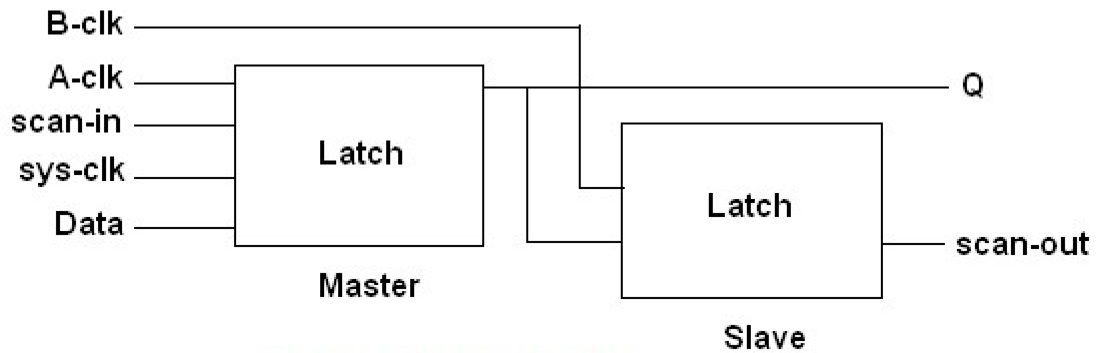


Figure 2.4: Level sensitive scan design

Advantages

– FSM is reduced to combinational logic as for as testing is concerned, Hazards and Races are eliminated, which simplifies test generation and fault simulation

Disdvantages

– Complex design rules are imposed on the designers

– Asynchronous designs are not allowed in this approach

– Sequential routing of latches can introduce irregular structure

– Test application becomes very slow process and not good for memory intensive designs

- Boundary Scan

JTAG or Boundary scan is primarily used for testing board connections, without unplugging the chip from the board. Boundary scan is accessed through 5 pins, TCK, TMS, TRST, TDI and TDO. It builds capability of observing and controlling pins into each chip to make board test easier. Chips with internal scan chains can access the chains through boundary scan for unified test strategy. Fig.2.5 shows Boundary scan method in which scan chains are built for boundary scan cells for different chips.

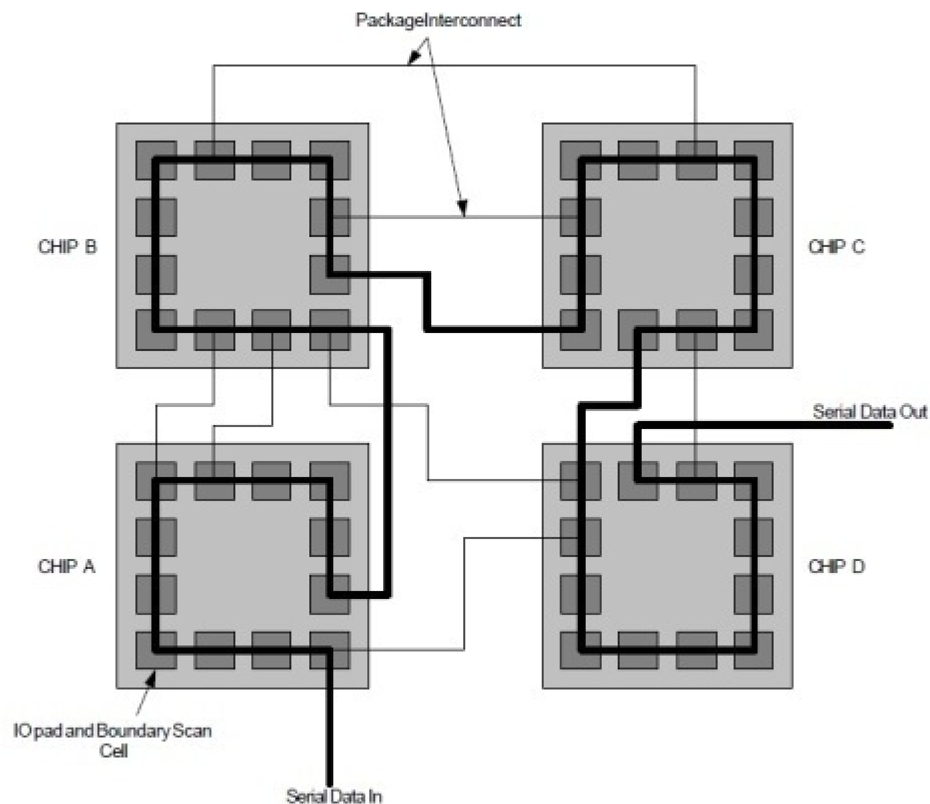


Figure 2.5: Boundary Scan

2.2 Fault classification

Fault is basically the physical defect in the circuit. Fault classification mainly explains the different kinds of faults that may present in Design under Test. It includes,

a. Permanent Faults

These kinds of faults will be present permanently in the design. It has four types

- Stuck-at fault: Fault in the logic gate results in one of its input or output node is fixed at either logic 0 or logic 1. In general for m number of inputs, there will be 2(m+1) number of stuck-at faults will be present.

- Delay fault: It includes transition and path delay faults. Transition delay fault model includes single node slow-to-rise and slow-to-fall faults. Path delay faults tests and categorizes critical timing paths in the design.
- IDDQ fault: IDDQ fault model assumes that the circuit defect will cause excessive current drain due to internal short circuit from node to ground or to power supply
- Bridging fault : Two or more signal lines are connected accidentally in the logic circuit

b. Temporary Faults

It includes Intermittent and Transient faults. Intermittent faults are recurring faults which will reappear on regular basics. Transient faults are non recurring and non repairable faults, because there is no physical damage to the hardware.

c. Equivalent Faults

Two faults are said to be equivalent if every test for one fault also detects the other.

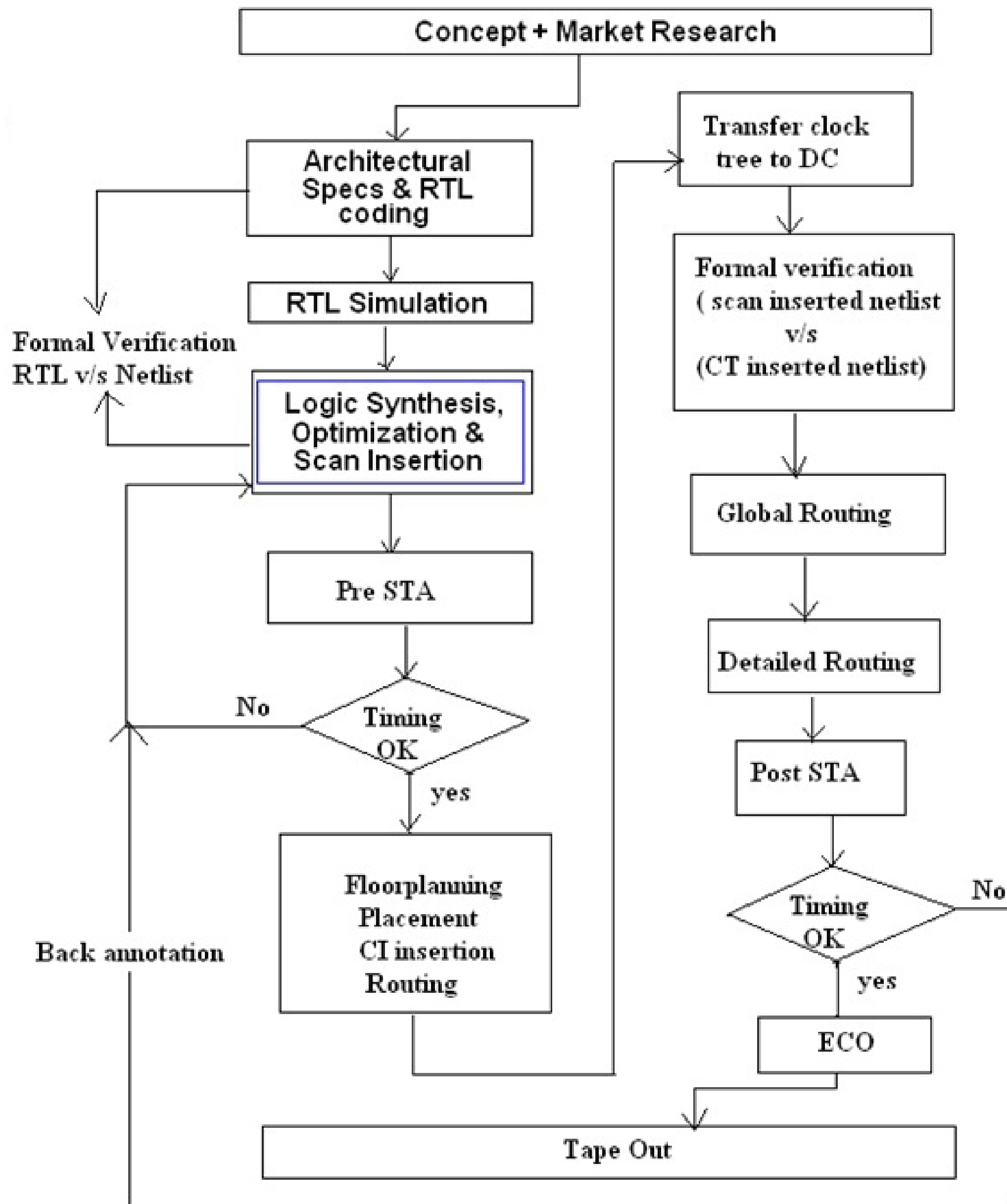
d. Redundant Faults

Faults are said to be redundant/latent its effect does not results on the output logic.

2.3 Traditional ASIC design flow

Chip design commences from the concept of idea dictated by the market. These ideas are then translated into architectural and electrical specification. The Architectural specification will define the functionality and partioning of chip. The electrical specification will define the inter connection of these blocks in terms of timing information. Next step is to implement these design specifications. In past, early days, these specifications are implemented manually by drawing schematics and layouts, which is very

time consuming and impractical for design reuse. To overcome these problems Hardware description language (HDL) were developed. Most commonly used are Verilog and VHDL. Fig.2.6 shows the flow chart of ASIC chip designing and my project part lies in the marked block.



Initially, RTL code is developed according to design specification by Verilog or VHDL. To check the functionality of the design RTL code is simulated using test bench. If the code meets all the design specifications, then synthesis is carried out. Synthesis is done by a Synopsis tool called Design Compiler (DC). DC translates the RTL design into a gate level optimized mapped netlist. These RTL and netlist are formally verified for logical equivalence by a formality tool.

Static Timing Analysis (STA) will allow the user to analyze all the critical path in the design. Prime Time (PT) is used for timing analysis. In pre-layout STA PT uses wire load model specified in the library to estimate the delays. The timing constraints are fed to PT by providing the relation between primary IOs and clocks. If the timing for all the critical path is satisfactory, a constraint file is developed for forward annotation to layout tools. This standard constraint file is Standard Delay Format (SDF).

If timing in pre-layout STA is satisfied, Then Placement and routing is done. It consists of five steps; Initial floor planning, Cell placement, Clock tree (CT) insertion, Global and then detailed routing. CT insertion is done to check the quality of cell placements. The netlist generated during synthesis lacks from CT information. Hence CT is re-routed to netlist and formally verified for CT inserted netlist and Original netlist. Global routing will estimates the actual delays. If timings are met, detailed routing is done where real delays are estimated. In next step, Post layout STA the extracted delays are back-annoted until timing requirements are satisfied. Before Tape out of the design, if there is any hardware bug encounters in the design, this bug can be removed by Engineering Change Order (ECO) instead of redesigning.

DFT insertion can be done directly if design under test (DUT) is in netlist (gate level) format. If the DUT is in RTL format, initially synthesis is performed to translate RTL design into a netlist and formal verification is done to check logical equivalence of RTL and netlist design. Once formality check is succeeded, then DFT insertion is done. Further Timing Analysis is done for delay check and TetraMAX is run to increase test coverage and to validate the scan chains.

2.4 Synthesis

Synthesis is a process to translate the RTL designs into a gate-level, optimized, mapped netlist. Fig.2.7 shows the synthesis flow which is used to convert RTL designs into gate level netlist.

- Develop HDL files

  Usually the input files for synthesis are written in Verilog or VHDL. When writing the HDL files, designers had to follow design partioning and coding guidelines to achieve the best synthesis result possible. This is given by the RTL designers. This step is not included in synthesis.

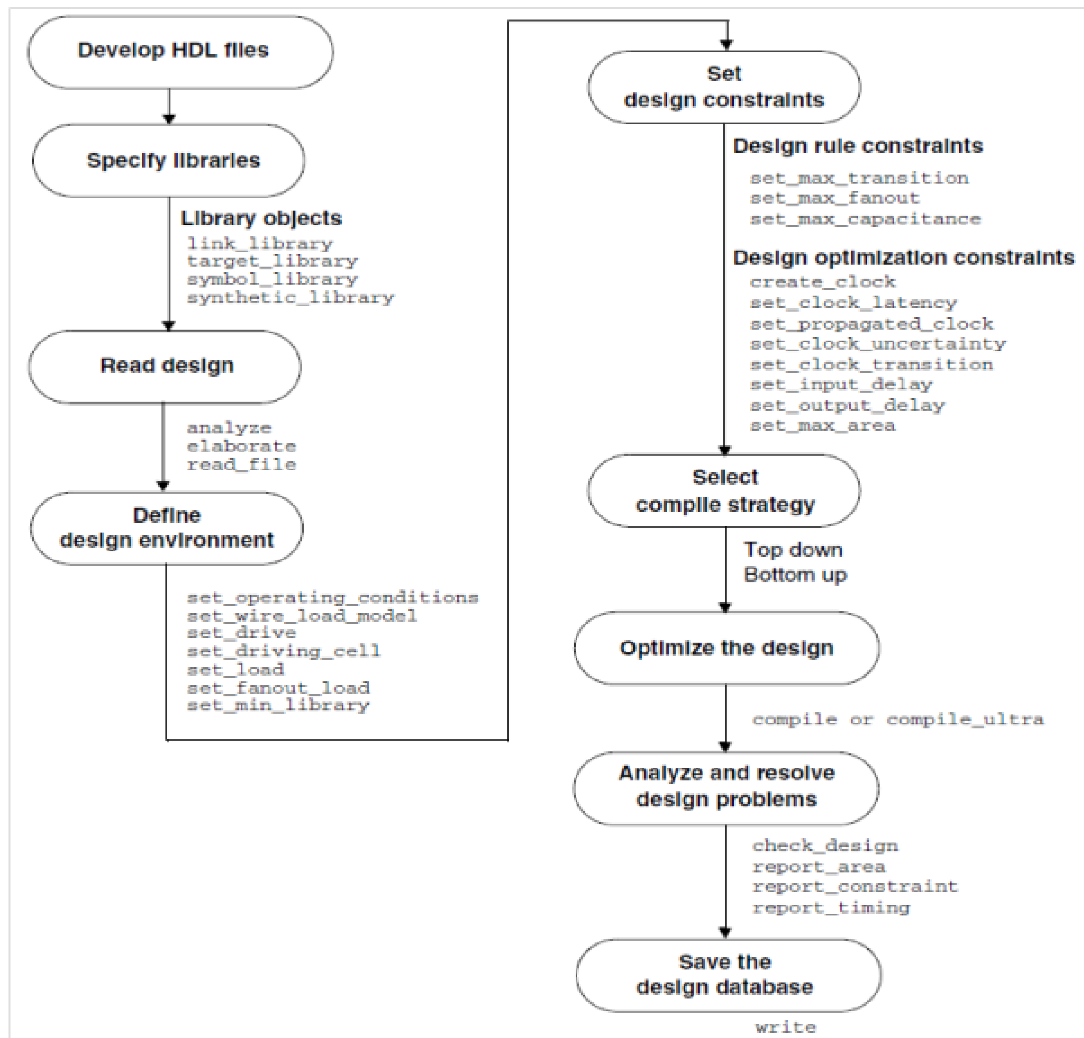


Figure 2.6: Basic Synthesis flow

- Specify libraries

  Initially, all the libraries required for synthesis are stored in .synopsys dc.setup file, which includes Link library, Target library, GTECH library and Design ware library. This setup file will automatically links during synthesis in the design environment.

  Ex:

  Set link library /sw/unicad/CLOCK65LPHVT/5.1/libs/ CORE65LPHVT wc 1.10V 125C 10y.db /sw/unicad/CORE65LPLVT/5.1/libs/ CORE65LPHVT wc

1.10V 125C 10y.db /sw/unicad/CORE65LPHVT/5.1/libs/ CORE65LPHVT wc 1.10V 125C 10y.db
Set target library /sw/unicad/CORE65LPSVT/5.1/libs/CORE65LPSVT wc 1.10V 125C 10y.db

- Read design

  Analyze and elaborate commands are used to read Verilog/VHDL design into dc shell Environment. Analyze command will stores the current design in an intermediate format in the specified library. Elaborate will builds the design from this intermediate format.

  Ex: analyze -format verilog/vhdl ¡ .v/.vhd file ¿ elaborate ¡ top design name ¿

- Define design environment

  In this step we will specify the operating condition (wire load or Zero wire load) of the design.

- Set design Constraints

  The design constraints like clock latency, clock transition, clock uncertainty, IO delays etc... are added to the design environment.

  Ex: create clock -name ¡ clk name ¿ -period 10 -waveform 0 5 [get ports ¡ port ame¿] set clock latency -max 1 [get clocks clk name] set clock uncertainty -setup 0.47 [get clocks clk name] set clock uncertainty -hold 0.25 [get clocks clk name] set input delay 5.0 min/max -clock ¡ clk name¿ [get ports ¡ port name¿]

- Select Compile Strategy

  There are two types of compiling the design, Top down and Bottom up. In top down type, the entire design is compiled at a time where as in bottom up type, the entire design is broken into subdesigns and compiled.

- Optimize the design

In this stage, optimization and mapping is done by compiling the design. Further compile -scan will do scan replacement in addition to optimization and mapping.

Ex: compile -scan

- Analyze and Resolve Design problems

  In this stage, the design problems are fixed by changing the constraints until the design requirements are met.

- Write stage

  The translated RTL design into the netlist is saved in the form of .v/.ddc file by using write command

  Ex: write -format verilog -hierarchy -output ¡ outputfile.v ¿

2.5 Formal Verification

Formal verification is a technique that performs the validation of the design using Mathematical methods. Formal verification is an alternate to verification through simulation. The main advantage is, verification is done by mathematical methods without using test vectors. Two versions of design (RTL-RTL, RTL-Netlist, Netlist-Netlist) are logically checked for logical equivalence. To perform this equivalence checking, formality tool (synopsys EDA tool) is used.

Equivalence checkers prove or disprove that one design representation is logically equivalent to another. That is to say, they are used to prove that two circuits will exhibit the same exact behavior under all conditions despite different representations. Fig.2.8 shows the basic idea of Formal verification setup in which Design A and Design B are reference and implementation design respectively.

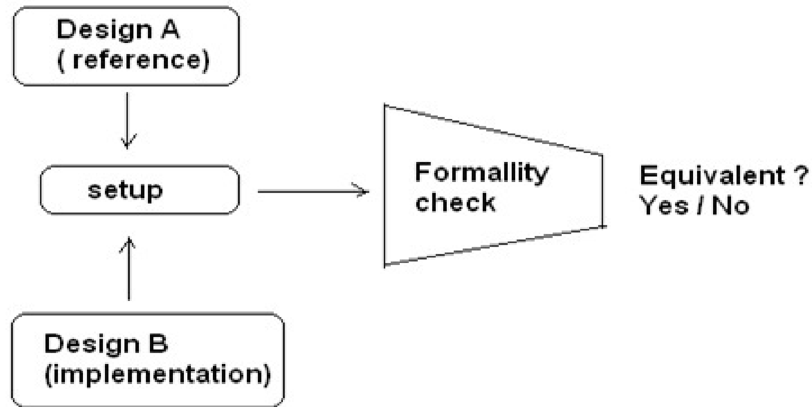


Figure 2.7: Formal verification

The Reference (golden) design and implementation design is loaded to fm shell and setup is performed. Formality tool automatically generates compare points in both versions of the design for matching. The compare points are:

- Primary outputs (Pos)
- Internal registers (sequential elements)
- Input pins of black boxes and
- Nets driven by multiple drivers.

The tool performs matching in five steps;

- Exact - name matching
- Name filtering
- Topological Equivalence
- Signature Analysis
- Compare point matching based on net names

  If all the compare points of Design A and Design B are matched, Equivalence check is done successfully

2.6 Timing Analysis

Timing analysis is performed by a synopsys tool called primetime (PT). PrimeTime is a full-chip, gate-level static timing analysis tool that is an essential part of the design and analysis flow for today's large chip designs. PrimeTime exhaustively validates the timing performance of a design by checking all possible paths for timing violations, without using logic simulation or test vectors.

2.6.1 Types of Checking performed

PrimeTime performs the following types of design checking:

- Setup, hold, recovery, and removal constraints
- User-specified data-to-data timing constraints
- Clock-gating setup and hold constraints
- Minimum period and minimum pulse width for clocks
- Design rules (minimum/maximum transition time, capacitance, and fan-out)

Static timing analysis is a method of validating the timing performance of a design by checking all possible paths for timing violations. PrimeTime checks for violations in the same way that we would do it manually, but with much greater speed and accuracy. To check a design for violations, PrimeTime breaks the design down into a set 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.

2.6.2 Timing Paths

The first step performed by PrimeTime for timing analysis is to break the design down into a set of timing paths. Each path has a start point and an endpoint. The

start point is a place in the design where data is launched by a clock edge. The data is propagated through combinational logic in the path and then captured at the endpoint by another clock edge. The start point of a path is a clock pin of a sequential element, or possibly an input port of the design (because the input data can be launched from some external source). The endpoint of a path is a data input pin of a sequential element, or possibly an output port of the design (Because the output data can be captured by some external sink). Fig.2.16 shows different timing paths. Path 1 is path between input port to reg, path 2 is reg to reg path, path 3 is path between reg to output port and path 4 is path between input to output port.

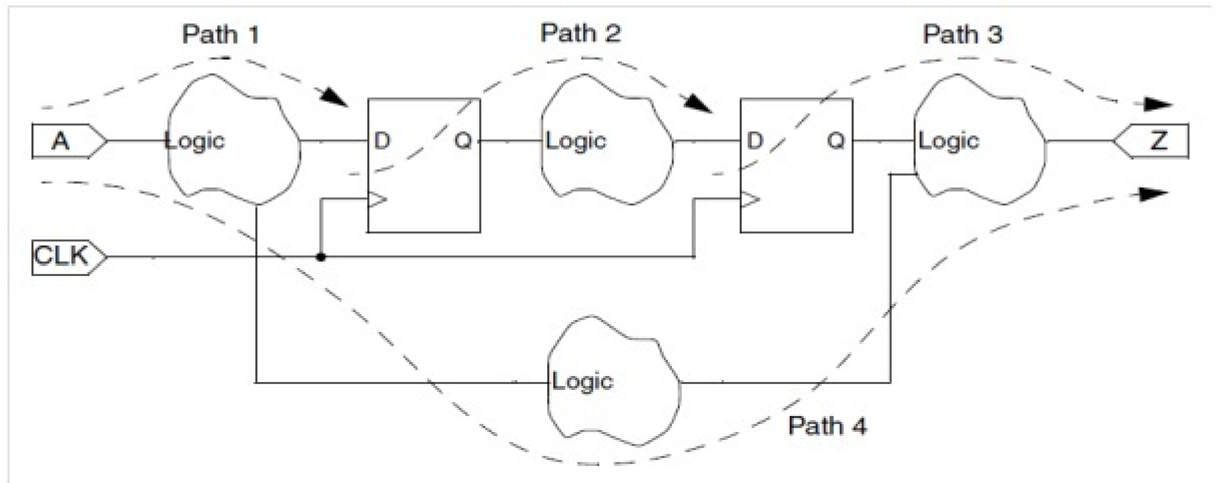


Figure 2.8: Timing paths

2.6.3 Delay Calculation

After breaking down a design into a set of timing paths, PrimeTime calculates the delay along each path. The total delay of a path is the sum of all cell and net delays in the path. Cell delay is the amount of delay from input to output of a logic gate in a path. PrimeTime calculates the cell delay from delay tables provided in the technology library for the cell. Net delay is the amount of delay from the output of a cell to the input of the next cell in a timing path. This delay is caused by the parasitic capacitance of the interconnection between the two cells, combined with net resistance and the limited drive strength of the cell driving the net

$$\text{Total delay} = \text{Cell delay} + \text{Net delay}$$

2.6.4 Constraints Checking

After PrimeTime determines the timing paths and calculates the path delays, it can check for violations of timing constraints, such as setup and hold constraints. A setup constraint specifies how much time is necessary for data to be available at the input

of a sequential device before the clock edge that captures the data in the device. This constraint enforces a maximum delay on the data path relative to the clock path. A hold constraint specifies how much time is necessary for data to be stable at the input of a sequential device after the clock edge that captures the data in the device. This constraint enforces a minimum delay on the data path relative to the clock path.

In addition to setup and hold constraints, PrimeTime can also check recovery/removal constraints, data-to-data constraints, clock-gating setup/hold constraints, and minimum pulse width for clock signals. The amount of time by which a violation is avoided is called the slack. For example, for a setup constraint, if a signal must reach a cell input at no later than 8 ns and is determined to arrive at 5 ns, the slack is 3 ns. A slack of 0 means that the constraint is just barely satisfied. A negative slack indicates a timing violation. Fig.2.17 shows block diagram of data path and clock path which is required to calculate setup/hold time and timing diagram of setup and hold check.

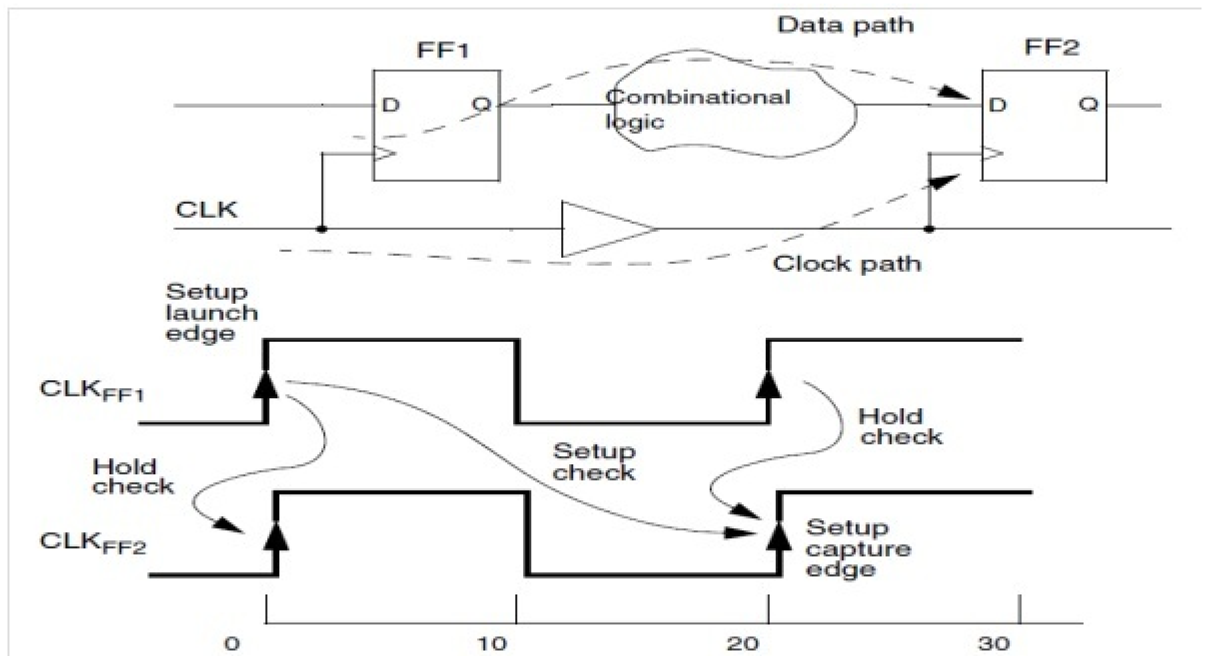


Figure 2.9: Setup and Hold checks

2.6.5 Timing Exceptions

When certain paths are not intended to operate according to the default setup/hold behavior assumed by PrimeTime, we should specify those paths (false paths, multi cycle paths etc...) as timing exceptions. Otherwise, PrimeTime might incorrectly report those paths as having timing violations.

2.7 Setup and Hold Checking for Latches

Latch-based designs typically use two-phase, non-overlapping clocks to control successive registers in a data path. In these cases, PrimeTime can use time borrowing to lessen the constraints on successive paths.

For example, consider the two-phase, latch-based path shown in figure. All three latches are level-sensitive, with the gate active when the G input is high. L1 and

L3 are controlled by PH1, and L2 is controlled by PH2. A rising edge launches data from the latch output, and a falling edge captures data at the latch input. For this example, consider the latch setup and delay times to be zero.

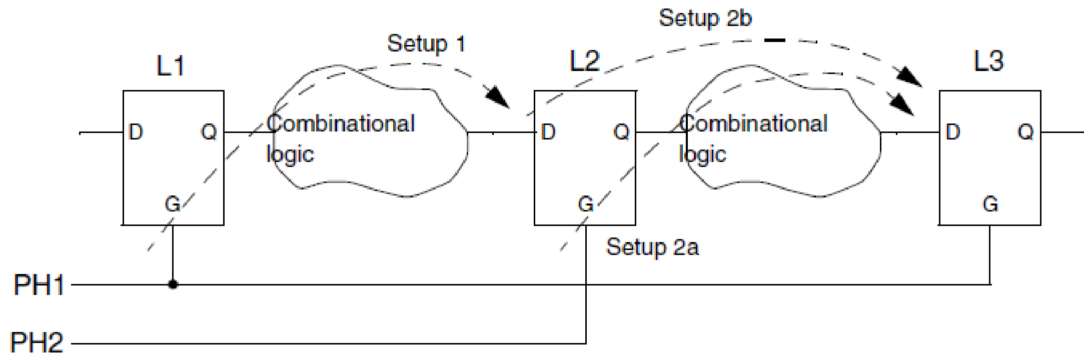


Figure 2.10: Latch-Based Paths

Figure shows how PrimeTime performs setup checks between these latches. For the path from L1 to L2, the rising edge of PH1 launches the data. The data must arrive at L2 before the closing edge of PH2 at time=20. This timing requirement is labeled Setup 1. Depending on the amount of delay between L1 and L2, the data might arrive either before or after the opening edge of PH2 (at time=10), as indicated by the dashed-line arrows in the timing diagram. Arrival after time=20 would be a timing violation.

If the data arrives at L2 before the opening edge of PH2 at time=10, the data for the next path from L2 to L3 gets launched by the opening edge of PH2 at time=10, just as a synchronous flip-flop would operate. This timing requirement is labeled Setup 2a.

If the data arrives after the opening edge of PH2, the first path (from L1 to L2) borrows time from the second path (from L2 to L3). In that case, the launch of data for the second path occurs not at the opening edge, but at the data arrival time at L2, at some time between the opening and closing edges of PH2. This timing requirement is labeled Setup 2b. When borrowing occurs, the path originates at the D pin rather than the G pin of L2.

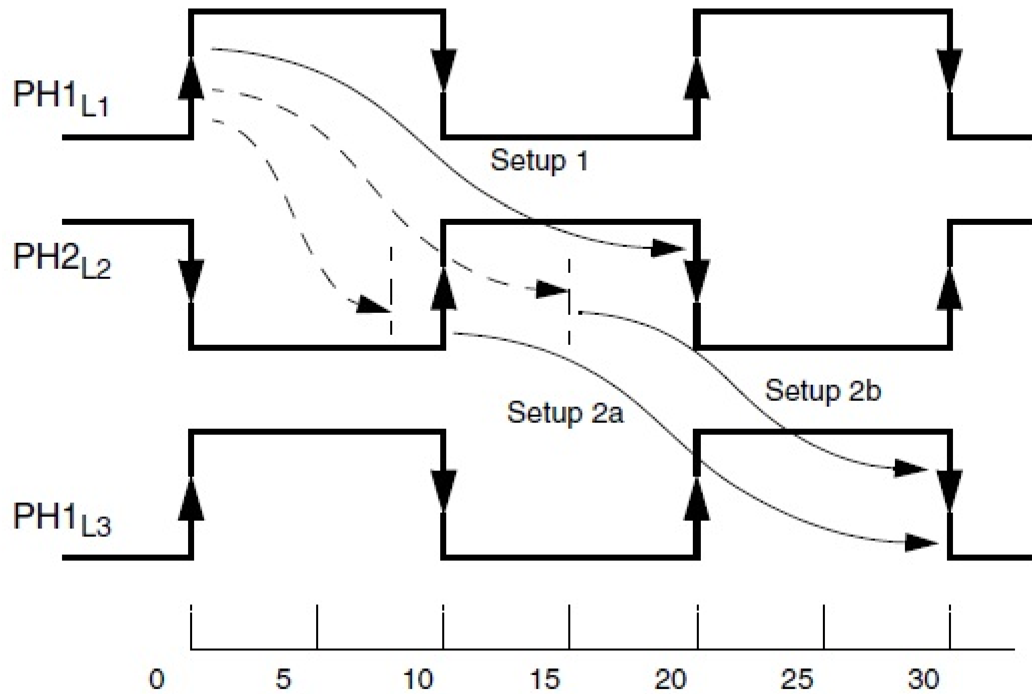


Figure 2.11: Time Borrowing in Latch-Based Paths

For the first path (from L1 to L2), PrimeTime reports the setup slack as zero if borrowing occurs. The slack is positive if the data arrives before the opening edge at time=10, or negative (a violation) if the data arrives after the closing edge at time=20.

To perform hold checking, PrimeTime considers the launch and capture edges relative to the setup check. It verifies that data launched at the startpoint does not reach the endpoint too quickly, thereby ensuring that data launched in the previous cycle is latched and not overwritten by the new data. This is depicted in Figure.

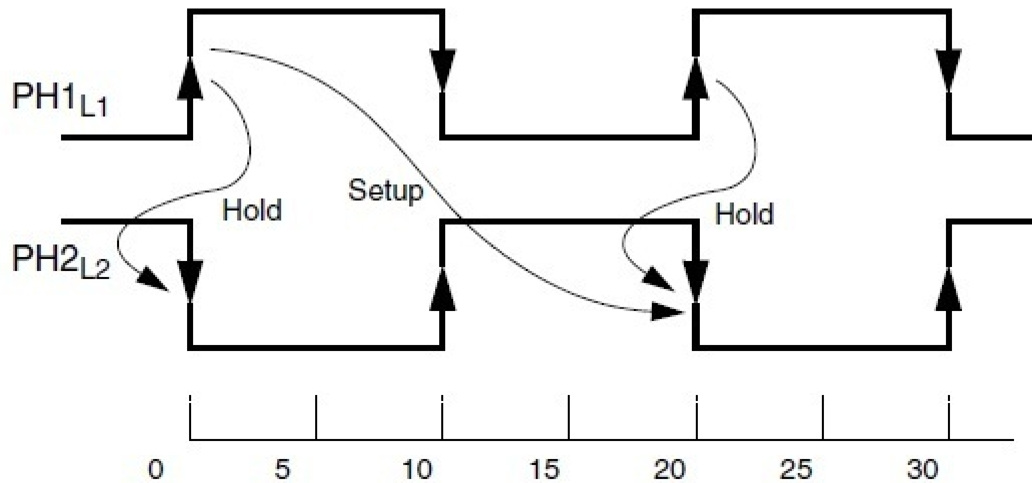


Figure 2.12: Hold Checks in Latch-Based Paths

2.7.1 Timing Exceptions

When certain paths are not intended to operate according to the default setup/hold behavior assumed by PrimeTime, you should specify those paths as timing exceptions. Otherwise, PrimeTime might incorrectly report those paths as having timing violations.

PrimeTime lets you specify the following types of timing exceptions:

- False path - A path that is never sensitized due to the logic configuration, expected data sequence, or operating mode.

- Multicycle path - A path designed to take more than one clock cycle from launch to capture.

- Minimum/maximum delay path - A path that must meet a delay constraint that you specify explicitly as a time value.

2.8 Setup and Hold Checks

This section provides examples of how setup and hold timing checks are done for a single operating condition, for simultaneous best-case/worst-case operating conditions, and for onchip variation.

2.8.1 Path Delay Tracing for Setup and Hold Checks

Figure shows how setup and hold checks are done in PrimeTime.

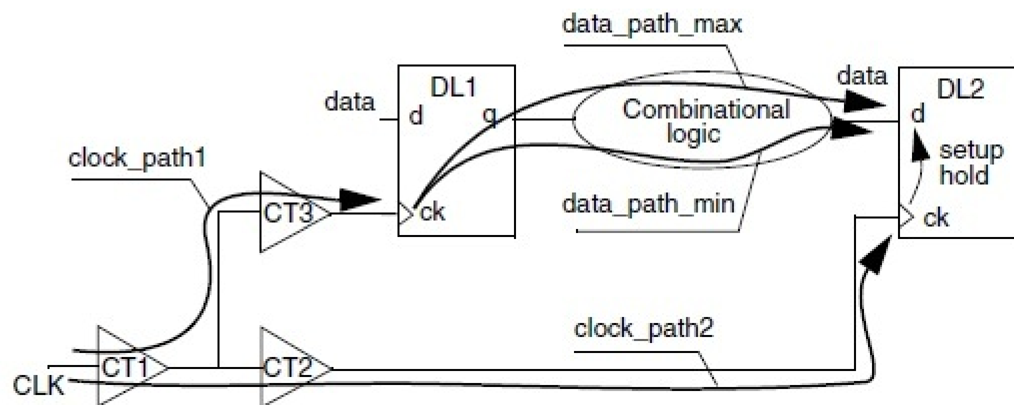


Figure 2.13: Design Example

- The setup timing check from pin DL2/ck to DL2/d considers
  - Maximum delay for clock-path1
  - Maximum delay for data path (data-path-max)
  - Minimum delay for clock-path2
- The hold timing check from pin DL2/ck to DL2/d considers
  - Minimum delay for clock-path1
  - Minimum delay for data path (data-path-min)
  - Maximum delay for clock-path2

The data-path-min and data-path-max values can be different due to multiple topological paths in the combinational logic that connects DL1/q to DL2/d.

2.8.2 Setup Timing Check for Worst-Case Conditions

Figure shows how cell delays are computed for worst-case conditions. To simplify the example, the net delays are ignored.

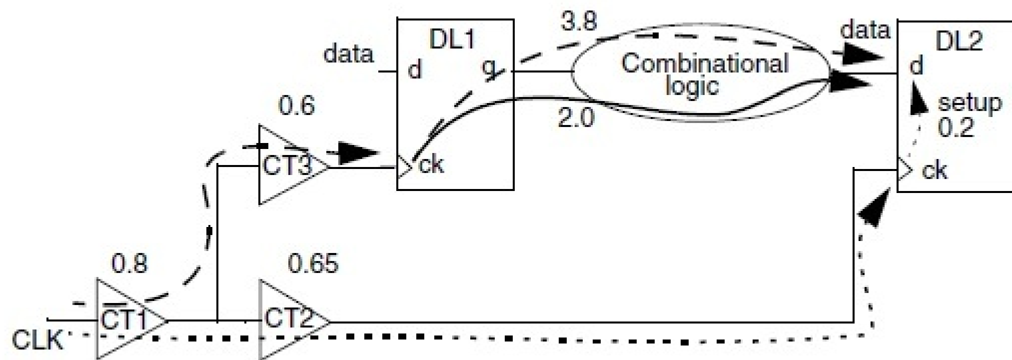


Figure 2.14: Setup Check Using Worst-Case Conditions

PrimeTime checks for a setup violation as follows:

clockpath1 +datapathmax - clockpath2 +setup !LnE! clockperiod

In the equation,

clockpath1 = 0.8 + 0.6 = 1.4

datapathmax = 3.8

clockpath2 = 0.8 + 0.65 = 1.45

setup = 0.2

The clock period must be at least 1.4 + 3.8 1.45 + 0.2 = 3.95.

2.8.3 Hold Timing Check for Best-Case Conditions

Figure shows how cell delays are computed for best-case conditions. Note that the cell delays are different from the delays in Figure.

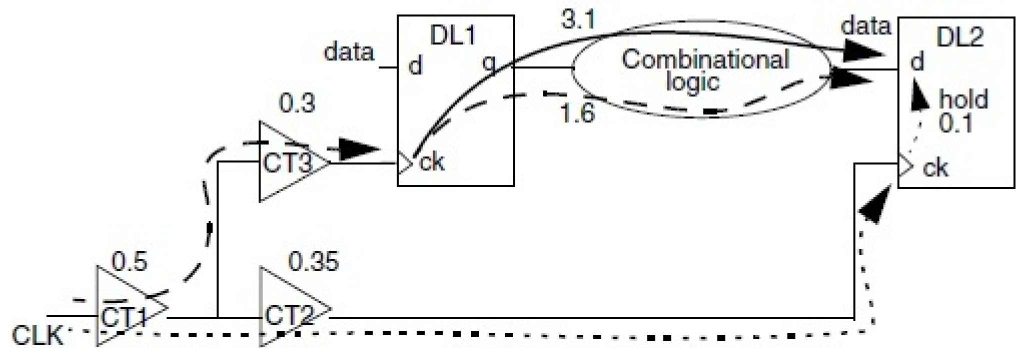


Figure 2.15: Hold Check Using Best-Case Conditions

PrimeTime checks for a hold violation as follows:

clockpath1+datapathmin-clockpath2 - hold !GnE! 0

In the equation,

clockpath1 = 0.5 + 0.3 = 0.8

datapathmin = 1.6

clockpath2 = 0.5 + 0.35 = 0.85

hold = 0.1

No hold violation exists because 0.8 + 1.6 0.85 0.1 = 1.45, which is greater than 0.

2.8.4 Simultaneous Best-Case/Worst-Case Conditions

PrimeTime can operate in a mode that computes delays simultaneously for best-case and worst-case conditions for hold and setup checks, enabling you to check both conditions in one timing analysis run. In terms of path delay tracing, PrimeTime uses worst-case conditions for setup checks and best-case conditions for hold checks. The timing reports for setup and hold checks are the same as for Figure and Figure.

In simultaneous mode, PrimeTime does not compare data arrival at worst-case conditions to clock arrival at best-case conditions. In this mode, the timing reports show delays computed in the same operating condition (worst case for setup or best case for hold).

2.9 Clock Reconvergence Pessimism Removal

Clock reconvergence pessimism is an accuracy limitation that occurs when two different clock paths partially share a common physical path segment and the shared segment is assumed to have a minimum delay for one path and a maximum delay for the other path. This condition can occur any time that launch and capture clock paths use different delays, most commonly with on-chip variation analysis. Automated correction of this inaccuracy is called clock reconvergence pessimism removal (CRPR).

PrimeTime performs CRPR, when enabled, at the same time as regular timing analysis. You need to enable CRPR before you do timing analysis. For information about enabling CRPR. The following examples demonstrate how the pessimism removal works.

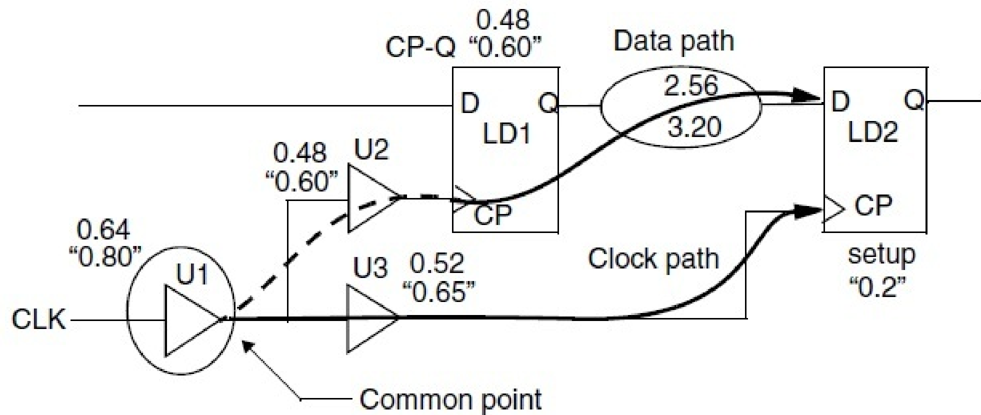


Figure 2.16: Clock Reconvergence Pessimism Example

Figure shows how the analysis is done. Each delay (considered equal for rising and falling transitions to simplify this example) has a minimum value and a maximum value computed for the minimum and maximum operating conditions.

The setup check at LD2/CP considers the clock path to the source latch (CLK to LD1/CP) at 100 percent worst case, and the clock path to the destination latch (CLK to LD2/CP) at 80 percent worst case.

Although this is a valid approach, the test is pessimistic because clock path1 (CLK to LD1/ CP) and clock path2 (CLK to LD2/CP) share the clock tree until the output of U1. The shared segment is called the common portion, consisting of just cell U1 in this example. The last cell output in the shared clock segment is called the common point, which is the output of U1 in this case.

The setup check considers that cell U1 simultaneously has two different delays, 0.64 and 0.80, resulting in a pessimistic analysis in the amount of 0.16. This amount, obtained by subtracting the earliest arrival time from the latest arrival time at the common point, is called the clock reconvergence pessimism.

This inaccuracy also occurs in an analogous way for the hold test at the LD2 latch.

2.9.1 Reconvergent Logic Example

Figure shows a situation where clock reconvergence can occur, even in the absence of on-chip variation analysis. In this example, there is reconvergent logic in the clock network. The two clock paths that feed into the multiplexer cannot be active at the same time, but an analysis could consider both the shorter and longer paths for one setup or hold check.

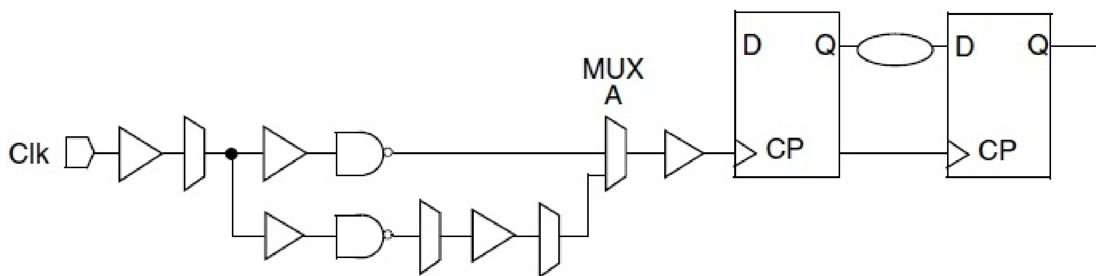


Figure 2.17: Reconvergent Logic in a Clock Network

2.9.2 Minimum Pulse Width Checking Example

The report-constraint command checks for minimum pulse width violations in clock networks (as specified by the set-min-pulse-width command) and at cell inputs (as

specified in the technology library). CRPR, when enabled, can increase the accuracy of minimum pulse width checking. For information about enabling CRPR.

For example, consider the circuit shown in Figure. The external clock source has early and late source latency set on it. In addition, the two buffers in the path have minimum and maximum rise and fall delay values defined.

The report-constraint command checks the pulse width of the clock signal in the clock network and upon reaching the flip-flop. For level-high pulse width checking, PrimeTime considers maximum delay for the rising edge and minimum delay for the falling edge of the clock (and conversely for level-low pulse width checking).

For the example shown in, in the absence of CRPR, the worst-case pulse width is very small and violates the pulse width constraint of the flip-flop. However, this analysis is pessimistic because it assumes simultaneous worst-case delays for rising and falling edges. In a real circuit, rising-edge and falling-edge delays are at least somewhat correlated. For example, for the delay from the external clock source to the CLK input port, if the rising-edge delay is at the minimum, 1.3, the falling-edge delay is probably equal or close to 1.3, and not at the maximum of +1.4.

To account for correlation between rising-edge and falling-edge delays, enable CRPR. In that case, PrimeTime adds a certain amount of slack back into the minimum pulse width calculation. The amount added is equal to the range of minimum rise delay or the range maximum fall delay for the path, whatever is smaller:

$$\mathbf{crp = min((Mr\ \ mr),\ (Mf\ \ mf))}$$

where crp = clock reconvergence pessimism, Mr = cumulative maximum rise delay, mr = cumulative minimum rise delay, Mf = cumulative maximum fall delay, and mf = cumulative minimum fall delay. For an example of this calculation applied to pulse width checking, see Figure.

2.10 DFT(scan) Insertion

Once the netlist is available, Scan insertion is done to increase the controllability and observability of the internal nodes in the design.

a. Modes of scan Operation

Scan operates in shift and capture cycles. Data is injected into the device through primary inputs and is shifted out of the device through the "SD" input port of the flops. Scan en (SE) port is active high for shift operation. Once the chain has been flushed out and compared, the scan en signal is toggled (driven low). Now a single clock pulse is applied to capture the data into the flops through the "D" inputs, before the scan en is toggled again (driven high) and the data shifted out for comparison. Fig shows scan shift and Fig shows scan capture operation respectively. Whenever scan en is high, the circuit will be operating in scan mode and when it is low, the circuit will operate in normal mode.

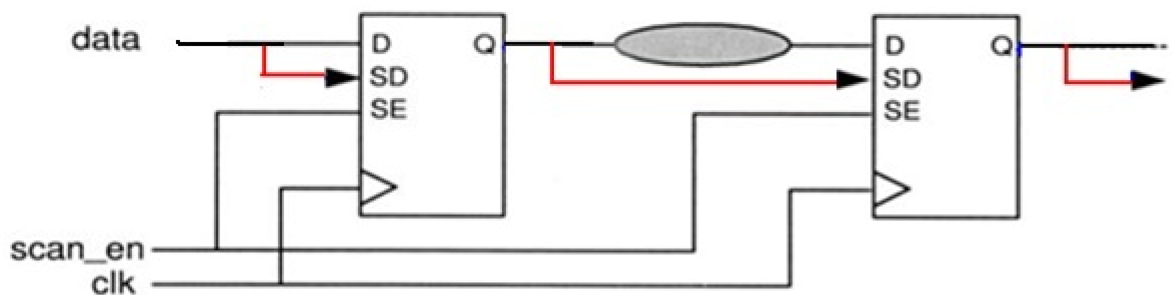


Figure 2.18: Scan shift

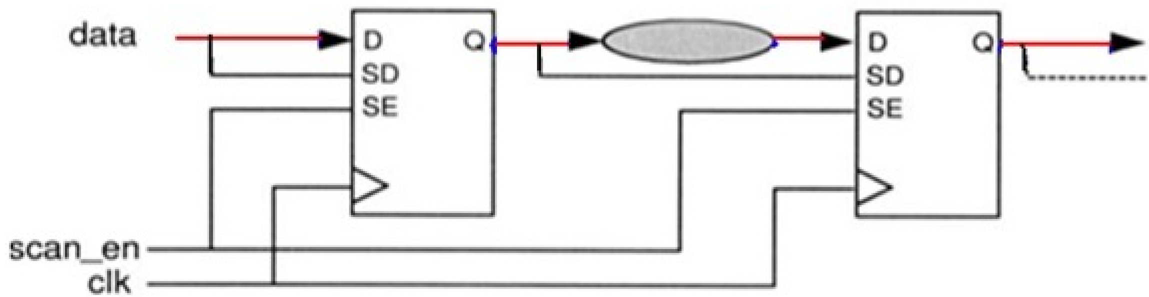


Figure 2.19: Scan capture

b. Scan Insertion (DFT insertion) flow

Fig. 2.11 shows the scan insertion flow for DUT. It includes the following steps,

- Read design : Design is read in the form of ddc/verilog into the dc shell
- Create test protocol : It will describes how the signal should operates in scan mode
- Pre DFT-DRC : Performs design rule checking to scan design and will list out design rule violations before DFT insertion
- Specify scan architecture : It will define type of scan style, number of scan chains, chain length, handling multiple clocks, lockup elements, registers to be omitted from scan chains in the design
- Preview DFT: Checks scan architecture before implementing to actual design. This allows for quick iteration cycles when changes need to be made in scan architecture
- Insert scan : Scan architecture is inserted to the design
- Post DFT-DRC : validates that scan chain trace properly after DFT insertion

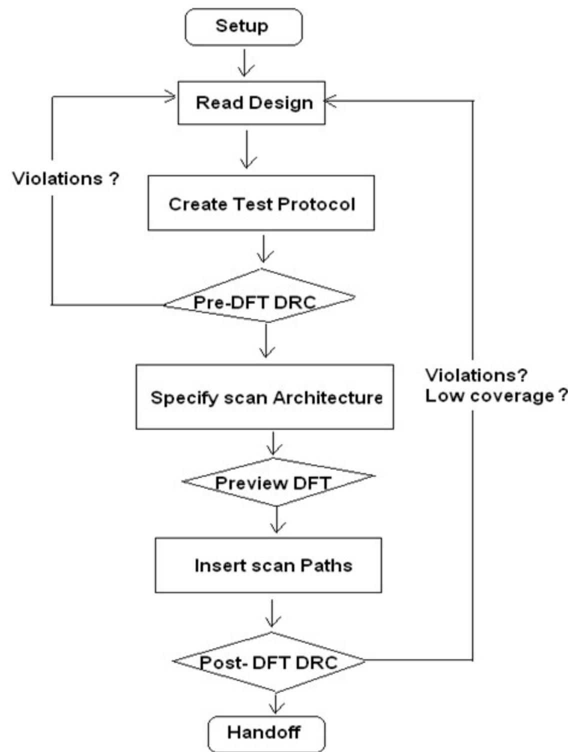


Figure 2.20: DFT insertion flow chart

2.11 DFT compression (codec insertion)

DFT Max is the next generation of DFT synthesis. In this Adoptive scan technique is used for data volume compression with no impact of test coverage. The main advantages are

- Tester cycle reduction
- Test application time reduction
- Uses minimum number of ports

In this technique de-compressor logic is added at the input side of flop to achieve controllability and compressor logic is added after flop for observability. Due to this addition of compressor and decompressor logic, Adaptive scan compression is also

known as Codec insertion. Fig.2.12 shows the codec insertion for scan chains without addition of any extra scan in/out ports.

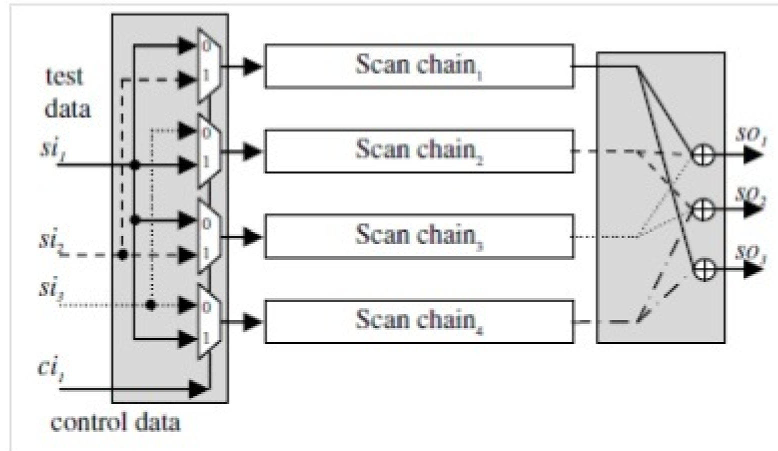


Figure 2.21: Compressor and decompressor logic added to scan chains

There are two modes of operation in DFT Max,

- Internal scan mode (regular scan) : In this mode, de-compressor and compressor logic are bypassed from accessing the scan chain
- Scan compression mode : DFT compiler allows de-compressor and compressor logic to access scan chain

2.11.1 Scan Compression working

Available long scan chains are split into shorter chains. Shorter chains required less time to load and less data to be loaded on tester. In Fig 2.13 two long scan chains with 12 flops in each chain is broken down into 6 chains with 4 flops in each chain without addition of extra scan ports. This reduces test application time due to reduction in chain length, which in turn reduces test cost.

Test application time = patterns * length of scan chain

With higher level of compression come higher area overhead, increased risk of routing congestion and only a small incremental improvement in Test Application Time Reduction (TATR) and Test Data Volume Reduction (TDVR).

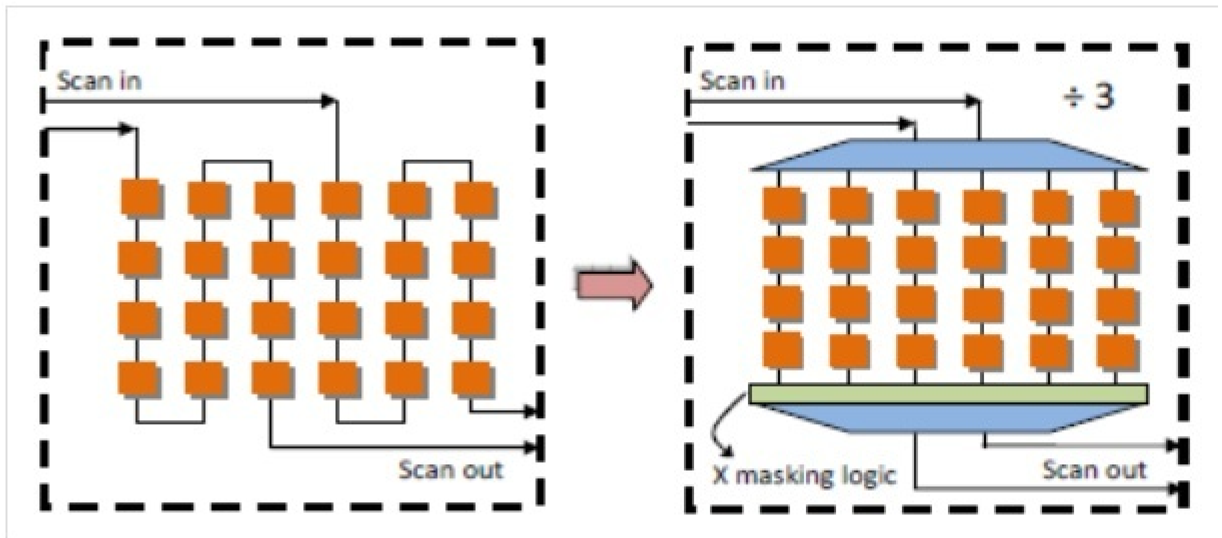


Figure 2.22: Adaptive scan compression

2.11.2 Hierarchical Adaptive Scan Synthesis (HASS)

In HASS Codec insertion is done in bottom-up flow. All the adaptive scan logic is placed at core level and codec is inserted block by block. At top level these codec inserted blocks are combined. This mainly reduces the routing congestion at top level and issues can be debugged easily. Fig.2.14 shows the HASS architecture at top level, where the codec insertion is done for core 1 and core 2 blocks.

An adaptive scan core contains scan chains that are configured in two modes of operations compression mode (scan compression mode) and reconfigurable scan mode (internal scan mode). A pure scan core contains scan chains configured in a single mode of operation (internal scan mode). At the chip level, these cores are integrated to provide two modes of operations:

- Compression mode - activates all adaptive scan chains as well as all pure scan core chains
- Reconfigurable scan mode - activates the reconfigured scan chains of each adaptive scan core as well as all pure scan core chains.

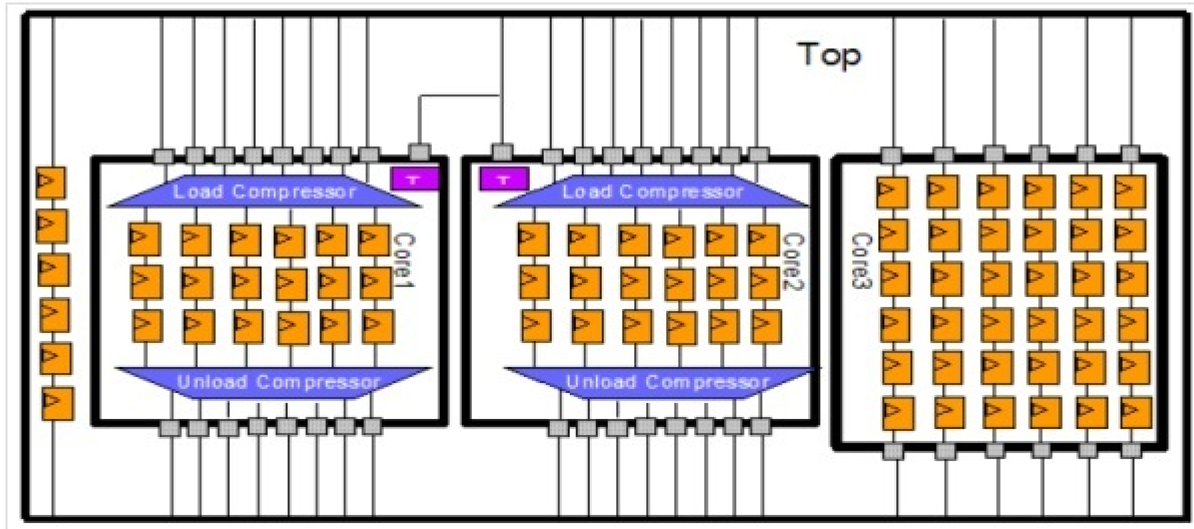


Figure 2.23: Hierarchical Adaptive Scan Synthesis Architecture

2.12 Defining the Operating Conditions

In most technologies, variations in operating temperature, supply voltage, and manufacturing process can strongly affect circuit performance (speed). These factors, called operating conditions, have the following general characteristics:

- Operating temperature variation

  Temperature variation is unavoidable in the everyday operation of a design. Effects on performance caused by temperature fluctuations are most often handled as linear scaling effects, but some submicron silicon processes require nonlinear calculations.

- Supply voltage variation

  The designs supply voltage can vary from the established ideal value during day-to-day operation. Often a complex calculation (using a shift in threshold voltages) is employed, but a simple linear scaling factor is also used for logic-level performance calculations.

- Process variation

This variation accounts for deviations in the semiconductor fabrication process. Usually process variation is treated as a percentage variation in the performance calculation.

When performing timing analysis, Design Compiler must consider the worst-case and best-case scenarios for the expected variations in the process, temperature, and voltage factors.

Chapter 3

Methodology

This chapter includes the methodology used to implement the project. It depends on the type of input (DUT) format. If DUT is in netlist format, then DFT insertion can be done directly. If DUT is in RTL format, before DFT insertion first synthesis is done to get the gate level design (netlist) and formal verification is done between RTL and netlist for logical equivalence check. Let's consider the DUT is in RTL format to cover the entire flow.

3.1 To convert RTL into Gate level netlist

Synthesis is a process to translate the RTL designs into a gate-level, optimized, mapped netlist. To perform synthesis, synopsys design compiler tool is used. Initially all the (link and target) libraries related to design are loaded to .synopsys dc.setup file. The advantage of setup file is, it will automatically link to dc shell environment.

- Read stage

  The input RTL (DUT) is read into dc shell by using following DC commands read verilog/vhdl or analyze and elaborate. Analyze command analyzes the design into an intermediate format and stores in the specified library, elaborate will builds the design from the intermediate format.

- Link stage

  After reading the design, set top module name in RTL to current design and link all the libraries in setup file. To perform this set current design and link command is used.

- Sourcing constraints

  All the required constraints for the design are sourced by source command

- Uniquification stage

  Removes multiply-instantiated hierarchy in the current RTL design by creating a unique design for each cell instance. For this uniquify command is used.

- Compile stage

  There are two methods of compiling the design, Top down and Bottom up. In compile stage, optimization and mapping is done. compile -scan will replaces normal flops with scan flops in addition to optimization and mapping.

- Write stage

  The translated RTL design into the netlist is saved in the form of .v/.ddc file by using write command

3.2 To convert RTL into Gate level netlist

The following script is used to get netlist from RTL in dc shell in Unix environment. The input to the dc shell is RTL (DUT) and the output is gate level optimized mapped netlist.
read verilog (.v RTL design)
set current design (top design name)
link
uniquify

source design constraints

compile

report timing /report constraints /report clocks

write -format (verilog/vhdl) - hierarchy -output ¡ outputfile.v/ddc ¿

3.3 DFT (scan) insertion and stitching

Scan insertion can be done for a netlist (DUT) using synopsys DFT compiler or Design compiler tool. In this project Design compiler tool is used for scan insertion. This part mainly includes Design Rule Checking (DRC), fixing the DRC violations and scan insertion. Design Rules are a series of parameters provided by semiconductor manufacturers that enable the designer to verify the correctness of a mask set. DRC check for DUT is done before and after the scan insertion, because these DRC violations will disturb the scan chain implementation for DUT

a. Steps to perform scan insertion

- Read stage

  Netlist (DUT) is read into dc shell. This netlist can be read in verilog/binary format (.v/.ddc) by using read verilog/ddc command

- Create test protocol

  Test protocol is created according to the user specification by using Create test protocol command. In this we need to define scan clocks, scan enable, reset signals, scan in/out ports.

- " Pre- DFT DRC

  This is design rule checking before the scan insertion. Analyze DUT and fix all DRC violations before scan insertion. Dft drc -pre command is used for pre DRC checks

- " Specify scan architecture

In this stage, we will specify scan flop style, number of scan chains, number of scan flops in each scan chain etc For this set dft signal command is used.

- " Preview DFT

  In this stage we can make changes in scan architecture according to the requirements and allows quick iteration cycles. Preview dft is the command.

- " Scan stitching

  All the scan flops are stitched and scan chain is implemented as specified in scan architecture using insert DFT command.

- " Post- DFT DRC

  After scan insertion, DRC check is performed once again and all the violations need to be fixed. After this test coverage can be obtained. If test coverage is low, once again netlist in resynthesize using required constraints to increase the coverage. Dft drc -post -coverage command is used.

b. Scripts generated to perform Timing analysis
   read verilog (codec inserted netlist)
   current design ¡top design name¿
   link
   update timing
   check timing
   report timing
   report constraints -all violators
   report clocks
   report analysis coverage

3.4 Parasitic Interconnect Corners

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

these are:

- Typical: This refers to the nominal values for interconnect resistance and capacitance.
- Max C: This refers to the interconnect corner which results in maximum capacitance. The interconnect resistance is smaller than at typical corner. This corner results in largest delay for paths with short nets and can be used for max path analysis.
- Min C: This refers to the interconnect corner which results in minimum capacitance. The interconnect resistance is larger than at typical corner. This corner results in smallest delay for paths with short nets and can be used for min path analysis.
- Max RC: This refers to the interconnect corner which maximizes the interconnect RC product. This typically corresponds to larger etch which reduces the trace width. This results in largest resistance but corresponds to smaller than typical capacitance. Overall, this corner has the largest delay for paths with long interconnects and can be used for max path analysis.
- Min RC: This refers to the interconnect corner which minimizes the interconnect RC product. This typically corresponds to smaller etch which increases the trace width. This results in smallest resistance but corresponds to larger than typical capacitance. Overall, this corner has the smallest path delay for paths with long interconnects and can be used for min path analysis.

Chapter 4

Result Analysis

This chapter includes the result of given DUT which is obtained at worst and the best case check after synthesis. Next includes Timing Analysis result netlist. Here I do the timing analysis of 16 sign-off corners (with the name of corner, which technology node(28nm), which spef is being used, standard cell which is used from the library). In this I had analyze the report-timing of the each corner and find the critical path for setup and hold.

			16 signoff TA corners									FUNCTIONAL			
Sr.No	Corner name	Spef used for the same	std cell db used for the same	setup slack for a setup critical path	Data path delay for setup critical path	Launch Clock path delay for setup critical path	Capture Clock path delay for setup critical path	crpr	Clock Skew for Setup critical path	hold slack for a hold critical path	Data path delay for hold critical path	Launch Clock path delay for hold critical path	Capture Clock path delay for hold critical path	crpr	Clock Skew for hold critical path
1	ss28_0.95V_OC_Cmax	OC_Cmax	ss28_0.95V_OC	-0.6158	4.1823	1.3315	1.0963	0.1217	0.1135	0.2754	0.2446	0.3875	0.4499	-0.062	0.0624
2	ff28_1.10V_OC_Cmin	OC_Cmin	ff28_1.10V_OC	1.9212	0.1448	0.2677	0.1958	0.038	0.0339	-0.1997	0.1832	0.4516	0.694	-0.073	0.1694
3	ss28_0.95V_OC_RCmax	OC_RCmax	ss28_0.95V_OC	-0.5753	3.7564	1.1951	0.6385	0.1133	0.4433	-0.3652	0.3309	0.7268	1.154	-0.1133	0.3139
4	ff28_1.10V_OC_RCmin	OC_RCmin	ff28_1.10V_OC	1.9021	0.1673	0.2623	0.1938	0.0339	0.0346	-0.2012	0.1832	0.4484	0.6813	0.062	0.1709
5	ss28_0.95V_125C_Cmax	125C_Cmax	ss28_0.95V_125(	-1.554	5.0541	0.8637	0.6504	0.1106	0.1027	-0.3862	0.3221	0.7514	1.1877	-0.1106	0.3254
6	ff28_1.10V_125C_Cmin	125C_Cmin	ff28_1.10V_125C	1.9196	0.1504	0.2744	0.2063	0.0393	0.0318	-0.2095	0.1856	0.4702	0.7342	-0.0812	0.1828
7	ss28_0.95V_125C_RCma:	125C_RCmax	ss28_0.95V_125C	-0.2417	3.7351	0.82	0.6777	0.1387	0.0036	-0.3413	0.2948	0.7051	1.1144	-0.1182	0.2911
8	ff28_1.10V_125C_RCmin	125C_RCmin	ff28_1.10V_125C	1.9012	0.1734	0.2667	0.2032	0.0351	0.0284	-0.2055	0.1855	0.4666	0.7084	-0.0628	0.179
9	ff28_1.10V_OC_Cmax	OC_Cmax	ss28_1.10V_OC	1.9012	0.1789	0.2546	0.1966	0.0254	0.0326	-0.2231	0.1833	0.4608	0.6866	-0.0488	0.177
10	ss28_0.95V_OC_Cmin	OC_Cmin	ss28_0.95V_OC	-0.3155	3.5464	1.1492	0.7859	0.1101	0.2532	-0.3112	0.3301	0.7101	1.1075	-0.1101	0.2873
11	ff28_1.10V_OC_RCMAX	OC_RCmax	ff28_1.10V_OC	1.9079	0.1554	0.2734	0.1987	0.0402	0.0345	-0.2371	0.1834	0.462	0.7337	-0.0809	0.1908
12	ss28_0.95V_OC_RCmin	OC_RCmin	ss28_0.95V_OC	-0.8235	4.3501	0.7558	0.5833	0.0858	0.0867	-0.3131	0.2938	0.6746	1.0235	-0.0858	0.2631
13	ff28_1.10V_125C_CMAX	125C_RCmax	ff28_1.10V_125C	1.8865	0.1856	0.2725	0.2067	0.037	0.0288	-0.2434	0.1857	0.4771	0.75	-0.072	0.2009
14	ss28_0.95V_125C_Cmin	125C_Cmin	ss28_0.95V_125(	-0.0202	3.6956	0.7946	0.6955	0.1201	-0.021	-0.2906	0.2948	0.69	1.0644	-0.1087	0.2657
15	ff28_1.10V_125C_RCmax	125C_Cmax	ss28_1.10V_125(	1.9063	0.1614	0.2807	0.2104	0.0418	0.0285	-0.246	0.1857	0.4834	0.7769	-0.0903	0.2032
16	ss28_0.95V_125C_RCmin	125C_RCmin	ss28_0.95V_125C	-0.8361	4.361	0.7755	0.5938	0.0931	0.0886	-0.2944	0.2947	0.6881	1.0513	-0.0931	0.2701

Figure 4.1: Worst Negative Slack(WNS)/Critical Path Path for Setup and Hold

Shown below figure is the case of the worst path/critical path in the design for the setup, in other words its the worst negative slack (WNS)for the setup. CMAX-SS-125C (means the corner(CMAX), is it fast or the slow corner, if the ss than its

slow nmos and slow pmos, if the ff than its fast nmos and fast pmos, if its fs than fast nmos and slow pmos and if the sf than slow nmos and fast pmos , and the last one is defining the temperature) corner contains the critical path for the setup and that is optimize with the other worst path(RCMIN-SS-0C) with the same path that contain for the CMAX-SS-125C.

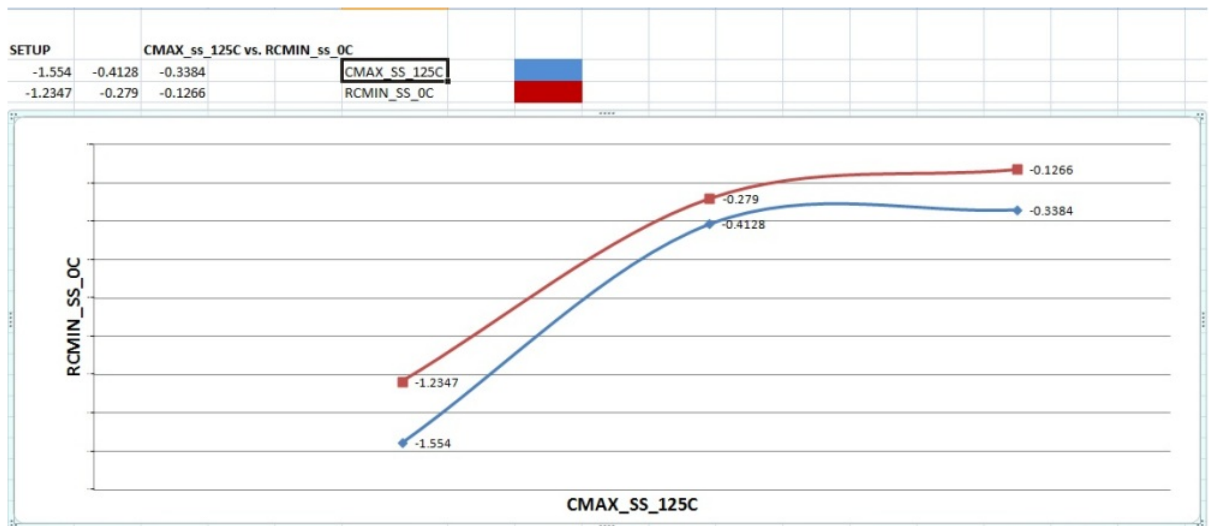


Figure 4.2: worst path/critical path for the setup

Shown below figure is the case of the worst path/critical path in the design for the hold, in other words its the worst negative slack (WNS)for the hold. RCMIN-SS-125C (means the corner(RCMIN), is it fast or the slow corner, if the ss than its slow nmos and slow pmos, if the ff than its fast nmos and fast pmos, if its fs than fast nmos and slow pmos and if the sf than slow nmos and fast pmos , and the last one is defining the temperature) corner contains the critical path for the setup and that is optimize with the other worst path(CMIN-SS-125C) with the same path that contain for the RCMIN-SS-125C.

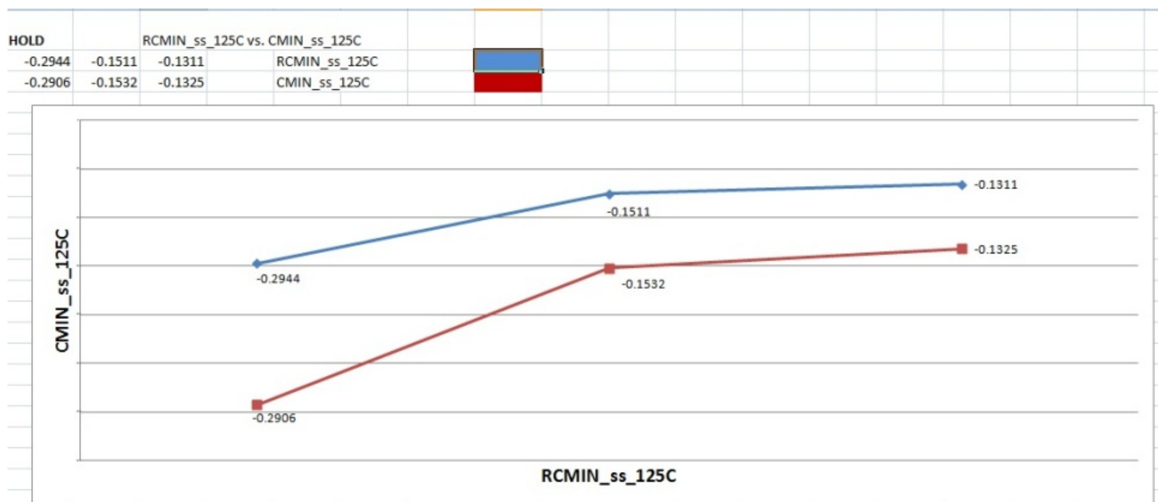


Figure 4.3: worst path/critical path for the hold

Here is the comparison for setup and hold worst slack between the SS(slow nmos and slow pmos) and FF(fast nmos and fast pmos) like, for the CMAX at 0C corner.

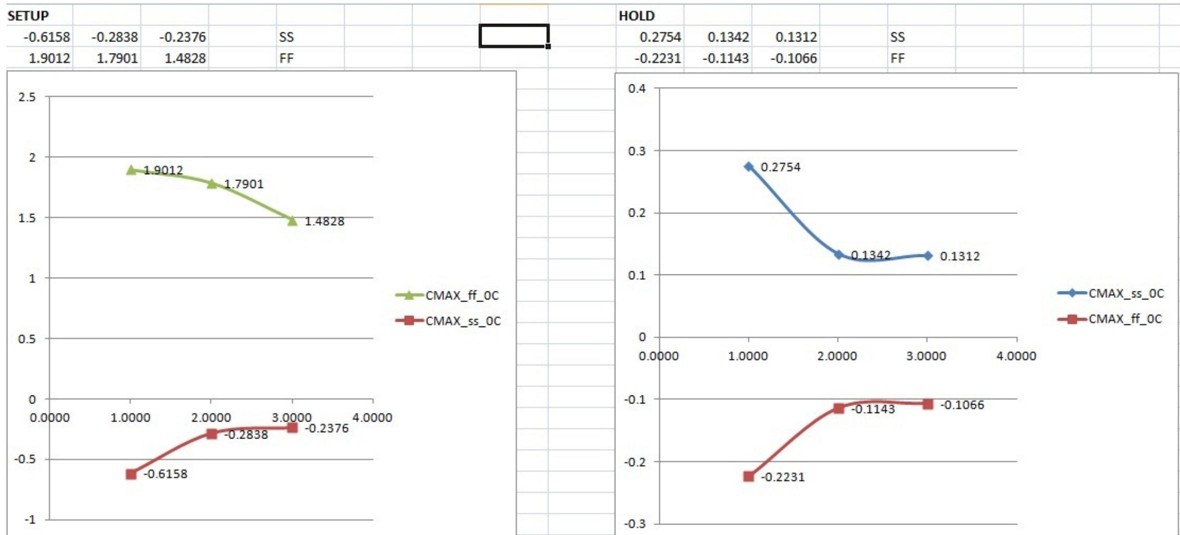


Figure 4.4: worst path/critical path for the setup and hold for CMAX-0C corner

Here is the comparison for setup and hold worst slack between the SS(slow nmos and slow pmos) and FF(fast nmos and fast pmos) like, for the CMAX at 125C corner.

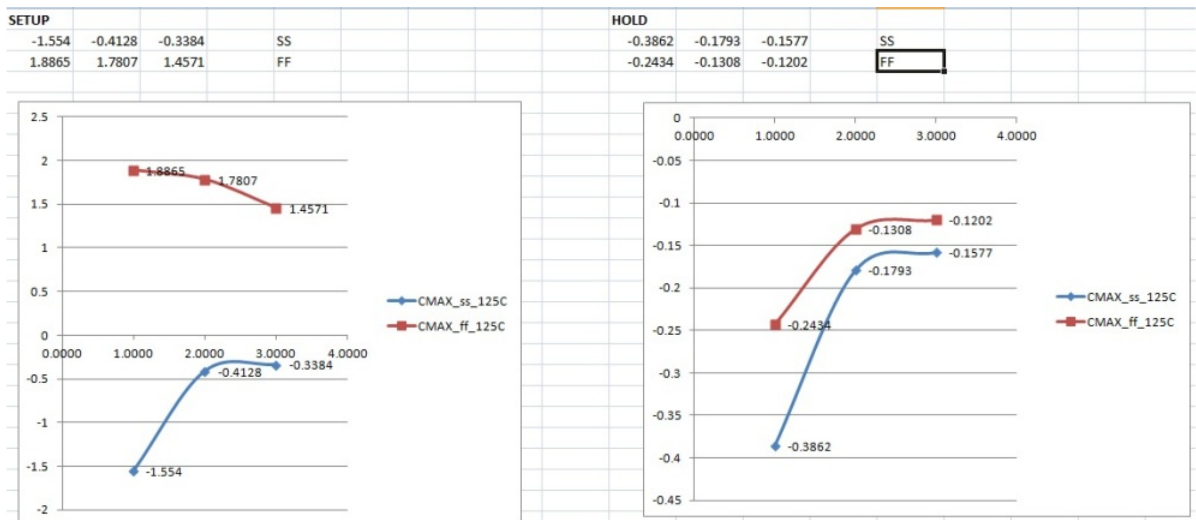


Figure 4.5: worst path/critical path for the setup and hold for CMAX-125C corner

In this project I had find timing analysis of 16 different corner's in the design. we have to find the some critical corners and that corners are optimize such a way that

all the corners covers in the critical corners. So, the benefit of this is to the next team (physical design (PD)) that Tool memories usage is less and Runtime will decreases drastically as a result of which PD team can do more iteration and do more analysis on this critical corners.

Project objective: plot the worst timing path for setup analysis and hold analysis for each timing corner to study the variation between different timing corners and reduce the timing corners for physical design flow.

References

[1] Reference Books

J Bhasker, Rakesh Chadha "Static Timing Analysis for Nanometer Designs"

David Harries, Neil H. E. Weste "CMOS VLSI Design"

[2] Reference Websites

http://solvnet.synopsys.com

http://vlsichipdesign.com

http://asic-world.com

[3] Referred Synopsys Manuals

Design Compiler Manual

Prime Time Manual

Scan Compression