SETUP & HOLD FIXING OF DIGITAL CIRCUITS WITH AUTOMATED HOLD FIXING USING MULTI SCENARIO ANALYSIS

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
[VLSI DESIGN & EMBEDDED SYSTEM]

Submitted by:
MANISH MITTAL
2K17/VLS/13

Under the supervision of
Mr. ALOK KUMAR SINGH
Associate Professor

ELECTRONICS & COMMUNICATION ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

JULY, 2019

ELECTRONICS & COMMUNICATION ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Manish Mittal, Roll No. 2K17/VLS/13, student of M.Tech (VLSI Design & Embedded System), hereby declare that the project Dissertation titled "Setup & Hold Fixing of Digital Circuits with Automated Hold Fixing Using Multi Scenario Analysis" which is submitted by me to the Department of Electronics & Communication Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associate-ship, Fellowship or other similar title or recognition.

Place: Delhi **MANISH MITTAL**

Date: 25th July, 2019

ELECTRONICS & COMMUNICATION ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "Setup & Hold Fixing of Digital Circuits with Automated Hold Fixing Using Multi Scenario Analysis" which is submitted by Manish Mittal, Roll No. 2K17/VLS/13, Electronics & Communication Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi
Date: 25th July, 2019

Mr. ALOK KUMAR SINGH
SUPERVISOR
ASSOCIATE PROFESSOR

ABSTRACT

SoC (System on Chip) signoff is considered as a critical part of the SoC design flow. From signoff perspective, an SoC has to pass through multiple criteria before releasing the final GDSII to the foundries for manufacturing Chips, such as Timing criteria, Phyical Verification (PV) criteria, Power Drawn Network (PDN) criteria, Formal Verification (FV) criteria and Conformal Low Power (CLP) criteria. These signoff checks along with many others are performed on an SoC. Timing checks are critical because if we intend to deliver an SoC operating at a desired frequency and speed, it's important that the chip meets its timing requirements along all the paths.

With continuously decreasing technology node, and increasing logic complexity within the chip, the number of scenarios required for timing analysis has also increased. An important stage in performing Static Timing Analysis of a chip is the Engineering Change Order (ECO), where the timing violations are fixed incrementally by giving feedbacks. Usually different paths fail under different conditions or scenarios, hence generating ECO's for each specific corner and analysing each corner is not recommended as it is time consuming and increases the cost of the chip. Hence Distributed Multi-Scenario Analysis (DMSA) is a feature provided by synopsys PrimeTime tool, which helps the user to simultaneously analyse violations across multiple corners and hence generate ECO's in a faster and more efficient way.

There are two kinds of timing violations that usually happens in a chip namely, setup violations and hold violations. Setup violations are frequency dependent, so they can be resolved by changing the operating frequency of the chip, but hold violations are frequency independent. Hence fixing hold violations require more number of ECO's to be generated.

In my report, I have discussed about an algorithm that efficiently generates Hold ECO's using PrimeTime tool using Distributed Multi-Scenario Analysis so that each and every violation that occurs across the chip across different scenarios can be captured and fixed.

ACKNOWLEDGEMENT

It gives me an immense pleasure to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed guide Mr. Alok kumar Singh, (Associate professor, ECE) for his valuable guidance, encouragement and help for completing this project work. His useful suggestions for this whole work and co-operative behavior are sincerely acknowledged.

I also wish to express my gratitude to Professor. Neeta Pandey for her constant support and guidance.

Mr. Puneet Dodeja, Team Manager & Technology Director, Qualcomm India, for accepting me as an intern in his team. I am grateful to him for his expertise, kind concern and continous encouragement.

Mr. Nitin Dhamija, Staff Engineer, Qualcomm India, for the long discussions that helped me in understanding the technical details of my work. I am also thankful to him for his unfailing support and for having faith in me.

At the end I would like to express my sincere thanks to all friends and others who helped me directly and indirectly during this project work.

MANISH MITTAL

2K17/VLS/13

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SoC – System on Chip

IP – Intellectual Properties

EDA – Electronic Design Automation

STA – Static Timing Analysis

RTL – Register transfer Level

GDS – Graphics data System

HDL - Hardware Description Language

DFT - Design for Testability

DRC – Design Rule Constraints

CMOS – Complementary Metal Oxide Semiconductor

OCV – On-chip Variations

AOCV – Advanced On-Chip Variations

POCV – Parametric On-chip Variations

PVT – Process Voltage Temperature

FEOL – Front End of Line

BEOL – Back End of Line

DTA – Dynamic Timing Analysis

ASIC – Application Specific Integrated Circuits

DMSA – Distributed Multi Scenario Analysis

SPEF - Standard Parasitic Exchange Format

PLL – Phase Locked Loop

ECO – Engineering Change Order

TCL – Tool command Language

Chapter 1

INTRODUCTION

Most of the System on Chip (SoC) vendor's today re-uses the Intellectual Property (IP) for designing of different SoC's. These Intellectual Properties (IP) are integrated together to obtain what we see as SoC. With times, not only does the chip density increases but also its complexity. With increasing number of transistors more functionality can be packed into the system and with the ever decreasing feature size the overall size keeps decreasing which leads to increase in the overall complexity of designing the chip. With such large number of logic placed, the overall interconnect parasitics have also increased manifold. Thus today's design methodology is more interconnect dominated rather than logic dominated, where large number of interconnect wires running close to each other introduces many undesirable effects in the design such as Crosstalk delay, Signal Integrity issues along with always increasing clock frequency which pose some serious challenges in Timing Signoff of the design.

Thus it has been noticed that the accurate timing closure of the design consumes a large part of the design time, thus special emphasis is always given to Timing requirements of the design right from the initial stages of the design stage. Timing closure is not only performed across single corner but a wide range of corners which accounts for variations in manufacturability which further increases the complexity. This is important because if Timing closure is not performed at different scenarios it leads to failure in later design stages which is difficult to debug and also quite expensive.

Designers try to reduce the number of Timing violations at the physical design stage, but still a large number of violations occur after the post route stage. Analysing and fixing these violations over the single scenario is not only time consuming but also irrevalant. Hence designers tend to fix these violations using Multi Scenario Analysis techniques provided by different EDA vendors through their Timing Signoff Tools

1.1 ASIC DESIGN FLOW

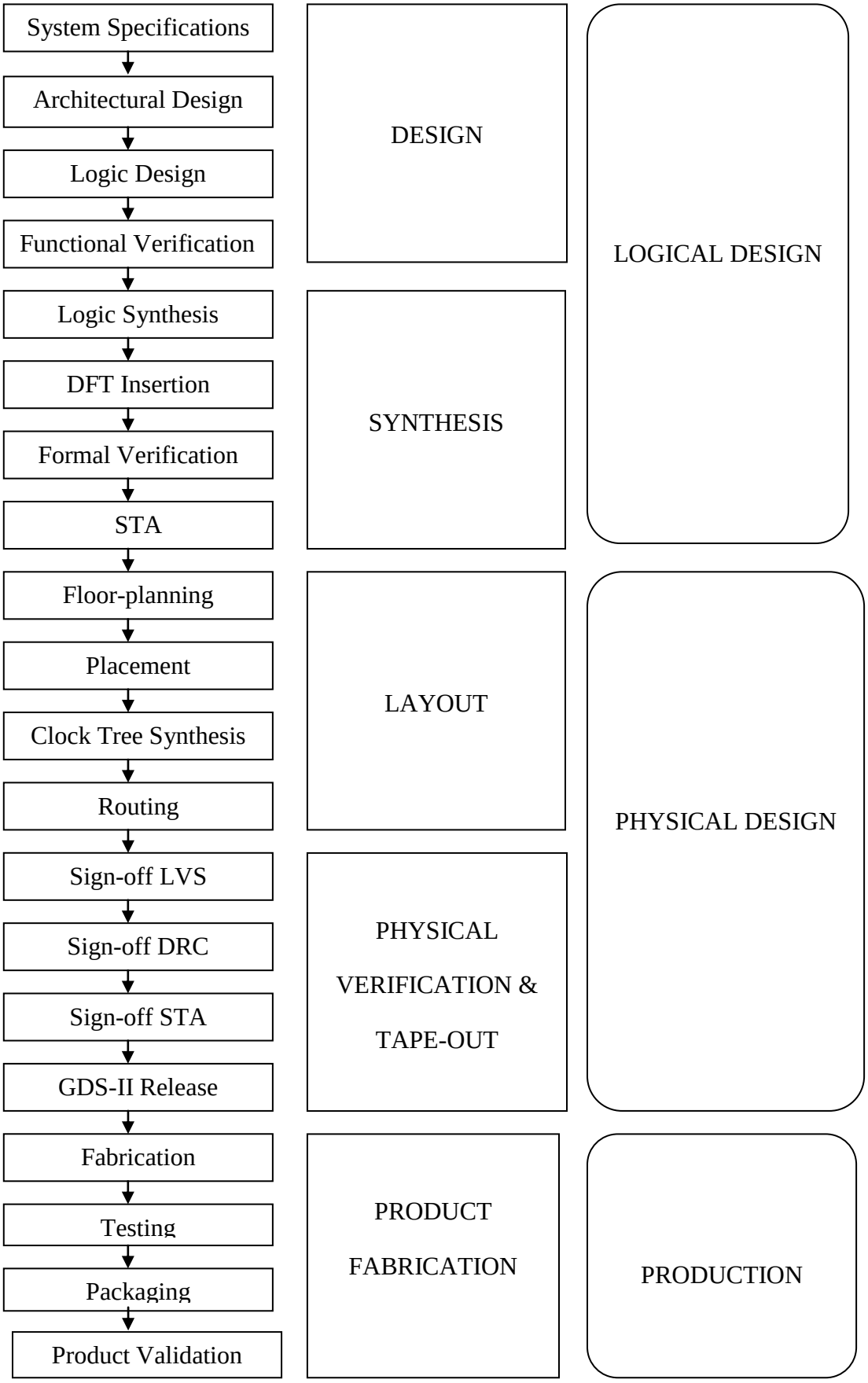

Figure 1.1 ASIC Design Flow

The flow is termed as RTL (Register Transfer Level) to GDS-II (Graphic Data System) flow and process to generate GDS-II is known as Tape-Out. The ASIC design flow is divided into Logical and Physical design flow i.e the Frontend and the Backend Process.

1.1.1 LOGICAL DESIGN

1.1.1.1 RTL DESIGN

The flow starts with High-Level design specification; the designer puts specification for Area, Speed and Power requirements. Then the designer starts setting Chip Architecture. RTL, Register Transfer Level, describing the functional behaviour using HDL, hardware description languages, Functional Verification, verifying the functionality using simulation.

1.1.1.2 SYNTHESIS

Synthesis, the first step of converting the RTL to gate stage netlist based on timing, power and area constraints**.** *DFT (Design for Testability),* this step is for preparing the design for testability. Scan insertion is a common technique that helps in making registers in the design that can be controlled and observed. *Equivalence Checking,* this step is for verifying the functionality of gate netlist against the RTL description using formal verification techniques. *STA,static timing analysis,* a method that checks the ability of the design whether it meets the timing requirements as required by the design statically without simulation.

The designer is responsible of specifying 'Timing Constraints' to model how the design needs to be constrained & the STA tools check that the design meets the timing requirements. The designer uses an industry standard format 'SDC' Synopsys Design Constraints. **STA on this stage acts as the connection between logical and physical design implementation.**

1.1.2 PHYSICAL DESIGN

1.1.2.1 LAYOUT

The flow starts with Floor planning; the logical blocks that are used in the design are placed considering many optimization factors to account for Area, Speed and Power. Then Placement occurs where the connections between blocks are routed. After Placement the next step is to build the Clock Tree Synthesis to distribute the clock and reduce clock skew that occurs in different parts of the design. Then routing the design is the final step to generate the layout. During the physical design implementation stage, STA flow is run multiple times to achieve a more correct timing analysis.

1.1.2.2 TAPEOUT

Two steps are needed to verify the layout

- LVS, Layout versus Netlist, matching the layout with the netlist generated after synthesis. DRC, Design Rule Checking, All rules that are being laid out by the semiconductor fabrication foundries where it will be fabricated into a chip are retained and followed.
- Signoff Static Timing Analysis is performed
  Finally, GDSII release, semiconductor foundries manufacture chips based on the GDSII.

1.2 TIMING CONSTRAINTS

From timing perspective, the designer creates timing constraints for synthesis which are a set of constraints or rules applied to a given collection of paths or nets that allows the chip to have the desired performance. Constraints can be any set of rules or definions about period, frequency, maximum and minimum delay between end points, or maximum or minimum net delay.

CONDITIONS THAT NEED TO BE MET:

- Clocks in the design
- Max allowed Transition time for clocks or signals
- Max allowed load or capacitance at pin or port
- Max allowed Delay for pin or port

BOUNDARY SETTINGS:

- Input transition Time for signals
- Output loading at the output
- Logic Settings for the design

EXCEPTIONS FOR THE SINGLE CYCLE PATH:

- False Path
- Multi-cycle path

1.3 INTER AND INTRA CHIP VARIATIONS

Regular scaling of the Complementary Metal Oxide Semiconductor (CMOS) devices have led to the increase in the number of device parameter variations such as gate oxide thickness, device channel length and width which in turn effects device performance. As the technology node decreases it becomes more complex to model these variations for efficient device manufacturing.

As an accepted notion in the semiconductor industry, where device geometry is continuously decreasing, the growing affect comes in the variations in static timing analysis. Signal integrity (SI), which was first noticed as a first order effect in 130-nm

process technology and then became more complicated over reducing geometry nodes, on-chip variations (OCV) which started at 130-nm and its effects are only increasing with decreasing geometry nodes. A starting solution to consider for OCV was to use a flat universal margin across the full chip. However, the increasing result of variation in modern designs and techniques requires a better and accurate device level variation methods.

On-chip variation (OCV), the present adopted model for variation in a designer's timing flow is the first stage of approach that applies a layer margin across the design. With more variations due to different parameters like process, voltage, and temperature, also due to increasing variations across the same die and in different dies, arriving at a single unique blanket margin is difficult. There are increasing concerns about OCV with respect to over-design, decreased design performance, and higher timing closing cycles which is Engineering Change order Phase. As time this was one of the most practical and safe way of applying the worst-case variation across the complete chip is now becoming less acceptable, design engineers have found ways to reduce and relax the OCV effects. Some important concerns such as "why do near cells see such a large difference in variation swing?" and "why do nets or cells in paths of different logic depths see the same variation?" this requires the need for improved and relaxed techniques and for the layer OCV technique to take a different form.

The AOCV method provides derates numbers as a function of logical depth. These variables provide more information and efficiency for margining approach by calculating how much a particular path is affected by the process variations.

Primetime Parametric on-chip Variation (POCV) is a method that reprsents the delay of a cell or an instance in the design as a variable that is unique to that instance.

POCV is a method developed by Synopsys which uses a statistical approach, which means it does not perform a full SSTA analysis of the design. Instead, what it does is it computes the delay changes by modelling the cell delay which are internal to design and driving load parasitic to calculate both the mean and sigma of a logic stage of the design.

Features of POCV

- POCV does Statistical derating of various parameters for variations
- POCV provides Single input format and it does characterization source for both the OCV's AOCV and POCV table data
- It provides Non-Statistical Reports for timing
- It does very Limited statistical timing reporting for paths
- POCV is Compatible with already existing Primetime functionality

POCV provides benefits over AOCV

- POCV Reduces Pessimism gap between two analysis types which is graph based analysis and the other is path based analysis.
- POCV provides Less overhead count for incremental or more timing analysis.

1.3.1 SOURCES OF VARIATIONS

1.3.1.1 SUPPLY VOLTAGE VARIATIONS

On a chip, each and every standard cell is connected to the supply voltage rails that is the VDD and GND rails through interconnects which have some finite amount of resistances. The length and width of these interconnects may be different which may result in varying resistances. These resistances result in voltage drop across the inter-connects, which results in incorrect voltage levels being received at the Standard cell pins. Along with the resistances, inductance effect also comes into picture which leads to further degradation of the voltage levels. Local effects like the On-chip variations causes various undesirable effects such as IR drop, Ground bounce, Slew variations, electro-migration. These effects have direct consequences on the delay characteristics of the cells which in turn alter the Timing closure of the design. We know that the speed of the circuit is directly related to the supply voltage which is supplied at its Supply voltage pins, hence lesser the supply voltage, slower is the circuit.

1.3.1.2 PROCESS VARIATIONS

Process Variations arises due to changes or variations in semiconductor fabrication process. Today, millions of instances are packed into single chip and the device geometries like the channel width, length, gate oxide thickness and many more are not identical for each and every instance. There are slight variations among instances on same chip and these variations needs to be accounted for desired performance of the design. Variations in device Geometries can lead to variations in critical parameters like the Threshold voltages which in turn will always affect the performance of the device. Process variations are caused due to variations in pressure, dopant concentrations

1.3.1.3 TEMPERATURE VARIATIONS

During the operation of a chip the temperature across it continuously varies. Due to continous switching activity, the dynamic and static power dissipation on the chip results in increasing temperature across the chip which than affects the desired efficiency of the chip. The temperature of the chip affects the threshold voltage or cutoff voltage of the device, increasing temperature decreases the threshold voltage, which in turn makes the device faster. But a higher increase in the temperature is undesirable for the overall performance of the device as it causes heating issues which in turn causes relaibility concerns for the chip.

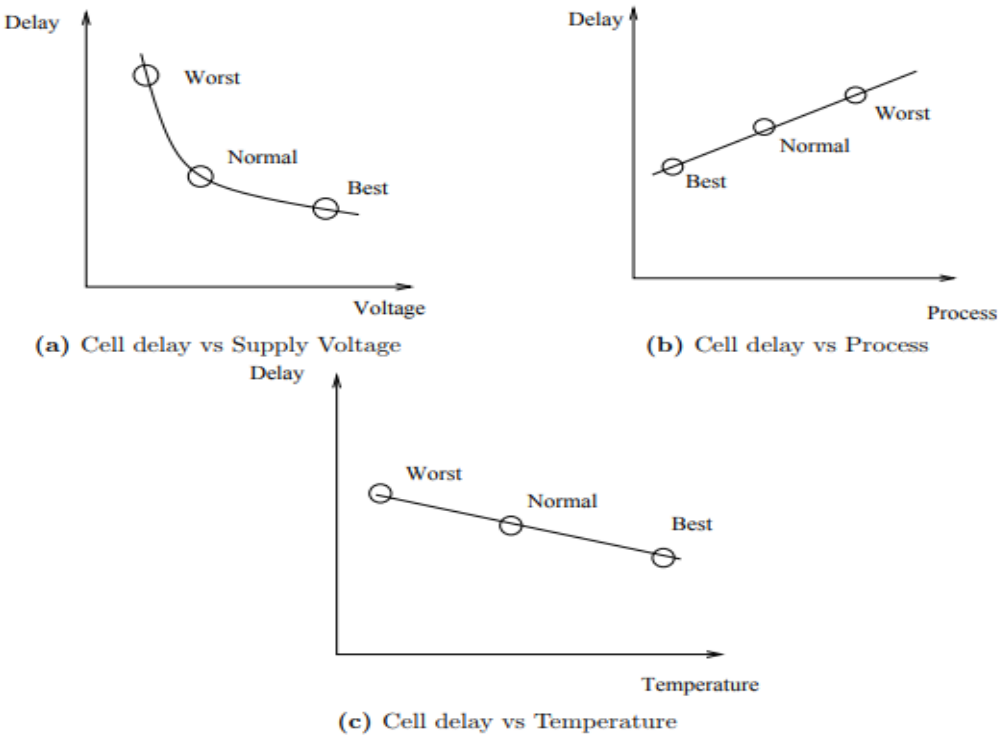

Figure 1.2: Cell delay trend with PVT variations [1]

1.4 SCENARIOS IN TIMING ANALYSIS

A scenario is a unique mix of operating conditions and operating modes for a given specifications. Scenarios are essentially required to ensure that the chip performs its intended operation under all possible conditions. Operating conditions denote various voltage levels at which the Chip is intended to work and different parasitic corners for worst and best delay calculations. The operating modes on the other hand represent different modes in which the chip is operated. There are widely two operating modes for SoC's

- Functional Modes
- Test Mode

1.4.1 OPERATING CONDITIONS

The operating conditions denote the variations due to three different parameters voltage, process and temperature. These variations are together denoted as Operating Conditions or also known as Corners. These are defined in the form of

library set which are provides in the Process kit based on the Technology by Foundries.

1.4.1.1 FRONT END OF LINE (FEOL)

Front End of Line (FEOL) denotes the lowest layer of IC Fabrication stage where the MOS/FINFET devices are fabricated. The variations in the device delays and other parameters with the variations in the PVT are denoted using the FEOL corners. On the basis of doping concentrations, carrier mobility usually three types of FEOL corners exist namely SLOW, FAST, and TYPICAL. FEOL corners are designated using two letter words; the first letter denotes the nMOS and the second letter denotes the pMOS. For example, for fast nMOS and fast pMOS the FEOL corner is deoted by **ff,** a fast nMOS and a slow pMOS is denoted by **fs.**

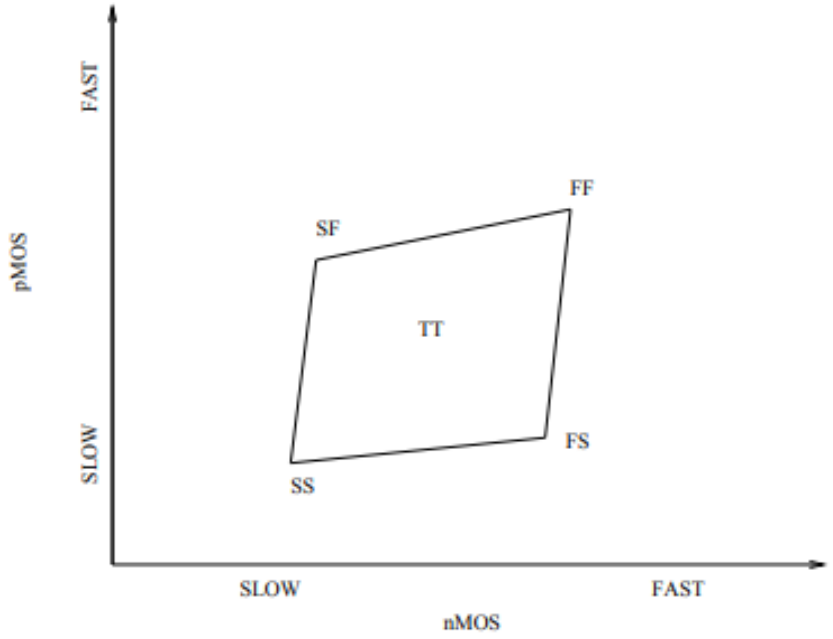

Figure 1.3: Design Corners [1]

1.4.1.2 BACK END OF LINE (BEOL)

The back end of line denotes the metal interconnects that are used for connectivity between different MOS structures. In today's VLSI world, the delay of these interconnects comprises a very important portion of the total delay of the circuit. Due to the On-Chip Variations as discussed earlier, the delay due to coupling effects caused because of the nearby interconnects is very useful in the calculation of the total

delay. With the increase in process complexity the number of BEOL has increased. Some of the BEOL corners for 8nm are:

- Capacitance Worst (cw)
- Capacitance Best (cb)
- Coupling Capacitance Worst (ccw)
- Coupling Capacitance Best (ccb)
- Resistance-Capacitance Worst (rcw)
- Resistance-Capacitance Best (rcb)

Chapter 2
OVERVIEW OF STATIC TIMING ANALYSIS

Timing analysis refers to analysing the design for timing violations. Dynamic Timing Analysis (DTA) and Static Timing Analysis (STA) are the two ways of performing timing analysis on a design. The former validates the design by verifying its functionality as well as the timing. This is done by simulating the design for various input stimulus. This process is extremely exhaustive as compared to static timing analysis, where the design is validated statistically for timing and does not depend on the input stimulus. Thus, for the timing analysis of Application-Specific Integrated Circuit (ASIC), containing millions of gates, STA is a faster and simpler approach. The main aim of static timing analysis is to make sure if the chip or design can be operated at rated speed without causing any timing issues.

Performing efficient Static timing analysis is a widely used technique of checking the timing behaviour of a design or circuit by checking and ensuring all the possible paths for timing are violations free. The approach is that PrimeTime tool breaks a design further down into different timing paths or timing arcs, it then calculates the propagation delay for the data signal along different and also each path, and finally checks for any violations of timing violations or design rules within the design.

2.1 VARIOUS TIMING PATHS IN DESIGN

PrimeTime tool when performing timing analysis does it by first breaking the design into several smaller timing paths and then perform analysis. Each timing path consists of the following elements:

- **Startpoint-** The start of a timing path where required signal data is launched or released at the active clock edge or where the data must be available at a specific time. A startpoint for timing analysis can be an input port of the design or a register clock pin.
- **Combinational Logic Network-** Elements that have no memory or internal state. Different combinational logic can be present along the data path such as Adders, Logic gates (OR, XOR, XNOR, MUX, AND, NAND), but none of the cells should be a sequential cell.
- **Endpoint-** The end of a timing path is where data is captured by an active clock edge when it has been released at the startpoint or where the data must be available at a specific time. An endpoint in the design can be either the input pin of the registers or an output port of the design.

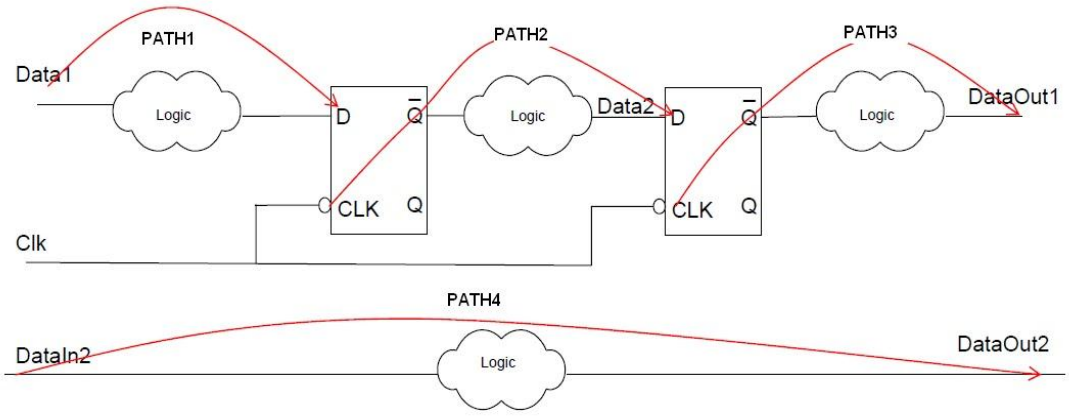

Figure 2.1: Timing Paths [2]

In the example, each logic block represents a combinational network having the data path delay.

PATH	STARTPOINT	ENDPOINT
Path 1	Input Port of design	Data input of data storing element
Path 2	Clock pin of data storing element	Data input of data storing element
Path 3	Clock pin of data storing element	Output Port of design
Path 4	Input Port of design	Output Port design

Table 2.1: Timing Paths

2.2 DELAY CALCULATION

The first step that tool performs is to break design into smaller paths, and then calculate delays of each cell and nets along each path and then report back the timing summary. The total accumulated delay along the path is the sum of its each cell delays and net delays.

- **Cell Delay**

  If there is a transition in the input signal of the cell, there will be corresponding transition in the output signal. The delay from this input transition to output transition of the cell is cell delay. If the standard delay files for the cell (SPEF/SDF) are not available, the tool than calculates the cell delay by viewing and applying values of delay from the lookup tables which are provided in the cell libraries.

- **Net Delay**

  The time that a data signal takes to traverse from the output pin of one cell till the input pin of next cell along the path is actually the net delay. The net delay is due to the different parasitic like the capacitance and resistance of the interconnecting wires present in between the two successive cells.

  The PrimeTime tool calculates net delays with the following methods:

  1. Estimate the delays using a different wire load models like Zero Wire Load Model; this method is used before layout, when the chip topography is unknown.
  2. Using delay values as provided by standard delay files.

2.3 CONSTRAINT CHECKS

Once different timing paths are known by PrimeTime and when it finds the path delays, it now analyses the paths for different timing violations, such as setup and hold constraints.

- A setup timing violation check makes sure that by what time the data signal should be made available at the data input pin of a sequential device like flip-flops or latches before the clock at the capture sequential block arrives. The setup constraint ensures a maximum delay on the data path.

- A hold check on the other hand ensures that by what time the data signal should be available before the capture edge at the capturing flip-flop arrives. The hold check is performed on the same edge of the clock.

PrimeTime not only checks for setup and hold violations but also it can check for clock gating checks, recovery and removal checks, and different Design rule violations for the clock like the minimum pulse width violations and minimum period violations for clock signals.

For timing violations, we introduce term called Slack, which denotes the amount of time by which a path has failed. For example, for a setup check, if the required time to reach at the input pin of the capture flop is 10ns, and the data signal arrives at 7ns, than in this case the slack is 3ns which is a positive quantity indicating the path is met. A slack of value 0 will show that the path is just barely met. A negative slack means that a timing violation has occured.

2.3.1 DIFFERENT TIMING CHECKS FOR FLIP-FLOP

The following example shows how PrimeTime tool evaluates the hold time and setup time checks for a flip-flop

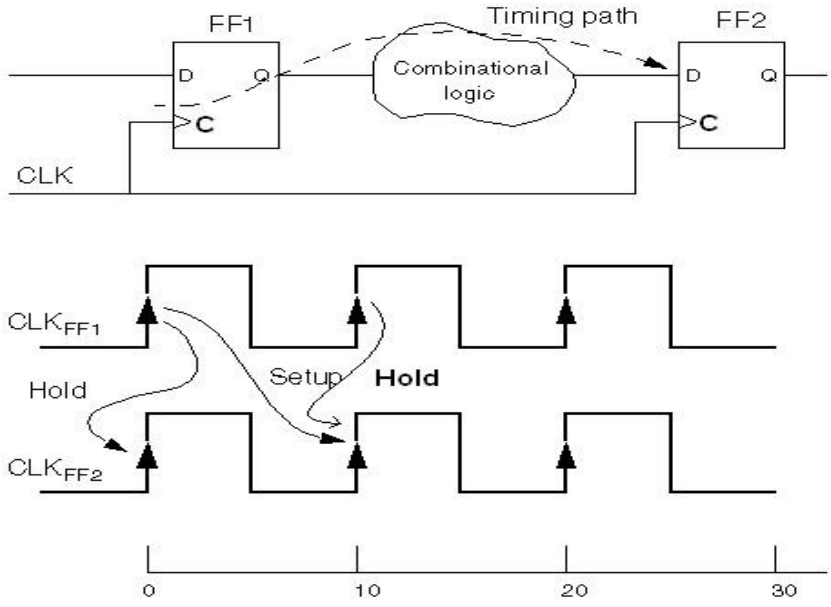

Figure 2.2: Setup and Hold check [2]

In the above example, consider the hold time and the setup time of the flip-flops to be 1 unit of time and 0 unit of time respectively, and the clock is defined in such a

way that its time period is 8 units of time. The units of time can be in either nanoseconds (ns) or picoseconds (ps), and these values are defined in the libraries.

PrimeTime checks for setup violations in one clock period. So, when the data is launched by FF1 at 0ns, the tool checks that the data should reach at the D pin of the FF2 atleast before required library setup time of the capture flip-flop FF2. If the data reached too delayed a timing violation is reported by the tool. For setup delay analysis, the tool considers the maximum values of each nets delay and maximum values of each cell delay along the data path.

For the hold delay calculation, the PrimeTime tool checks that the data launched at the clock edge of the launch register FF1 reaches the data pin D of the Capture registers FF2 atleast after library hold time of the register FF2. For hold checks, the tool considers the minimum delays of each nets and celss along the data path.

2.4 THE PRIMETIME STA ANALYSIS FLOW

Table 2.2 shows steps to perform PrimeTime static timing analysis:

STEP	TASK	TYPICAL COMMANDS
1	First Read the design data, which includes either a gate-level netlist or verilog and associated logic libraries	set search_path set link_path read_verilog link_design
2	Specify timing and design rule constraints	set_input_delay set_output_delay set_min_pulse_width set_max_capacitance set_min_capacitance set_max_fanout set_max_transition
3	Specify clock characteristics	create_clock set_clock_uncertainty set_clock_latency set_clock_transition
4	Specify different timing exceptions	set_multicycle_path

		set_false_path set_disbale_timing
5	Specify the analysis environment and different conditions such as operating mode conditions and delay models	set_operating_conditions set_driving_cells set_load set_wire_load_model
6	Specify case and mode analysis settings	set_case_analysis set_mode
7	Back-annotate delay and parasitics	read_sdf read_parasitics
8	Apply variation	read_ocvm set_aocvm_coefficient set_aocvm_table_group
9	Specify power information	load_upf create_power_domains create_supply_net create_supply_set create_supply_port connect_supply_net set_voltage
10	Specify options and data for signal integrity analysis	set si_enable_analysis true read_parasitics – keep_capacitive_coupling
11	Apply options for specific design techniques	set_latch_loop_breaker set_multi_input_switching_coefficient define_scaling_lib_group
12	Check the design data and analysis setup	check_timing check_constraints report_design report_port report_net report_clock report_path_group

		report_cell report_lib
13	Next we do a complete timing analysis and than analyse the results	report_global_timing report_timing report_constraints report_analysis_coverage
14	Generate the engineering change orders (ECOs) for either to fix different timing violations or recover power	set_eco_options fix_eco_drc fix_eco_timing fix_eco_power write_changes
15	Save the primetime session	Save_session

Table 2.2: PrimeTime STA Flow

2.5 WORKING WITH DESIGN DATA

2.5.1 LOGIC LIBRARIES

A logic library describes the timing and functions of macro cells in an ASIC technology by using the Library Compiler tool. Other Synopsys tools, such as the Design Compiler synthesis tool and the IC Compiler place-and route tool, also use these logic libraries.

A logic library contains library cell descriptions that include

- Cell, bus and pin structure that describes each cell's connection to the outside world
- Logic function of output pins of cells.
- Timing analysis and design optimization information, such as the pin-to-pin timing relationships, delay parameters, and timing constraints for sequential cells
- Other parameters that describe area, power, and design rule constraints

PrimeTime can read logic libraries in the .db and .lib formats. The libraries can have different units of time, capacitance, and voltage.

2.5.2 READING AND LINKING THE DESIGN

Before performing timing analysis, we need to read and link the design and logic libraries. PrimeTime can read the following file formats:

INPUT DATA	SUPPORTED FILE FORMATS
Design Data	Binary database (.db) Milkyway Synopsys logicsl database (.ddc) verilog VHDL
Logic libraries	Binary Database (.db) Synopsys Library Compiler format (.lib)

Table 2.3: Different Input File Formats for PrimeTime

To read and link the design data,

1.) Specify the directories in which PrimeTime searches for designs, logic libraries, and other design data such as timing models. To do this, set the search_path variable. For example

set_app_var search_path "./abc/design /abc/libs"

PrimeTime searches the directories in the order that is specified.

2.) Specify the libraries in which PrimeTime finds elements in the design hierarchy by setting the link_path variable. For example:

set_app_var link_path "* STDLIB.db"

The variable can contain an asterisk (*), library names, and file names. The asterisk instructs PrimeTime to search for a design in memory. PrimeTime searches libraries in the order that is specified. The first library in the path is usually considered as the main library.

3.) Read the design into memory

read_verilog TOP.v

If the search path includes files that contain the subdesigns, we need to read only the top-level design.

4.) Link the design to resolve References to library cells and subdesigns:

link_design TOP

During the design linking, the tool automatically loads the sub-designs if the subdesign names match the file names.

2.5.3 WORKING WITH DESIGN OBJECTS

Designs are hierarchical entities composed of objects such as cells, ports, and nets.

In PrimeTime tool, a design contains the objects in the following table:

OBJECT CLASS	DESCRIPTION	COMMAND
cell	Instance in the design; can be a hierarchical block or primitive library cell	get_cells
clock	Clock	get_clocks
design	Design	get_designs
lib	Library	get_libs
Lib_cell	Cell in a logic library	get_lib_cells
Lib_pin	Pin on a library cell	get_lib_pins
Lib_timing_arc	Timing arc on a library cell	get_lib_timing_arcs
Net	Net in the current design	get_nets
Path_group	Group of paths for cost-function calculations and timing reports	get_path_groups
Pin	Pin of lower level cell in design	get_pins
Port	Port of current design	get_ports
Timing_arc	Timing arc	get_timing_arcs
Timing_path	Timing path	get_timing_paths

Table 2.4: Design Hierarchy in PrimeTime

To constrain the design, we need to perform detailed timing analysis, and locate the source of timing problems; we need to access the design objects. We can do this by creating collection with the appropriate “get” command. For example, the get_ports command creates a collection of ports.

2.5.4 WORKING WITH ATTRIBUTES

An attribute is a string or value associated with an object that carries some information about that object. We can write programs in TCL to get attribute information from the design database and generate custom reports on the design.

PrimeTime provides the following commands for setting, reporting, listing and creating attributes.

ATTRIBUTE COMMAND	DESCRIPTION
define_user_attribute	Creates a new attribute for one or more object classes
get_attribute	Retrieves the value of any attribute from a single object
list_attributes	Shows attributes defined for object class
remove_user_attribute	Removes a user defined attribute from one or more object
report_attribute	Displays the value of all attributes on one or more object
set_user_attribute	Sets a user defined attribute on one or more attribute

Table 2.5: PrimeTime Attribute Commands

2.6 CONSTRAINING THE DESIGN

One of the most important criteria that needs to be fulfilled before we can perform timing analysis of a design is to apply relevant timing constraints in the design. Timing constraints usually apply various restrictions for a data signal or clock signal to arrive at a input pin of the device or gates or to be relevant at a device output.

- For Input delay: set_input_delay
- For Output delay: set_output_delay
- To define a new clock: create_clock

2.6.1 INPUT DELAYS

To do constraint checking at the inputs of the design, the tool needs information about the signal arrival times of various inputs pins or ports of the design. The **set_input_delay** command is used to denote the time taken for the external paths to the port of the Design under test. The tool than will make use of this delay information to find the timing violations at the input port and in the transitive fanout from that input port. With this command we than specify the minimum value of delay and maximum

delay from arrival of active clock signal to the data signal arrival at the design input port.

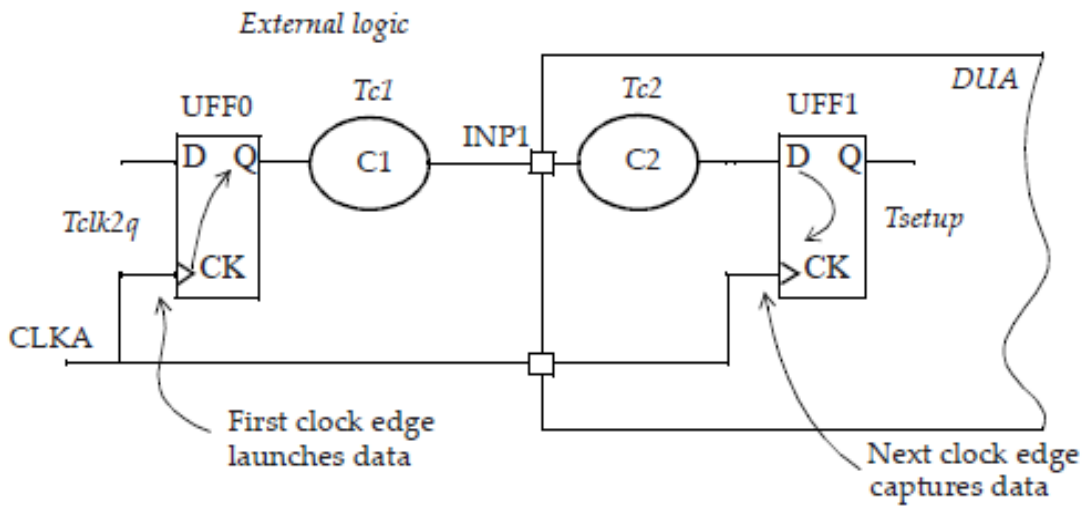

Figure 2.3: Input Delay [2]

In the above example the timing analysis path from register UFF0 to register UFF1, the path from UFF0 till INP1 is external to the Design Under Analysis (DUA). The delay combined from Clock to q delay (Tclk2q) of UFF0 and Combinational Logic delay (Tc1) is provided as an input delay at the INP1 port of the DUA. The Physical Design Tool when optimizing the design takes this input delay into consideration and optimizes the delay from port INP1 till D pin of UFF1 in such a way that it meets the path within required time interval.

2.6.2 OUTPUT DELAYS

To do constraint checking at the outputs of the design, we need to provide data about the timing requirements at the outputs to the PrimeTime tool. To specify the delay of an output port to a register, use the **set_output_delay** command.

With this command, we specify minimum amount of delay and maximum delay between the output external port of the Design under Test (DUA) and external sequential device that captures data from that output port. This setting will ensure at what time which data signals should be made ready at the output port of the design under test to meet the timing requirements for setup and hold requirements for the external sequential element:

- Maximum_output_delay = length_of_longest_path_to_register_data_pin + library_setup_time_of_the_register or latch.
- Minimum_output_delay = length_of_shortest_path_to_register/latch_data_pin – hold_time

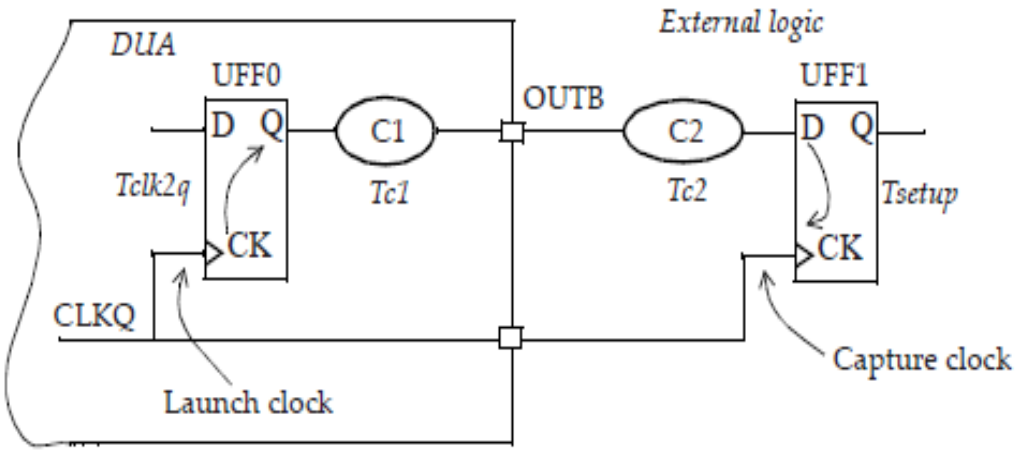

Figure 2.4: Output Delay [2]

In the Figure 2.4, for the timing path from UFF0 to UFF1, the path from OUTB till D pin of UFF1 is outside of Design Under Analysis (DUA). The summation of delay of combinational logic logic block Tc2 and required library setup time of UFF1 Tsetup comprises the external output delay. This obtained value of the net output delay is than provided at the OUTB port of the DUA. The Physical design tool when optimizing the path for timing violations will take into consideration this value so that it can optimize the path from CK pin of UFF0 till OUTB port.

2.6.3 DESIGN RULE CONSTRAINTS

PrimeTime checks for violations of design rule constraints that are defined in the library or by PrimeTime commands. Rules for DRC checks include

- Maximum limit for transition time of the signals
- Maximum and minimum limits of capacitance
- Maximum limit of fanout

To report design rule constraint violations in a design, use the **report_constraint** command.

2.7 CLOCKS

PrimeTime supports the following clocks:

- **Multiple Clocks**

  In a design we have many clocks that are different from each other and have have different time periods and also have different waveforms and different rise and fall edges. A virtual clock will have no real or actual source in the given design itself.

- **Skew and Delays of Clock Signals**

  Clocks when start from their sources arrive at the clock pins of their respective flops of latches with different delays due to various delay factors such as clock latencies (clock source latency and clock network latency), denoted using the clock skew. For multi-clock designs, we can specify inter-clock skew. We can specify an ideal delay of the clock network for analysis before clock tree generation, or we can specify that the delay needs to be computed by PrimeTime for analysis after clock tree generation. PrimeTime also supports checking the minimum pulse width along a clock network.

- **Gated Clocks**

  We can analyse a design that has gated clocks.A gated clock is one which passes through some gating logic elements such as multiplexers. PrimeTime tool will check for setup and hold violations on the gating signal.

- **Generated Clocks**

  We can analyse a design that has generated clocks. Clocks which are Generated are ones which are generated from already exiting clocks, such as a clock which is multiplied or divided by some amount to generate a new clock, but having the same source but different frequencies.

- **Clock Transition Time**

  We can specify the transition times of clock signals at register clock pins. The transition time of signal is defined as the amount of time a particular signal takes to change its logic state.

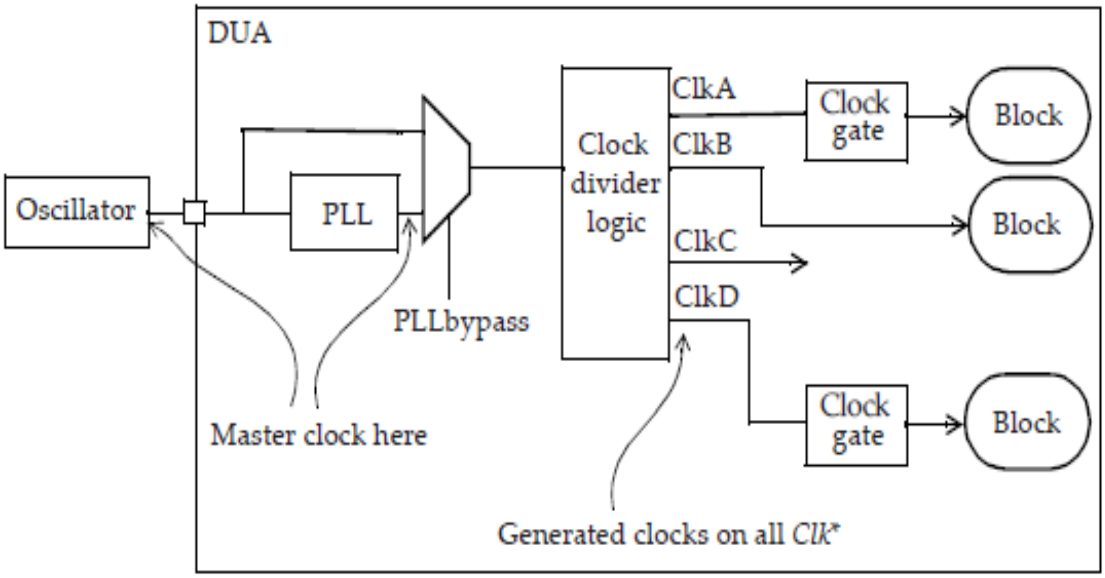

Figure 2.5: Clock Distribution in Typical ASIC [2]

2.7.1 SPECIFYING CLOCKS

Multiple Clocks are defined for a system. We need to define each clock. The details for clock signals include:

- Clock Period and waveform for different scenarios
- Latency (insertion delay) of clock signal across scenarios
- Uncertainty (skew) value of the clock signals
- Divided or multiplied clocks
- Different Clock gating checks to be performed
- Different transition times for complete or incomplete clock networks

PrimeTime analyzes paths between different registers or between register and ports. For a design with multiple interacting clocks, PrimeTime determines various phase connections among the clocks at the starting flops and clocks at the endpoint flops.

2.7.2 DEFINING CLOCKS

In order to determine and create any clock for the design, we use the **create_clock** command. This command creates a clock at the specified source. A source for the clock can be either at an input port of our design under test or an internal pin of any sequential block.

A clock when created with the create_clock command has an ideal waveform. After we create the clock, we must either propagate the clocks or define different parameters of the clock to perform correct timing analysis.

As soon as we define the create_clock command, it creates a path group which has the same name as that of the clock. This group contains all paths ending at points clocked by this clock.

We can define:

create_clock –period 12 –waveform {0.0 6.0} {ck1 ck2}

PrimeTime supports analysing multiple clocks propagated to a single register.

2.7.3 CREATING VIRTUAL CLOCK

The same create_clock command is used to define the virtual clock, these clocks are used for clocking external devices. A virtual clock has no actual source in the current design, but we can use it for setting input or output delays.

To create a virtual clock named virtual_clock1, we use

create_clock –period 3 –name virtual_clock1 –waveform {0.5 1.5}

2.7.4 APPLYING COMMANDS TO ALL CLOCKS

The get_clocks command selects clocks for a command to use, for example, to ensure that a command works on the CLK clock and not on the CLK port.

In order to get different properties of clocks having names which starts with with CLK1 and a period greater than or equal to 10.0 :

report_clock [get_clocks –filter "period >= 10.0" CLK1*]

The **all_clocks** command is equivalent to the PrimeTime **get_clocks *** command. The all_clocks command returns a token representing a collection of clock objects. The actual clock names are not printed.

The **remove_clock** command is used to remove all the clocks in the design.

To remove all clocks with names that start with CLKB, enter

remove_clock [get_clocks CLKB*]

2.7.5 SPECIFYING CLOCK CHARACTERISTICS

When we define a clock using **create_clock** command, the clocks that are created are ideal, means they do not have any delay values associated with them nor any parasitic. In order to define real clocks, we must either propagate the clocks, or define

certain characteristics with respect to the clocks. The important parameters of a clock are clocks latency (either source latency or network latency or both) and clock uncertainty values.

Clock Latency values usually consists of either clock source latency or clock network latency or a summation of both the latencies. The source latency for a clock is usually the time clock signal takes to travel from its actual point of origin (can be output port of Root Clock Generator RCG, or PLL) to clock definition point in the design under test. On the other hand, network latency for a clock is the time a clock signal takes to travel from the clock defined point till the clock pin of either register or latches in the design.

The difference in the arrival times of the clocks at different flops along the path leads to Clock Skew or also known as clock uncertainties. Further effects of on chip variations on the design leads to vaiations in the actual arrival of the clock signals at the endpoints, leading to more stringent requirements for different timing checks either setup or hold checks.

- **Setting Clock Latency**

  There are two ways by which we can define clock latencies in our design. We can either;

  - The first method requires the clocks in the design to be actually propagated across the entire design, so that the tool can calculate the delays accurately. But this method is only useful after the Clock Tree has been built in the design using the physical design flow.
  - The other method to define latencies for clocks in the design to explicitly apply certain delay values at different pins or ports of the design, so that the tool when performing timing analysis takes into account these delay values and calculates the actual delay. This method is not very accurate, hence used only during placement stage, when clock tree is not built.

- **Specifying Clock Source Latency**

Clock source latency can be applied for either ideal clocks or even propagated clocks in the design. Therefore the delay at a register clock pin is actually the sum of both clock source latency and clock network latency.

In order to apply source latencies for clocks we use:

set_clock_latency –source

The –**source** switch here indicates that the delay value is to be accounted for the source delay and not the network delay.

The –**early and –late** switches along with the **set_clock_latency** command can be also be used for to indicate the external uncertainties values for the clocks.

For defining clock network latency, the –**source** switch is dropped.

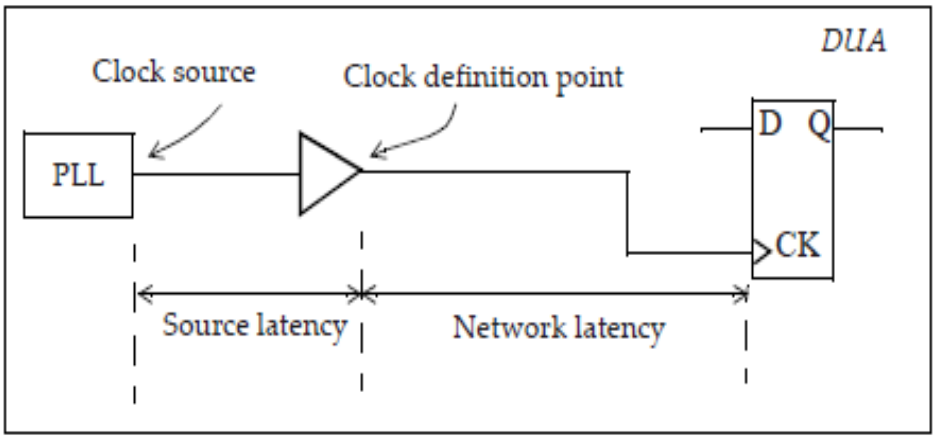

Figure 2.6: On chip Clock Source [2]

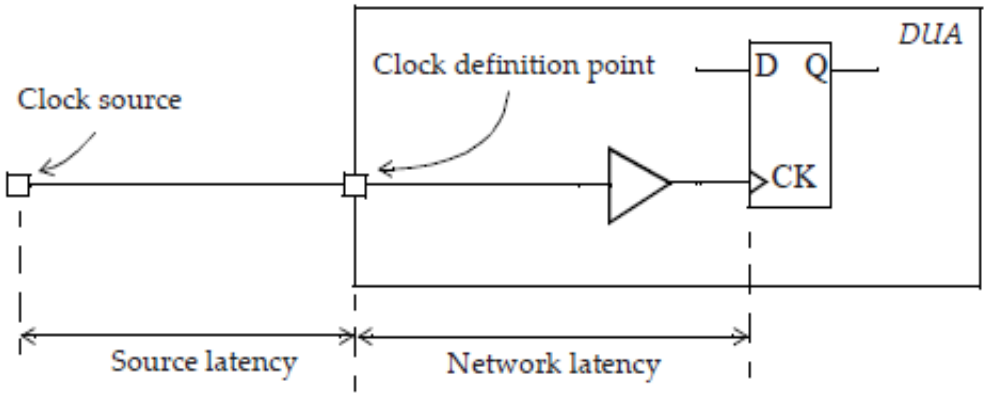

Figure 2.7: Off-chip Clock Source [2]

- **Dynamic Effects of Clock Latency**

Dynamic effects on the clock source latency, such as phase-locked loop (PLL) clock jitter can be modelled using the –dynamic option in the **set_clock_latency**. This option allows specifying a dynamic component of the clock source latency. Clock

reconvergence pessimism remobal (CRPR) handles the dynamic component of clock latency in the same way as it handles the PrimeTime Signal Integrity delta delays.

We can model clock jitter using **set_clock_uncertainty** command. However, the clock uncertainty settings do not affect the calculation of crosstalk arrival windows and are not considered by CRPR. The **set_clock_latency** command allows us to specify clock jitter as dynamic source clock latency.

- **Setting Clock Uncertainty**

  Clock uncertainties are applied to clock signals to account for extra pessimism. The edges of clock signals do not always arrive at the specified time instant, but infact due to various factors the edges of clock signals arrive either early or late than the specified time instant. For hold checks, the clock uncertainty values are added to the total delay, whereas for setup checks the uncertainties are subtracted. By default if nothing is mentioned, the tool picks the uncertainty value for both setup and hold.

To apply different uncertainty values of different clocks we use the **set_clock_uncertainty** command.

For clock uncertainty, we can specify these values for either same clock domain or different clock domains. For different clock domains we can specify Inter clock uncertainties, and for applying uncertainties values among similar clock domains we use intra clock uncertainties.

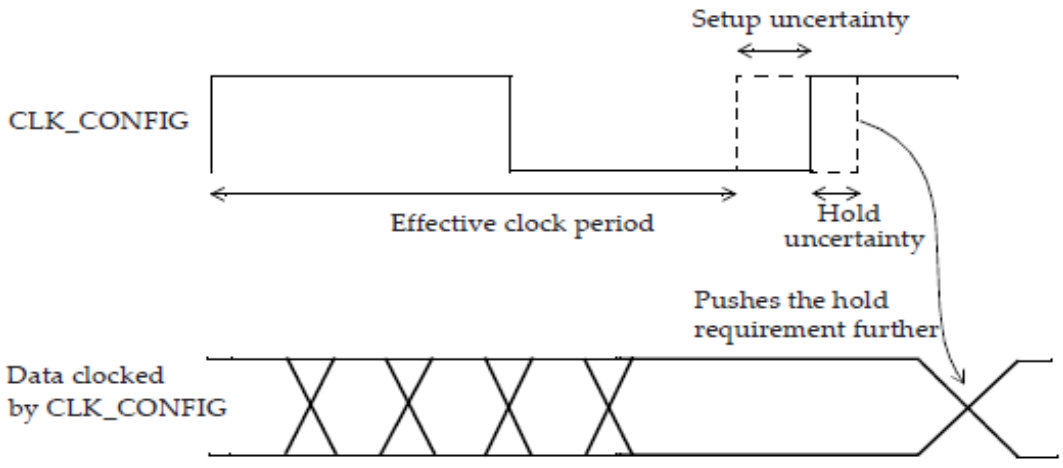

Figure 2.8: Specifying Clock Uncertainty [2]

- **Setting Clock Jitter**

To set clock jitter on a master clock, we run the **set_clock_jitter** command. This command sets the same clock jitter properties on all clocks generated from the specified master clock. If we do not specify a master clock, the command sets the jitter on all clocks.

To remove the clock jitter, run the **remove_clock_jitter** command. This command automatically removes the clock jitter properties from the generated clock as well. If we do not specify a master clock, the command removes the jitter from all clocks.

To report the clock jitter, run the **report_clock_jitter** command. The report shows the cycle jitter, duty cycle jitter and the master clock with jitter that is used by the generated clocks.

2.7.6 USING DIFFERENT CLOCKS

Today's SoC's are much complex than they were few years back, as a result of which multiple clocks are needed to be defined for a single design, each clock used for specific operation. Many of such clocks can be related to each other and their relationship usually depends on the way they are generated. Two clocks can be synchronous, asynchronous, or exclusive.

2.7.6.1 CLOCKS WHICH ARE SYNCHRONOUS

Clocks are said to be synchronous, if they share same source from which they are generated and also have same or related phases. The tool assumes the clocks that complete a timing path to be synchronous by default until and unless as specified by the users explicitly.

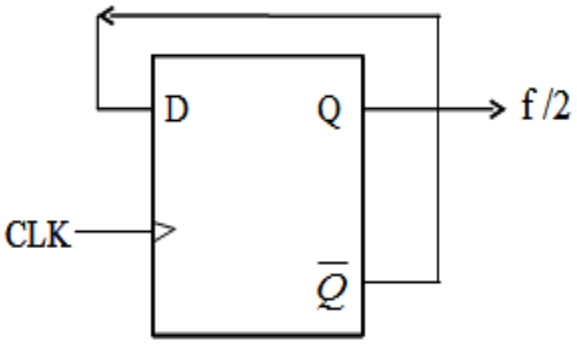

Figure 2.9: Synchronous clock Generation [2]

create_clock –period 6 –name CLK [get_ports CLK]

create_generated_clock –name CLK_OUT –source [get_ports CLK]

divide_by 2 [get_pins UFF0/Q]

A design might have various synchronous clocks defined in its database, and there will be many paths which have its launch clock from one clock domain and capture clock from another clock domain. To perform timing analysis for such paths, primetime will expand the clocks to the least common multiple of all the clocks in the paths and perform timing checks based on the expanded clock period.

2.7.6.2 CLOCKS WHICH ARE ASYNCHRONOUS

Clocks which do not have their sources common or which does not have any phase relationship among them are termed as Asynchronous clocks. Clocks are asynchronous when they are generated from different sources, like one clock is generated from PLL located outside the chip, whereas other chip is generated within the chip.

If we define two domains of clocks to be asynchronous, the tool will not perform any timing analysis on the paths that have these clocks as launch clocks and capture clock. In addition, if you are doing crosstalk analysis, PrimeTime SI assigns infinite arrival windows to the nets in aggressor-victim relationship between the two clock domains.

2.7.6.3 CLOCKS WHICH ARE EXCLUSIVE

A design might have clocks that are exclusive of each other. This means that ther may be multiple clocks defined on a particular pin or port of the design, but based on the either operating modes or conditions, only one of the clocks is enabled for analysis. Example, clocks which are defined for Test mode are disabled during the Functional mode of operation.

By default, PrimeTime tool analysis the paths between clocks which are mutually exclusive. Inorder to reduce extra effort of the tool , we can either declare such paths as false paths or use the **set_clock_groups –logically_exclusive** command.

To declare clocks CLKM1 and CLKM2 to be exclusive:

set_clock_groups –logically_exclusive –group {CLKM1} –group {CLKM2}

Using the above command, will force the PrimeTime to ignore or disable the paths that originate from CLKM1 domain clock and end at CLKM2 clock domain or

vice-versa. In order to find different clock groups that are defined in the design, we use the **–group** command. We avoid setting false paths between clock domains that have been declared to be exclusive because doing so is redundant.

The **set_clock_groups –asynchronous** command will actually create a group of clocks that are actually asynchronous in the design with respect to each other. Asynchronous clock group assignments are separate from exclusive clock groups assignments, even though both types of clock groups are defined with the **set_clock_groups** command. When we define clock groups (which gets defined automatically when we use **create_clocks)** the clocks can be either logically exclusive or even physically exclusive. Clocks are logically exclusive due to multiplexing and physically exclusive when they are physically separated. For clocks that are physically exclusive, the crosstalk phenomenon between two different signal lines cannot occur, so in that case we use the –**physically_exclusive** option rather than the –**logically_exclusive** option. This way we can prevents the tool from performing unnecessary crosstalk analysis between the clock nets.

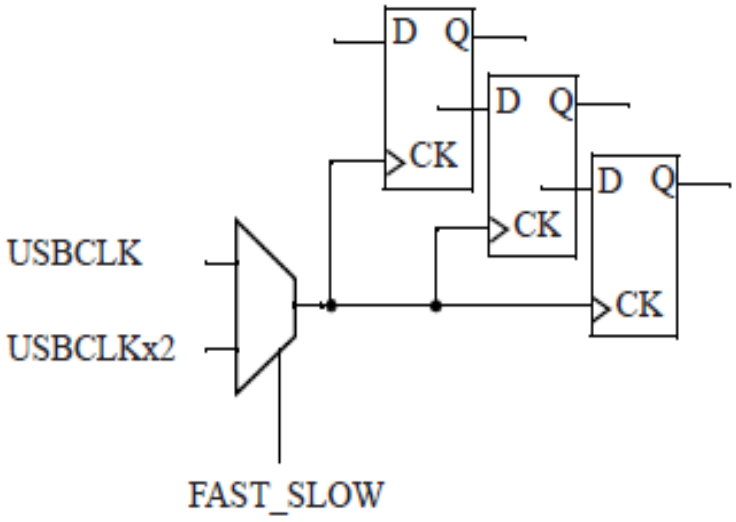

Figure 2.10: Mutually Exclusive Clocks [2]

2.7.7 SPECIFYING CLOCK GATING SETUP AND HOLD CHECKS

We know that the clock networks are the ones having most switching activity on them, hence the power consumption of clock networks is also high, in order to reduce the power consumption of clock networks, a very efficient technique of clock gating is used. In clock gating the clock signal is usually passed through some combinational logic other than normal inverters and buffers, such as AND gate or Multiplexers.

PrimeTime automatically performs timing checks for setup and hold violations on clock gating inputs of logic cells. This check is performed only for combinational gates where one signal is a clock that can be propagated through the gate, and the gating signal is not a clock.

2.8 TIMING PATHS AND EXCEPTIONS

2.8.1 TIMING PATH GROUPS

PrimeTime organizes paths into groups. This path grouping can affect the generation of timing analysis reports. For example, the **report_timing** command, when used with the **–group** option, this switch will report the worst of all the paths in each of the listed path groups.

In Design Compiler, path grouping also affects design optimizations. Each path group can be assigned a weight (also called cost function). The higher the weight, the more effort design compiler uses to optimise the paths in those groups. We can assign weights to path groups in PrimeTime, but this weight information is not used in PrimeTime. PrimeTime implicitly creates a path group each time we create a new clock with the **create_clock** command; hence The name of the path group is the same as the clock name. PrimeTime assigns a path to that path group if the endpoint of the path is a Flip-Flop clocked by that clock. PrimeTime also creates the following path groups implicitly:

- ****clock_gating_default**** - The group of paths that end on combinational elements used for clock gating.
- ****async_default**** - The group of paths that end on asynchronous preset/clear inputs of flip-flops.
- ****default**** - The group of constrained paths that do not fall into any of the other implicit categories; for example, a path that ends on an output port.

In addition to these implicit path groups, we can create our own user-defined path groups with the help of **group_path** command. This command also lets you assign any particular path to a specific path group.

Unconstrained paths do not belong to any path group. To report unconstrained paths, set the **timing_report_unconstrained_paths** variable to true. The **report_timing** command reports unconstrained paths as belonging to a path group called “(none)”.

2.8.2 SPECIFYING TIMING PATHS

The **report_timing** command, the timing exceptions command (such as **set_false_path**), and several other commands allow a variety of methods to specify a single path or multiple paths for a timing report or for applying timing exceptions. One way is to explicitly specify the **–from** $startpoint and **-to** $endpoint options in the **report_timing** command for the path.

For a timing path, its start point can either be the clock pin of a sequential block, or the input port of the Design under test, or even the D pin of a Latch. Similarly the endpoints for a timing path can either be the D pin of the sequential block, or the output port of the Design Under Test, or the D pin of other latch.

PrimeTime also supports a special form where the startpoint or endpoint becomes a **–through** pin, and a clock object becomes the **–from** or **–to** object. We can use this method with all valid startpoint and endpoint types, such as input ports, output ports, clock flip-flop pins, data flip-flop pins, or clock-gating check pins.

2.8.3 OVERVIEW OF TIMING EXCEPTIONS

Timing Exception	Command	Description
False path	set_false_path	This command Prevents analysis of the specified path. Extra runtime is saved
Minimum and maximum path delays	set_max_delay, set_min_delay	This command will over write the setup and hold values with the ones specified in the command.
Multicycle path	set_multicycle_path	This command basically specifies how many clock cycles are required for data propagation from startpoint till endpoint.

Table 2.6: PrimeTime Timing Exceptions

Each of the above mentioned timing exception can be applied to either a single path or a group of paths that belong to a particular clock group, or even to paths that pass through some points.

For the given design, if we wish to view all the timing exceptions that has been applied to the design we can use the **report_exception** command.

2.8.4 FALSE PATHS

False paths are those paths which are physicslly and logically present in the design, but these paths should not be analysed for timing violations. For example, a path can be present between two multiplexers, such that the select pins of each multiplexer receives opposite signals with respect to each other, in such a scenario, if the signal is propagated from A0 pin of MUX1, than automatically signal from the A1 pin of MUX 2 will be transferred and vice-versa.

For example, to define false paths between two pins, MUX1/A0 to pin MUX2/A0:

set_false_path –from [get_pins MUX1/A0] –to [get_pins MUX2/A0]

Once we declare a path to be False, all the timing related constraints that has been applied to the path will be removed. PrimeTime will still perform the calculation of this path delays, but PrimeTime will not report if any timing violations occur on this path, it to be an error, even if the delay calculated is too large or short.

If we wish to declare false paths for all the paths that are formed between two different clock domains, we can use a set of two commands:

set_false_path –from [get_clocks clock_main1] –to [get_clocks clock_main2]

set_false_path –from [get_clocks clock_main2] –to [get_clocks clock_main1]

2.8.5 MULTI-CYCLE PATHS

We use the **set_multicycle_path** command, if we wish to indicate that a particular path takes more than one cycle to complete its timing data transfer. Such paths are long paths, which cannot complete its timing checks in a single cycle of clock, hence these paths require multi-cycle exceptions to be applied to them. PrimeTime will than calculate the setup or hold requirements of the paths based on the specified number of multi-cycles.

In the figure 2.11, the path from UFF0 to UFF1 is designed to take three clock cycles rather than one.

set_multicycle_path 3 –setup –from [get_pins UFF0/Q] –to [get_pins UFF1/D]

set_multicycle_path 2 –hold –from [get_pins UFF0/Q] –to [get_pins UFF1/D]

In usual scenario we wish that the hold check scenario remains as it is, but as the Hold check is always performed at the preceeding edge of the setup checking edge,

hence PrimeTime by default will perform hold check after two clock cycles which is not desired. Hence to keep hold checking at the desired edge we set multicycle path of 2 for hold check so that hold checking is performed two edges before the edge at which the setup check is performed.

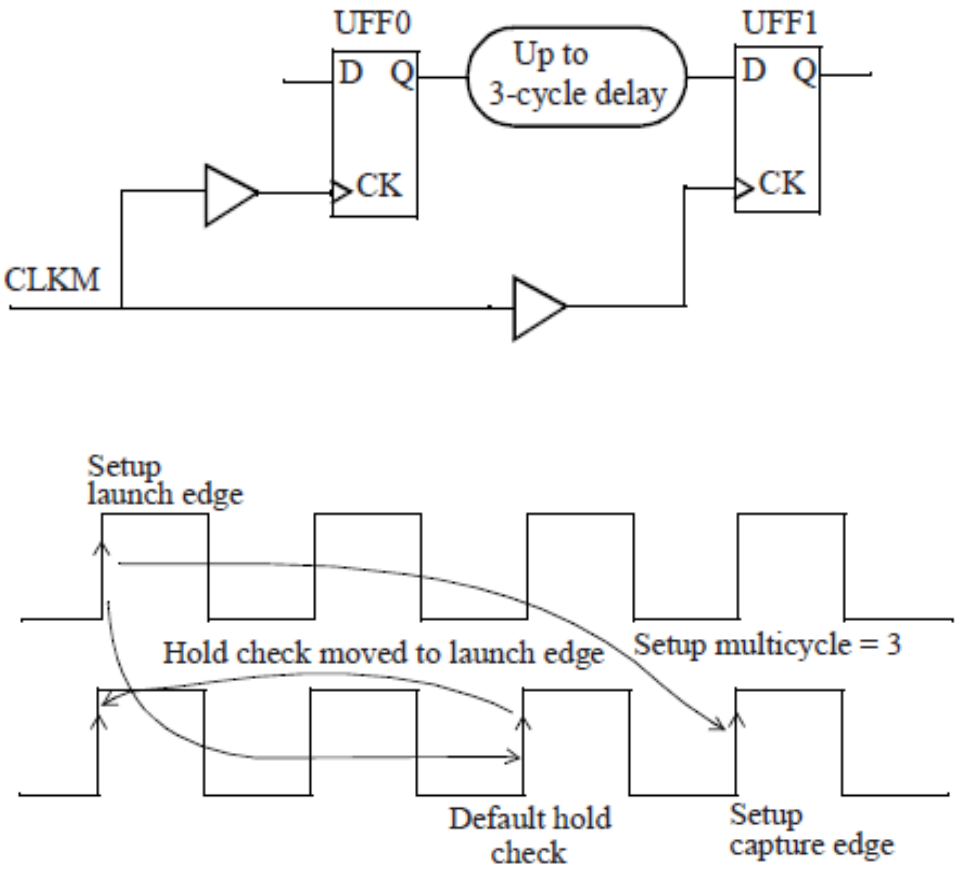

Figure 2.11: Multicycle Path [2]

2.9 PERFORMING CASE ANALYSIS

Case analysis is used if we wish to restrict a particular value of the signal from propagating forward in the logic. If we set a case analysis on a particular pin or port of the design for the particular signal, the signal corresponding to that constant is propagated forward in the design, and the corresponding inverted value is held back.

For example, figure 2.12 shows a multiplexer. Setting the CLK_SEL[0] signal to 0 blocks the PLLdiv16 clock from propagating to the output and therefore disables the timing arc from PLLdiv16 to MIICLK.

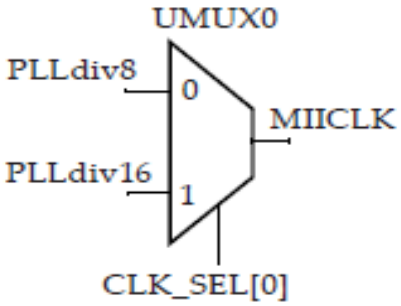

Figure 2.12 Selecting Clock Mode for timing analysis [2]

2.9.1 SETTING AND REMOVING CASE ANALYSIS VALUES

To set case analysis values on specified pins or ports, we run the **set_case_analysis** command. When setting case analysis, we can specify either of the following:

- **Constant logic 0, 1 or static**

  This command sets case analysis for the test port to constant logic 0:

  set_case_analysis 0 [get_ports test]

  The logic value static means constant 0 or 1, without any transition, so that the connected net cannot act as a crosstalk aggressor or experience a change in delay as a crosstalk victim.

- **Rising of Falling Transition**

  This command sets case analysis for the RESET port to a rising transition:

  set_case_analysis rising [get_ports RESET]

  In case of conflict, the following rules apply:

  - A set_case_analysis setting has priority over a built in constant value (for example, in the verilog netlist).
  - A newer set_case_analysis setting has priority over an older setting on the same port or pin.
  - A set_case_analysis value set directly on a port or pin has priority over a conflicting case analysis value propagated to that port or pin.
  - Where propagated case analysis values are in conflict, logic 0 has the highest priority, then the static case setting, then logic 1.

If we wish to remove case analysis values that is applied, we run the **remove_case_analysis** command.

2.10 READING PARASTICS

The **read_parasitics** command can read parasitic data file in the following formats:

- Galaxy parasitic Database (GPD)
- Standard Parasitic Exchange Format (SPEF)
- Detailed Standard Parasitic Format (DSPF)
- Reduced Standard Parasitic Format (RSPF), version IEEE 1481-1999
- Milkyway (PARA)

The **read_parasitics** command will load the current design data with the parasitic information (RC delay) of the cells and nets. A SPEF or RSPF is an ASCII file that can be compressed with gzip. Specifying the format in the command is optional because the reader can automatically determine the file type.

When reading parasitic file, by default PrimeTime assumes that capacitance values specified in the SPEF files do not include the pin capacitance. PrimeTime uses the pin capacitance values specified in the Synopsys design libraries, any pin capacitance values specified in SPEF are ignored. We must ensure that the coupling capacitance in the SPEF file is symmetric. To verify that the coupling capacitance contains only symmetric coupling, read in the design and then use both the **–syntax_only** and **–keep_capacitance_coupling** options of the **read_parasitics** command. PrimeTime checks for asymmetric coupling and issues warning messages when this type of issue is identified. We ensure that the SPEF files contain valid coupling capacitance before proceeding.

The reduced and detailed RC networks specified in SPEF files are used to compute effective capacitance dynamically during delay calculation. The capacitance value reported by most report commands, like the **report_timing** and **report_net** command, is the lumped capacitance, also known as Ctotal. Ctotal is the aggregate total of capacitance values of a net as specified in the SPEF, to which pin capacitance is also added.

Parasitic files are often very large and time consuming for PrimeTime to read. To minimize the read time, make sure that the data file is on a local disk that can be

accessed directly, not across the network. Compressing a SPEF file using gzip can improve overall processing time because the file is much smaller.

Parasitic data files can be large (200 MB or larger) and can contain many parasitic. To verify that the tool back-annotated all cell drivers,run the **report_annotated_parasitics** command.

To create a report for annotated parasitic data files and verify that all RC networks are complete, use the **report_annotated_parasitics –check** command, as shown:

Pin Type	**Total**	**RC pi**	**RC Network**	**Not annotated**
Internal net drive	22456	0	22456	0
Design input port	4	0	4	0
	22460	0	22460	0

Table 2.7 Example of SPEF Annotation Summary

2.11 SETUP AND HOLD FIXING METHODS

There are different methods adopted in VLSI industries to fix setup and hold timimg violations. Sometime fixing one kind of violation may lead to introduction of some other kind of violations

2.11.1 FIXING SETUP VIOLATIONS

Setup violations in a design occur when the delay in the data signal path is larger in comparison to the data arrival time at the capture edge. Setup violations in a circuit can be fixed by using either of the following methods:

a) Replace the High device threshold voltage (HVT) cells with Low Threshold Voltage cells (LVT) in the data path. This increases the current through the cells and cells become faster hence the delay of the path decreases. It has its own limitation as replacing the HVT cells with LVT cells, the leakage current through the cells increases.

b) Replace the weak driving Strength cells with more drive strength cells along the data path. The delay of the cell indirectly depends on the drive strength (W/L) ratio of the device. Higher the drive strength, lower is the delay of the cell.

c) Reduce the wire delay across the signal data path. The wire delay of the design can be reduced by inserting buffers between two cells connected by a particular wire. By inserting the buffers, the transition time reduces, which in turn reduces the delay.
d) Removing redundant buffers from the data path.
e) Instead of placing buffers along the path to reduce the wire delay, two inverters can be connected in serial, so that the overall stage delay of the design decreases.
f) Balancing clock network. If the capture clock arrives early the overall window available for data to reach the capture flop reduces thereby making the path susceptible for setup violations. Adding delays deliberately in the capture clock path can help overcome setup violations, but care must be taken that the successive paths clocked by the same clock are not critical.
g) By using **Retiming Flops**. Retiming flops are used as pipeline stage for fixing setup violating paths which cannot be fixed by any above mentioned paths. These paths are known as Hard Rock paths, which cannot be met by normal fixing, so an extra Flop is introduced in-between the path so that the path can be completed in cycles. This technique essentially does not require a path to be named as Multi Cycle Path which reduces the frequency of operation, but just acts as a pipeline stage. But this method requires extra overhead of area for extra Flop inserted and also RTFs needs to be included in the design at the very initial stage of the design.

2.11.2 FIXING HOLD VIOLATIONS

The hold violations in a design occur when the delay of the path is less than the hold time requirements. The hold violations can be overcome by:

a) Reducing the Driving power of the cells. Reducing the driving power increases the delay of the path.
b) Inserting buffers and inverters across the valid data path to increase the delay of the path.
c) Clock balancing. If the capture clock arrives delayed at the capture flip flop, the hold violation may occur. Hence it is important that the

clock network is properly balanced without introducing excessive delays in the clock paths, which may lead to the hold timing violations.

d) By using **Lock Up Latches**. Lock up latches are commonly used for hold fixing in scan based designs where hold violations are due to large clock skews as in scan test mode the clocks can be asynchronous, leading to larger skews between clocks. Lockup latches are transparent latches that are placed right after the launch flip-flops and are clocked using the same clock as the launch flop clock.

2.12 PRIMETIME DISTRIBUTED MULTI SCENARIO ANALYSIS

Inorder Verify a chip design it is required that multiple PrimeTime timing runs are performed to analyse the operation under multiple operating conditions and various operating modes..

The number of scenarios for a design is

Scenarios = [sets of operating conditions] x [Modes]

The PrimeTime tool can analyse several scenarios in parallel with distributed multi-scenario analysis (DMSA). Rather than analysing each scenario in sequence, DMSA technique will use a master PrimeTime process which at first will set up workers than execute those workers, and finally controls multiple worker processes. primetime allocates one worker for each scenario. We can distribute the processing of scenarios onto different hosts running in parallel, reducing the overall turnaround time. Total runtime is reduced when we share common data between different scenarios.

A single script can control many scenarios, making it easier to set up and manage different sets of analyses. Instead, DMSA algorithm provides a very useful way to perform the analysis of multiple working conditions and modes for a given design and to distribute those analysis tasks onto different hosts.

2.12.1 DEFINITION OF TERMS

The following terms describe aspects of distributed processing:

- **Baseline Image**

  Image that is produced by combining the netlist and the common data files for a scenario.

- **Command Focus**

  Current set of scenarios to which analysis commands are applied. The command focus can consist of all scenarios in the session or just a subset of those scenarios.

- **Current Image**

  Image that is automatically saved to disk when there are more scenarios than hosts, and the worker process must switch to work on another scenario.

- **Master**

  Process that manages the distribution of scenario analysis processes.

- **Scenario**

  A scenario is a unique combination of operating conditions and operating modes

- **Session**

  Current set of scenarios selected for analysis

- **Task**

  Self-contained piece of work defined by the master for a worker to execute.

- **Worker**

  Process started and controlled by the master to perform timing analysis for one scenario, also called slave process.

There is no limit to the number of scenarios that we can create. To create a scenario, the **create_scenario** command is used, which specifies the scenario name and the also the required scripts that are used to apply the analysis conditions and mode settings for the scenario.

The scripts are divided into two groups: common data scripts and specific data scripts. The common data scripts are shared between two or more scenarios, whereas the specific data scripts are specific to the particular scenario and are not shared. This grouping helps the master process manage tasks and to share information between different scenarios, minimizing the amount of duplicated effort for different scenarios.

2.12.2 OVERVIEW OF THE DMSA FLOW

Before we start multi-scenario analysis, we must set the search path and create a .synopsys_pt.setup file

- Setting the search path
- .synopsys_pt.setup file

2.12.2.1 SETTING THE SEARCH PATH

In multi-scenario analysis, we can set the **search_path** variable only at the master. When reading in the search path, the master resolves all relative paths in the context of the master. The master process then automatically sets the fully resolved search path at the worker process. For example, we might launch the master in the /remote1/test/ms directory, and set the **search_path** variable with the following command:

set search_path ". .. ../scripts"

The master automatically sets the path of the worker in the following:

/remote1/test/ms /remote1/test /remote/ms/scripts

The recommended flow in multi scenario analysis is to set the search path to specify the location of

- All files for scenarios and configuration
- All tcl scripts and netlist, SDF, library, and parasitic files to be read in a worker context.

2.12.2.2 .SYNOPSYS_PT.SETUP FILE

The master and workers source the same set of .synopsys_pt.setup files in the following order:

1) PrimeTime install setup file at
   Install_dir/admin/setup/.synopsys_pt.setup
2) Setup file in home directory at
   ~/.synopsys_pt.setup
3) Setup file in the master launch directory at
   $sh_launch_dir/.synopsys_pt.setup

2.12.3 DMSA USAGE FLOW

We can use DMSA capability for timing analysis in PrimeTime and PrimeTime SI as well as for power analysis in PrimePower. This dramatically reduces the turnaround time.

A typical multi-scenario analysis has the following steps:

1) Start PrimeTime in the multi-scenario mode by running the pt_shell command with the **–multi_scenario** option. Alternatively, from a normal PrimeTime session, set the **multi_scenario_enable_analysis** variable to true.
2) Create the scenarios with the **create_scenario** command. Each **create_scenario** command specifies a scenario name and the PrimeTime scripts files that apply the conditions for that scenario.
3) Configure the compute resources that we want to use for the timing update and reporting by running the **set_host_options** command. This command does not start the host, but it sets up the host options for that host.
4) Verify the host options by running the **report_host_usage** command. This command also reports peak memory and CPU usage for the local process and all distributed processes that are already online. The report displays the host options specified, status of the distributed processes, number of CPU cores each process uses, and licenses used by the distributed hosts.
5) Request compute resources and bring the hosts online by running the **start_hosts** command.
6) Select the scenarios which we wish for the session using the **current_session** command. The command specifies a list of scenarios previously created with the **create_scenario** command.
7) Change the scenarios in the current session that are in command focus, using the **current_scenario** command. The command specifies a list of scenarios previously selected with the **current_session** command.
8) View the analysis report and fix validation issues.
    a) Start processing the scenarios by executing the **remote_execute** command or performing a merged report command at the master.
    b) When the processing of all scenarios by the worker processes is complete, we can view the analysis reports. Locate the reports generated by the **remote_execute** command under the directory we specified with the **multi_scenario_working_directory** variable. Alternatively, if we issue the

remote_excute command with the **–verbose** option, all information is displayed directly to the console at the master. The output of all merged reporting commands is displayed directly in the console at the master.

c) Use ECO commands to fix the timing and design rule violations.

2.12.4 ECO FIXING

In PrimeTime Tool, an Engineering Change Order (ECO) is an incremental change in a chip design to reduce design rule constraints (DRC) violations, timing violations, or power. The PrimeTime tool finds these design issues and corrects them by sizing cells, replacing cells, or inserting buffers, and it writes out the changes in script format so that we can implement the changes in other tools.

The commands to perform ECO fixing are:

- fix_eco_drc
- fix_eco_timing
- fix_eco_power

After ECO fixing is complete, to implement the changes, we run the **write_changes** command, which is used to write out the changes in sourceable format, and run the script in the ICC or ICC2 tool, or even third party vendor tools like INNOVUS by carefully updating the script into INNOVUS compatible script. After the changes are implemented, we should perform parasitic extraction and run timing analysis again in PrimeTime tool.

In the PrimeTime tool, we can perform ECO fixing with or without physical placement data:

- **Physically aware mode –** The tool uses only the design netlist, parasitic data, and physical placement data. It replaces cells and inserts buffers only where there is room to make the changes, and the **write_changes** command writes out the location of each change for fast and accurate physical implementation.
- **Logic only mode –** The tool uses only the design netlist and detailed parasitic data, without considering physical placement data.

The ECO fixing flow is compatible with single-core analysis, multicore analysis, and distributed multi-scenario analysis (DMSA).

Chapter 3

DESIGN IMPLEMENTATION AND TIMING FIX

Performing complete timing analysis of a SoC is quite complex and also requires continuous iterations to fix the timing violations on the chip which is a time consuming process. In this chapter a relatively simpler design compared to a SoC is first designed and then various analyses are performed to close the timing requirements of the design without any Timing fails. As discussed in the previous chapters there are different methods to fix the Timing violations for both Setup and Hold violations.

Here some of the Industry adopted techniques are used to fix the timing fails of the design. Also most of the times fixing one of the violations may lead to violations in other domain hence complete coverage of multiple scenarios is also crucial in closing timing for a given design.

3.1 DESIGN SETUP

The primary inputs required for Static Timing Analysis flows are:

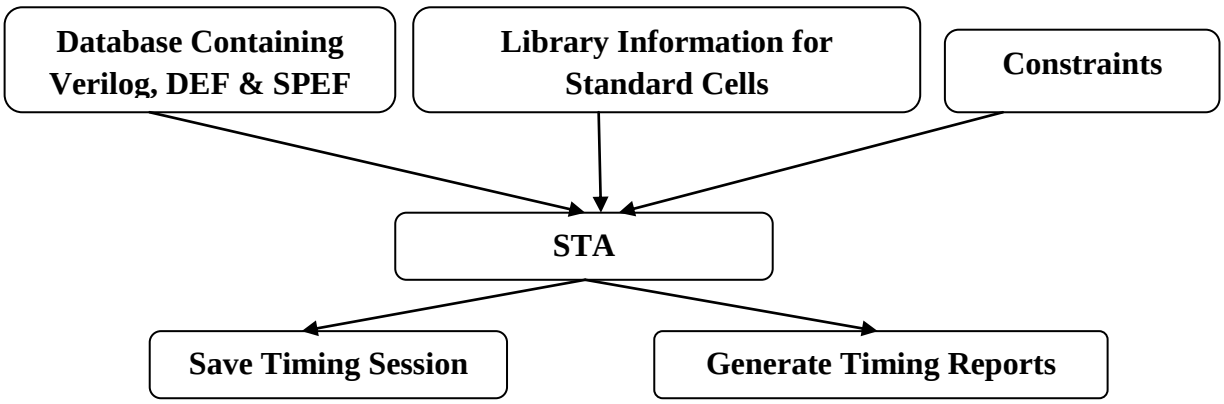

Figure: 3.1 Primary STA Flow

3.1.1 READ IN DESIGN DATA

- Reading the verilog which defines the design of the circuit (.v file)
  The veilog file containing the connectivity of different cells and the circuit is read into by using the **read_verilog** command in PrimeTime
- Linking the paths for the standard cell libraries (.lib files)
  The path for the standard cell libraries containing detailed information about the standard cells as supported by the foundries is set using the **set_app_var link_path** command in PrimeTime.
- Link the design
  Once the required files are read into finally the design is linked using **link_design** command in PrimeTime

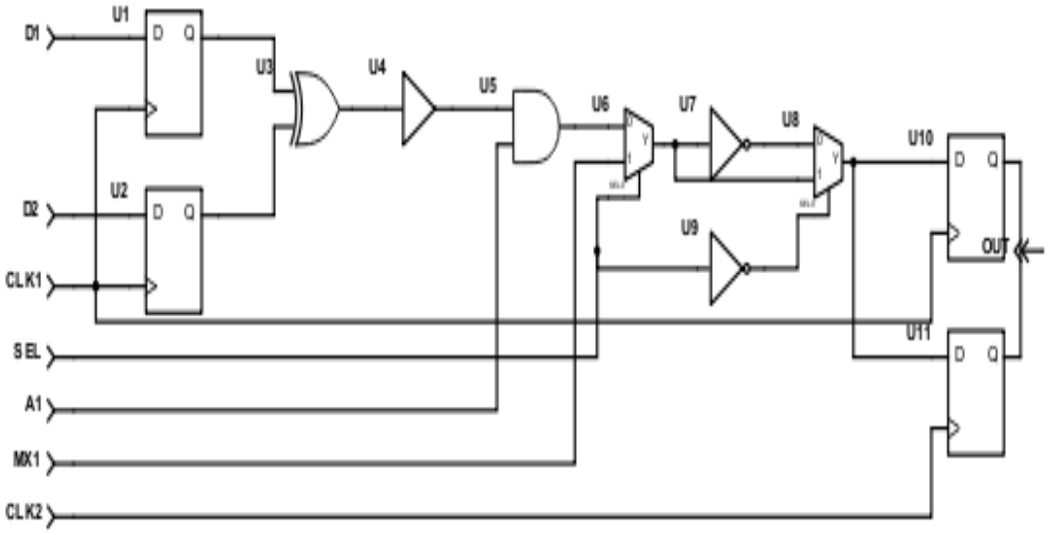

Figure: 3.2 Test Case Schematic

3.1.2 CONSTRAINING THE DESIGN

Once the design data is read into and the design is linked, it is required that the design is correctly Constrained using suitable constraints required for the design. Using different PrimeTime commands the design is constrained and a constraint file with **.sdc** extension contains all the required constraint of the design. This file is than provided as an input to STA process flow as depicted in Figure 3.1.

One limitation of Static Timing Analysis is that it does not check timing on paths which have not been constrained. Thus, all paths must be constrained to enable their analysis. Some of the constraints applied to the design are

- set_input_delay –clock CLK1 –max 3.0 [get_ports D1]
- set_output_delay –clock CLK1 –max 2.0 [get_ports OUT]

3.1.3 CLOCKS

For the given design, the Clocks are defined using the following commands:

- create_clock –name CLK1 –period 8 –waveform {0.0 4.0} [get_ports CLK1]
- create_clock –name CLK2 –period 6 –waveform {0.0 3.0} [get_ports CLK2]
- set_clock_uncertainty –setup 0.2 [get_clocks *]
- set_clock_uncertainty –hold 0.5 [get_clocks *]
- set_clock_latency 1.2 [get_clocks CLK1]
- set_clock_latency 5.0 [get_clocks CLK2]

3.1.4 TIMING EXCEPTIONS

3.1.4.1 MULTI-CYCLE PATHS

For the design, the Timing Exceptions for Multicycle paths are applied as follows:

- set_multicycle_path 3 –setup –from [get_pins U2/Q] –to [get_pins U11/D]
- set_multicycle_path 2 –hold –from [get_pins U2/Q] –to [get_pins U11/D]

3.1.4.2 FALSE PATHS

For the design, the Timing Exceptions for false paths are applied as follows:

- set_false_path –from [get_pins U6/A] –to [get_pins U8/A]
- set_false_path –from [get_pins U6/B] –to [get_pins U8/B]

3.1.5 SPECIFYING CASE ANALYSIS

For the design the case analysis is applied on the select pin of the multiplexers:

- set_case_analysis 0 [get_ports SEL]

3.1.6 READING PARASITICS

During the Physical implementation stage of the Design, the design team dumps out the **.spef** files using the PNR tool (e.g, Innovus by cadence, ICC2 by Synopsys) which is than provided as an input to the STA process flow as depicted in Figure 3.1. Alternatively the SPEF (Standard Parasitic Exchange Format) files can also be linked to design using the **read_parasitics** command in PrimeTime. This process is known as Back-Annotation.

- read_parasitics my_design.spef

3.2 TIMING CHECKS

3.2.1 SETUP TIMING CHECKS

For the design, setup check is performed using the command:
report_timing –from U1/CK –to U10/D

```
Startpoint: U1 (rising edge-triggered flip-flop clocked by CLK1)
Endpoint: U10 (rising edge-triggered flip-flop clocked by CLK1)
Path Group: CLK1
Path Type: max
Point                                                  Incr  Path
-----------------------------------------------------------------
clock CLK1 (rise edge)                                 0.00  0.00
clock network delay (ideal)                            1.20  1.20
input external delay                                   3.00  4.20 r
U1/CK (DFF )                                           0.00  4.20 r
U1/Q (DFF )                                            0.16& 4.36 f
U3/a (EX-NOR)                                          0.04& 4.40 f
U3/z (EX-NOR)                                          0.25& 4.65 f
U4/a (BUF)                                             0.10& 4.75 f
U4/z (BUF)                                             0.15& 4.90 f
U5/a (AND)                                             0.30& 5.20 f
U5/z (AND)                                             0.45& 5.65 f
U6/a (MUX)                                             0.22& 5.87 f
U6/y (MUX)                                             0.28& 6.15 f
U8/a (MUX)                                             0.74& 6.89 f
U8/y (MUX)                                             0.67& 7.56 f
U10/D (DFF )                                           0.23& 7.79 f
data arrival time                                            7.76
clock CLK1 (rise edge)                                 8.00  8.00
clock network delay (ideal)                            1.20  9.20
clock uncertainty                                      -0.20 9.00
library setup time                                     -0.40 8.60
data required time                                           8.60
-----------------------------------------------------------------
data required time                                     8.60
data arrival time                                      7.76
-----------------------------------------------------------------
slack (MET) 0.84
```

Figure 3.3: Setup Timing Report

In the above timing report for setup check, the path meets setup timing requirements by 840ps. If there would have been violation, it can be met by any one the technique mentioned in the **section 2.11.1.**

3.2.2 HOLD TIMING CHECKS

For the design, hold timing check is performed using the following command:

Report_timing –path_type min –from U2/CLK –to U11/D

```
Startpoint: U2 (rising edge-triggered flip-flop clocked by CLK1)
Endpoint: U11 (rising edge-triggered flip-flop clocked by CLK2)
Path Group: CLK2
Path Type: min
Point                                                  Incr  Path
-----------------------------------------------------------------
clock CLK1 (rise edge)                                 0.00  0.00
clock network delay (ideal)                            1.20  1.20
U2/CK (DFF )                                           0.00  1.20 r
U2/Q (DFF )                                            0.16& 1.36 f
U3/b (EX-NOR)                                          0.06& 1.42 f
U3/z (EX-NOR)                                          0.25& 1.67 f
U4/a (BUF)                                             0.10& 1.77 f
U4/z (BUF)                                             0.15& 1.92 f
U5/a (AND)                                             0.30& 2.22 f
U5/z (AND)                                             0.45& 2.67 f
U6/a (MUX)                                             0.22& 2.89 f
U6/y (MUX)                                             0.28& 3.17 f
U8/a (MUX)                                             0.74& 3.91 f
U8/y (MUX)                                             0.67& 4.58 f
U11/D (DFF )                                           0.23& 4.81 f
data arrival time                                            4.81
clock CLK2 (rise edge)                                 0.00  0.00
clock network delay (ideal)                            5.00  5.00
clock uncertainty                                      0.50  5.50
library hold time                                      0.35  5.85
data required time                                           5.85
-----------------------------------------------------------------
data required time                                     5.85
data arrival time                                      4.81
-----------------------------------------------------------------
slack (VIOLATED) -1.04
```

Figure 3.4: Hold Timing Report

In the hold timing report, the path violates hold timing requirements by 1.04ns. The hold timing violations can be met by any one of the technique mentioned in **section 2.11.2.**

3.3 TIMING FIXES

The hold violation of the circuit is met using Lock-Up technique.

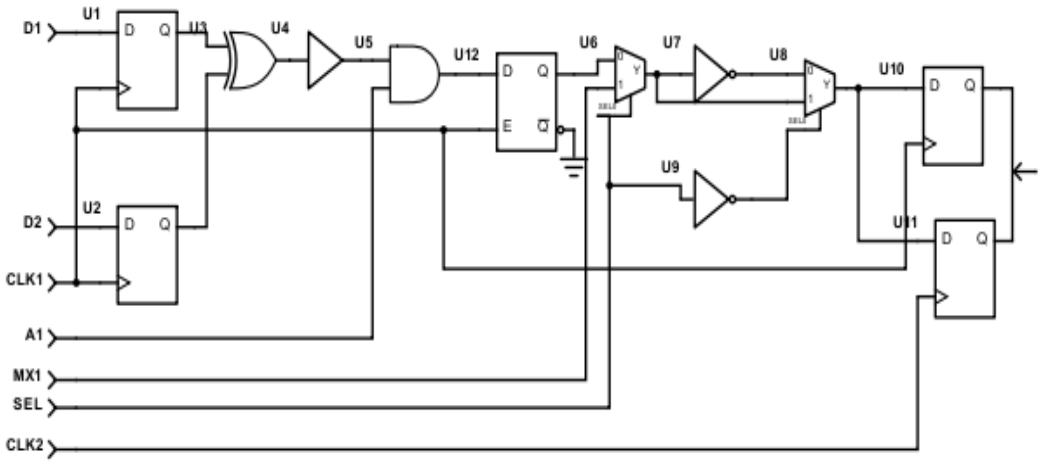

Figure 3.5: Test Case Schematic with Lock-Up Latch

```
Startpoint: U2 (rising edge-triggered flip-flop clocked by CLK1)
Endpoint: U11 (rising edge-triggered flip-flop clocked by CLK2)
Path Group: CLK2
Path Type: min
Point                                                   Incr  Path
-----------------------------------------------------------------------
clock CLK1 (rise edge)                                  0.00  0.00
clock network delay (ideal)                             1.20  1.20
U2/CK (DFF )                                            0.00  1.20 r
U2/Q (DFF )                                             0.16& 1.36 f
U3/b (EX-NOR)                                           0.06& 1.42 f
U3/z (EX-NOR)                                           0.25& 1.67 f
U4/a (BUF)                                              0.10& 1.77 f
U4/z (BUF)                                              0.15& 1.92 f
U5/a (AND)                                              0.30& 2.22 f
U5/z (AND)                                              0.45& 2.67 f
U12/D (DLAT)                                            0.10& 2.77 f
U12/Q (DLAT)                                            1.36& 4.13 f
U6/a (MUX)                                              0.22& 4.35 f
U6/y (MUX)                                              0.28& 4.63 f
U8/a (MUX)                                              0.74& 5.37 f
U8/y (MUX)                                              0.67& 6.04 f
U11/D (DFF )                                            0.23& 6.27 f
data arrival time                                             6.27
clock CLK2 (rise edge)                                  0.00  0.00
clock network delay (ideal)                             5.00  5.00
clock uncertainty                                       0.50  5.50
library hold time                                       0.35  5.85
data required time                                            5.85
-----------------------------------------------------------------------
data required time                                      5.85
data arrival time                                       6.27
-----------------------------------------------------------------------
slack (MET) 0.42
```

Figure 3.6 Hold Violation Fixed

3.4 DMSA ENVIRONMENT SETUP

The algorithm for setting up the Distributed Multi Scenario Analysis (DMSA) is shown below:

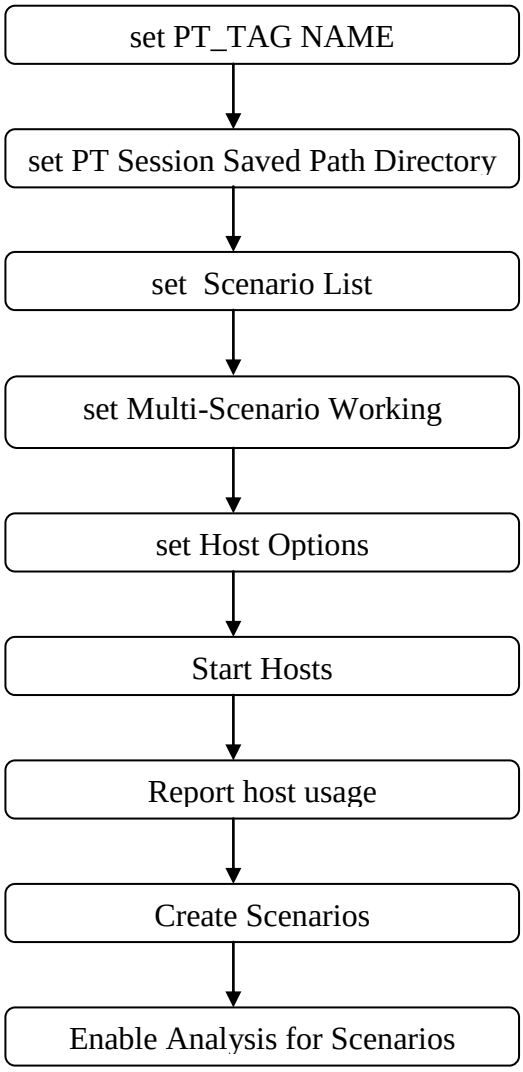

Figure 3.7 DMSA Setup Environment Algorithm

The DMSA environment is setup up using the above alogirthm, and PrimeTime commands using the Tool Command Language (TCL). Once the environment is set analysis of different timing violations is done as described in **section 2.12.3** and **2.12.4.**

A list of Scenarios is created for which we wish to perform the timing analysis. This list of scenarios can be created based on different algorithms which extract the dominant corners for which maximum timing violations occur.

3.4.1 DOMINANT CORNER EXTRACTION

As discussed in Chapter 1, with decreasing technology nodes and increasing chip complexity, the number of scenarios for analysis for a SoC has increased exponentially. The general trend for selecting the Dominant corner/scenarios is based on the maximum number of endpoints that violates for a particular scenario. The corner having having most number of violating endpoints is usually considered to be the dominant corner, but this sometimes can be misleading. For example in Table 3.1

Scenarios	No. Of Violating Endpoints	Worst Negative Slack (ns)
Scenario 1	371	-0.4
Scenario 2	233	-5.4
Scenario 3	177	-1.5

Table 3.1 Worst Case Delay Scenarios

From table 3.1 we see that the dominant corner based on the maximum number of violating endpoints (NVP) is Scenario 1, but based on the Worst Negative Slack (WNS) Scenario 2, is the dominant scenario. There are different algorithms that take into account either NVP or WNS as a deciding factor for Dominant corner; there are also algorithms which consider using both NVP and WNS as deciding factor for dominant corner extraction [1]. We have considered a Design A for which Dominant corners are extracted to setup DMSA environment.

Scenario	NVP	WNS
Func_0.550_0.765_tt_rcw_ccw_0c_ptsi_pocv	437	-0.560
Func_0.660_0.675_tt_rcw_ccw_125c_ptsi_pocs	789	-0.012
Func_0.720_0.765_tt_cw_ccw_125c_ptsi_pocv	956	-0.234
Func_1.090_1.090_ff_cb_ccb_0c_ptsi_pocv	11877	-0.380
Func_1.090_1.090_fs_rcb_ccb_125c_ptsi_pocv	8856	-0.670
Func_0.800_0.835_tt_rcw_ccw_0c_ptsi_pocv	5988	-1.2

Table 3.2 Worst Case Delay Scenarios for Design A

From Table 3.2, we find out the worst scenarios for the Design A. According to number of FEP counts, corner **Func_1.090_1.090_ff_cb_ccb_0c_ptsi_pocv** seems to be the worst scenario, but according to Worst Negative Slack (WNS), corner **Func_0.800_0.835_tt_rcw_ccw_0c_ptsi_pocv** seems to be worst. Hence using our algorithm, we consider our scenarios for DMSA setup as

- Func_1.090_1.090_ff_cb_ccb_0c_ptsi_pocv
- Func_1.090_1.090_fs_rcb_ccb_125c_ptsi_pocv
- Func_0.800_0.835_tt_rcw_ccw_0c_ptsi_pocv

3.5 GENERATING ECO PRIMETIME DMSA

The algorithm to generate Engineering change order (ECO) using the PrimeTime tool is given below:

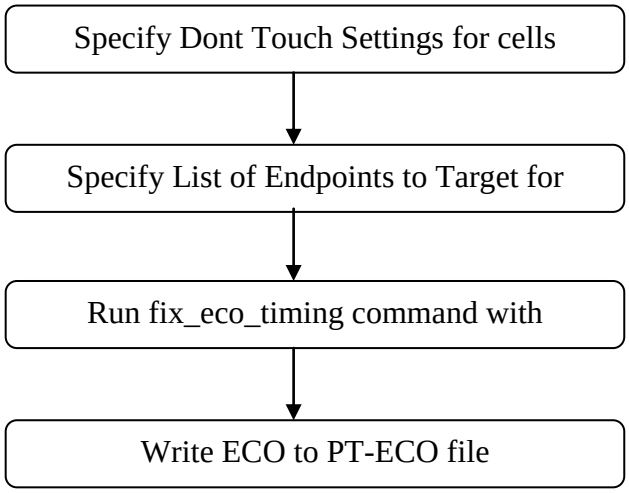

Figure 3.8: Generating PrimeTime ECO's Algorithm

The above algorithm generates ECO file PT-ECO that contains the required ECO to fix Hold Violations. The command used to generate Hold Eco is:

fix_eco_timing –setup_margin 0.500 –type hold –methods insert_buffer –buffer_list {buf1 buf2 buf3 buf4 buf5} –path_selection_options {-delay_type min –pba_mode none –to $endpoints –max_paths 100000 –nworst 5000 –slack_lesser_than 0}.

The ECO file generated is in PrimeTime ECO format, if the Physical Design tool used is ICC2 the ECO file can be sourced directly, but if the PD tool used is INNOVUS, the PT-ECO file needs to be converted into INNOVUS compatible ECO file, so that it can be sourced for Physical Implementation.

Chapter 4

RESULTS & DISCUSSIONS

At first the setup environment is discussed along with different tools and their versions used for the Analysis. After the setup is done, results are analysed in details along with each iterations of the run for hold fixing once the dominant corners were were selected in section 3.4.1.

4.1 EXPERIMENTAL SETUP

Before we start performing ECO cycles and also analysis different results it is important the tool environments are set correctly. The different tools that are used along with their versions used are shown in Table 4.1

Operating System	SUSE Linux Enterprise Linux Version v11.4
Number of CPU's	4 CPU Cores
Tool	1) Synopsys PrimeTime Tool Tool Version 2017.12.SP3-3 2) Dorado Tweaker Tool Tool Version 2019.05.02 (T1-mmc)
Scripting Language	TCL Version v8.5

Table 4.1: Different Tools and their Versions

Once the environment is setup, the first step is to analyse different timing reports generated from the Static Timing Analysis runs. Basically for multi-corner runs, we analyse the merged reports generated, by taking into account worst fails for each scenarios for each clock groups. We basically obtain merged reports for 4 different scenarios:

- Functional Setup Merged Reports
- Test Setup Merged Reports

- Functional Hold Merged Reports
- Test Hold Merged Reports

Here the number of FEPs and WNS given in Table 3.2 are for Functional-Hold Violations. For comparisons we also use Tweaker Tool by Dorado which is widely used tool for Hold fixing. The results obtained from ECOs generated from both the Methods are compared and analysed to check which method provides better results.

In traditional ECO fixing method, each corner wise specific ECOs are generated and than implemented, this method requires more number of iterations to be performed and also increases the implementation days by manifold.

4.2 DESIGN SUMMARY

Table 4.2 shows detailed summary of each set of ECO iterations run for hold fixing using PrimeTime DMSA. The number of Dominant corners selected for DMSA is 3 out of total of 15 scenarios. Once the dominant corners are selected, the next step is to generate the list of failing Endpoints that we need fix using ECO. Once the Endpoints list is obtained, DMSA environment is setup using the selected scenarios. Here detail of each round of ECOs generated and timing fixed are shown.

	INITIAL DATA	**ECO1**	**ECO2**
Total Paths	255178	255178	255178
Number of FEPs	29343	2311	267
WNS	-1.2	-0.36	-0.012
TNS	645.43	-178.03	-29.01
Buffers inserted for fixing HOLD	-	7845	1244

Table 4.2 PrimeTime DMSA ECO results.

	INITIAL DATA	**ECO1**	**ECO2**
Total Paths	255178	255178	255178
Number of FEPs	29343	4623	894
WNS	-1.2	-0.40	-0.140
TNS	645.43	-165.22	-21.34
Buffers inserted for fixing HOLD	-	9456	1765

Table 4.3 Tweaker ECO results

4.3 INFERENCES

After running PrimeTime DMSA based ECO iterations and also Tweaker based ECO iterations, we can now summarise the details in a summary table.

Criteria	Traditional Method	Tweaker Method	DMSA Method
Effort	40 Days	8 Days	4 Days
Number of ECO cycles	12	4	2

Table 4.4: Summary

CONCLUSION

The results that are obtained from using DMSA method for generating ECO's for Static Timing Analysis proves to be better as compared to already existing technologies like the one using Dedicated ECO Tools like Tweaker. Also generating ECO's manually for each scenarios is a time consuming and in today's world where we find new technology coming every year, it is hard to invest long duration for a chip to be taped out since its design phase. Hence most if the Chip Design vendors usually keep their chip design phase time to be as small as possible in order to compete with other vendors.

Engineering Change Order (ECO) play a crucial role in determining the overall design time for a chip, hence efforts are always given to reduce the ECO Cycle time. Timing ECO's in particular are most crucial, as the number of timing violations in a chip are more compared to other Signoff check violations.

The method described below reduces the manual effort days by nearly half for ECO's generation and to be rolled in compared to other technologies, also since the ECO's are generated using PrimeTime tool itself, the tool using which the Timing Analysis is being performed, there is better correlation between the data generated using ECO and Timing analysis.

Future works may include optimizing the algorithm to further enhance the efficiency of ECO's and also reducing the overall cycle times.

REFERENCES

[1] Aditi Sharma "Smart and Efficient Multi Scenario SoC Timing Closure and ECO Generator" Indraprastha Institute of Information Technology, Delhi, June 2015.

[2] J.BHasker and Rakesh Chadha. "Static Timing Analysis For Nanometer Designs: A Practical Approach" Springer, 2009.

[3] Primetime and Primetime SI User Guide.

[4] S. Narendra J. Tschanz A. Keshavarzi S. Borkar, T. Karnik and V. De. "Parameter variations and impact on circuits and microarchitecture". Design Automation Conference, pages 338–342, 2003.

[5] Yao-Kai Yeh Jui-Hung Hung, Yung-Sheng Tseng, and Tsai-Ming Hsieh. "A New ECO Technology For Functional Changes And Removing Timing Violations" International Symposium on Quality Electronic Design, pages 1–5, 2011.

[6] Arvind NV Sreeram C Vish Visvanathan-Shailendra Dhuri Roopesh Chander Patrick Fortner Subra Sripada Qiuyang Wu Rajagopal KA, Sivakumar R. "A comprehensive solution for true hierarchical timing and crosstalk delay signoff" International Conference on VLSI Design, pages 1–6, 2006.

[7] Joel R. Phillips Luis Guerra e Silva, L. Miguel Silveira. "Efficient Computation of the Worst Delay Corner" Design, Automation & Test in Europe Conference & Exhibition, pages 1–6, April 2007.

[8] Wei Chen Yongqiang Lu Qiang Zhou Weixiang Shena, Yici Cai and Jiang Hue. "Useful clock skew optimization under a multi-corner multi-mode design framework" Quality Electronic Design (ISQED), pages 62–68, March 2010.

[9] M. Jasmin. "Optimization Techniques for Low Power VLSI Circuits" Middle-East Journal of Scientific Research, pages 1082–1087, 2014.