A DISSERTATION ON

STATIC TIMING ANALYSIS OF A DEEP-SUBMICRON DESIGN

Submitted in partial fulfilment of the requirement for the award of the degree
Of
MASTER OF TECHNOLOGY
In
VLSI DESIGN & EMBEDDED SYSTEMS

Submitted By:

BABUL ANUNAY (02/VLS/2k10)

Under the supervision and guidance of

Mr Anurag Gupta	**Mrs. Rajeshwari Pandey**
Design Manager	**Associate Professor**
Freescale Semiconductors	**DTU**

Department of Electronics & Communication Engineering

Delhi Technological University

(Formerly DELHI COLLEGE OF ENGINEERING)

BAWANA ROAD

DELHI - 10042

CERTIFICATE

It is certified that Babul Anunay (02/VLS/2K10), student of Master of Technology, VLSI Design & Embedded Systems, Department of Electronics & Communication Engineering, Delhi Technological University (Formerly Delhi College of Engineering), have submitted the dissertation entitled 'STATIC TIMING ANALYSIS OF A DEEP-SUBMICRON DESIGN' under my guidance towards partial fulfilment of the requirements for the award of the degree of Master of Technology (VLSI Design & Embedded Systems).

This dissertation is a bona-fide record of project work carried out by him under my guidance and supervision. His work is found to be excellent and his discipline impeccable during the course of the project.

I wish him success in all his endeavours.

Mrs Rajeshwari Pandey (**Assoc. Prof**.)
Project Guide,
Electronics & Communication
Delhi Technological University
Delhi -110042

Prof. Rajiv Kapoor
Head of Department
Electronics & Communication
Delhi Technological University
Delhi- 110042

Noida
25-06-2012

CERTIFICATE

This is to certify that the project entitled **STATIC TIMING ANALYSIS OF A DEEP-SUBMICRON DESIGN** was carried out by BABUL ANUNAY (BID 40070) at **FREESCALE SEMICONDUCTORS, NOIDA** under my guidance during **January, 2012** to **June, 2012**.

Mr Anurag Gupta
Design Manager
Freescale Semiconductors, Noida

ACKNOWLEDGMENTS

Industrial project is a period to strengthen our theoretical, practical and technical concepts, which enhances our skills in the field of technology. The successful realization of the project is an outgrowth of a consolidated effort of the people from disparate fronts. It's only with their support and guidance that the developer could meet the end.

I am very thankful to Freescale Semiconductor that it provided me with immense practical exposure through this internship period. Needless to say, this project will prove to be very beneficial for me in my career. During my stay of six months in the company, I have accumulated many debts, only a proportion of which I have space to acknowledge here. This experience has not only added to my technical knowledge but has helped me to get an insight in to quality concepts that are the attributes required to be a successful professional.

Firstly, I would like to express my heartiest gratitude and thanks to my project guide Mrs. Rajeshwari Pandey (Asst. Prof.), Department of Electronics & Communication; Delhi Technological University, Delhi, for continuous inspiration, encouragement and guidance in every stage of preparation of this thesis.

I would like to thank my manager Mr. Anurag Gupta, my project guide, Mrs. Neha Mathur, Mr. Siddharth Taneja, Mr Shailesh Gupta and Mr Rahul Kumar for their guidance and encouragement.

I would also like to express my gratitude to Mr. Prateek Gupta, Mr Amol Agarwal,Mr Ashish Maitra, Mr. Ashish Goel, and whole of the STA team members for providing me with many valuable inputs and help from time to time. I am also grateful to the Ms. Ashu Dua, Ms Priyanka Baweja and the HR team at Freescale India for their support during my project.

I am extremely thankful to Prof.Rajiv Kapoor (Head of the Dept.), Prof. Asok Bhattacharya and Dr Neeta Pandey, Dept of Electronics and Communication Engineering, Delhi Technological University, Delhi, for the motivation and inspiration during the entire duration of this thesis. I am also thankful to all teaching and non-teaching staff at DTU, and my fellow students, who have helped me directly or indirectly in completion of this thesis.

This acknowledgement is incomplete without a special thanks to My Freescale Buddy Mr Gourav Kapoor and Mr Naman Gupta who taught me the basic fundamentals of STA , tcl scripting and were always ready to help me in a way so that I can learn more in the most effective manner.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

KEYWORDS: Static Timing Analysis, Standard Delay Format, Gate Level Simulation, Standard Parasitic Exchange Format.

Timing holds a very important part in any VLSI design and incorporates the sense of realism into the design. Static Timing Analysis (also referred as STA) is one of the many techniques available to verify the timing of a digital design. Static timing analysis is a complete and exhaustive verification of all timing checks of a design, without requiring simulation.

The more important aspect of static timing analysis is that the entire design is analyzed once and the required timing checks are performed for all possible paths and scenarios of the design. Further , the design can be analyzed in various corners to ensure the proper functioning of the design in all the scenarios. Thus, STA is a complete and exhaustive method for verifying the timing of a design.

The project is aimed at understanding as well as performing clocking and timing closure of the whole SoC chip. This elaborates on the basic flow adopted to check the timing of a chip, procedures to meet them and debug the errors. The tool used here is ETS which helps in forming detailed reports for finding the violations and simulating the timing behaviour of the circuit.

CHAPTER 1
INTRODUCTION

Semiconductors have been at the heart of most technological progress in the past few decades and have had a profoundly positive impact on the human condition. It can be said that this is the “golden age of electronics”. Human ingenuity has enabled the creation of products that customers never asked for, but now cannot live without - everything from computers to digital cameras, mp3 players and mobile phones. The most successful products and applications are the creative use and convergence of technologies and application domains. Truly speaking, this is only the beginning. The consumerization of electronics, globalization of markets and the human aspiration for a better quality of life is creating unprecedented opportunities that only the semiconductor industry can help realize.

With rapid advances in semiconductor processing technologies, the density of gates on the die increased in line with what Moore's law predicted. This helped in the realization of more complicated designs on the same IC. Over the last few years, with the advent of bleeding edge technology applications like HDTV and 3rd generation mobile devices, an increasingly evident need has been that of incorporating the traditional microprocessor, memories and peripherals -or in other words the whole system - on single silicon. This is what has marked the beginning of the SoC era.

During the project at Freescale Semiconductors India Pvt. Ltd., the project work was assigned in the Physical Design Department and required to carry out tasks concerned with timing analysis. Following discussion gives a details of tasks assigned and the means/methods used to accomplish the same in the allotted time.

UNIX as an operating system is very flexible and is widely used in industries. Familiarity with UNIX (operating system) is an important step in getting started with the project. This was learned and explored during the project period.

1.1 UNIX

UNIX has been in use for more than three decades. Originally it rose from the attempt in the early 1960s to develop a reliable timesharing operating system. Bell Labs developed a system that provided a work environment described as "of unusual simplicity, power, and elegance". Because of Microsoft’s aggressive marketing practices, millions of users who have no idea what an operating system is have been using Windows operating systems given to them when they purchased their PCs. Many others are not aware that there are operating systems other than Windows.

Advantages Of UNIX

- UNIX is more flexible and can be installed on many different types of machines, including main-frame computers, supercomputers and micro-computers.
- UNIX is more stable and does not go down as often as Windows does, therefore requires less administration and maintenance.
- UNIX has greater built-in security and permissions features than Windows.
- UNIX possesses much greater processing power than Windows.
- UNIX is the leader in serving the Web. About 90% of the Internet relies on Unix operating systems running Apache, the world's most widely used Web server.
- Software upgrades from Microsoft often require the user to purchase new or more hardware or prerequisite software. That is not the case with UNIX.
- The mostly free or inexpensive open-source operating systems, such as Linux and BSD, with their flexibility and control, are very attractive to (aspiring) computer wizards. Many of the smartest programmers are developing state-of-the-art software free of charge for the fast growing "open-source movement".
- UNIX also inspires novel approaches to software design, such as solving problems by interconnecting simpler tools instead of creating large monolithic application programs.
- UNIX has an awesome text editor which is much more powerful than MSOffice.

1.2 Tcl

Tcl is the acronym for "Tool Command Language" (it is pronounced "tickle"). Tcl is actually divided into two things: a language and a library. Tcl is a simple textual programming language, intended for issuing commands to interactive programs such as text editors, debuggers and shells. It has a simple syntax and it is also programmable. Tcl users can write command procedures to provide more powerful commands than those given in the built−in set.

Second, Tcl is a library package embeddable in applications. The Tcl library consists of a parser for the Tcl language, routines to implement the Tcl built −in commands, and procedures which allow each application to extend Tcl with additional commands specific to that application. The application program generates Tcl commands and passes them to the Tcl parser for execution. Commands may be generated by reading characters from an input source, or by associating command strings with elements of the application's user interface, such as menu entries, buttons, and other widgets. When the Tcl library receives commands it parses them into component fields and executes built−in commands directly. For commands implemented by the application, Tcl calls back to the application to execute the commands. In many cases commands will make recursive invocation s of the Tcl interpreter by passing in additional strings to execute (in fact procedures and conditional−looping commands all work in this way).

An application program can obtain many advantages by using Tcl for its command language: Tcl provides a standard syntax: once users know Tcl, they will be able to issue commands easily to any Tcl−based application. Tcl succeeds to provides programmability. All a Tcl application needs to do is to implement a few application−specific low−level commands. Tcl provides many utility commands and a general programming interface for building up comp lex command procedures. By using Tcl, applications need not reimplement these features.

It is important to note that Tcl was designed thinking that the programmer should actually use two or more languages when designing large software systems. One for manipulating complex internal data structures, or where performance is important, and another, such as Tcl, for writing very small scripts that glue together the other pieces, providing hooks for the user to extend. For the Tcl script writer, ease of learning, ease of programming and ease of gluing are more important than performance or facilities for complex data structures and algorithms.

Tcl was designed to make it easy to drop into a lower language when you come across tasks that make more sense at a lower level. In this way, the basic core functionality can remain small and one need only bring along pieces that one particular wants or needs.

CHAPTER 2
SYSTEM-ON-CHIP DESIGNS

Today’s feature-rich products require embedded system solution with complex System-on-Chip (SoC) to meet market expectations of high performance at low cost and lower energy consumption. SoCs are complex designs with multiple embedded processors, memory subsystems, and application specific peripherals. While the potential is huge, the complexities are several, and countering these to offer successful designs is a true engineering challenge.
SoCs make up more than 40% of chip market today.

These trends are ample evidence that SoCs are growing in importance in the semiconductor industry. The reasons are not far to look: SoCs make available, on a single piece of silicon, the embedded IP and high system-level integration required for performance demanding applications today. This enables semiconductor manufacturers to cost-effectively meet specific system requirements while delivering competitive time-to-market advantage.

Paradoxically, if the opportunities look promising, the challenges are no less daunting. While opportunities come in the form of drastic reduction in the overall cycle time of the system with superior performance levels, challenges are in the form of deep sub-micron complexities, faster timing closure requirements, verification challenges and the need for an extensive portfolio for pre-verified IP components.

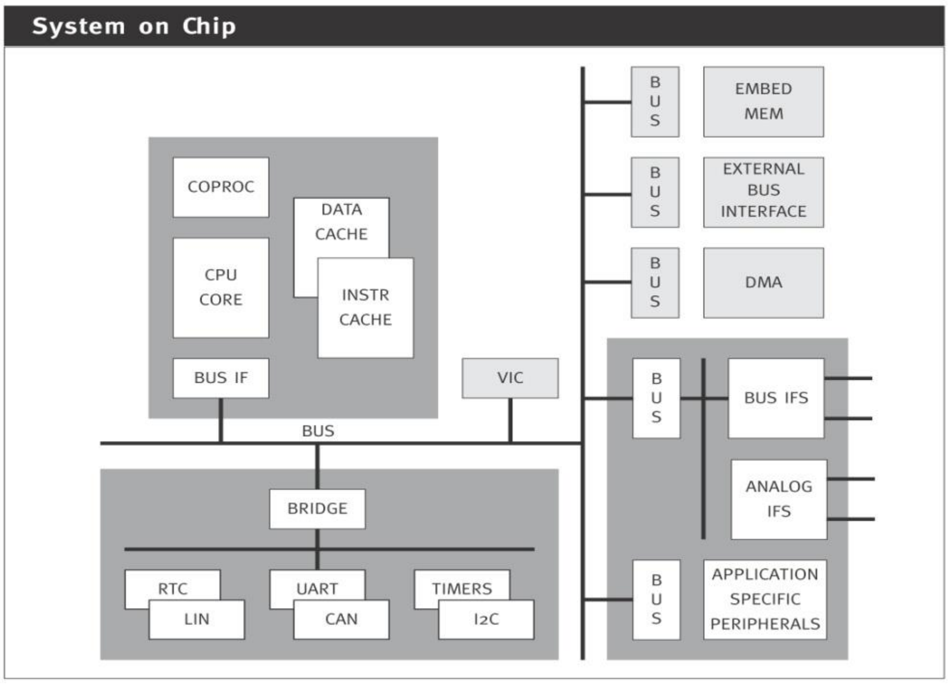
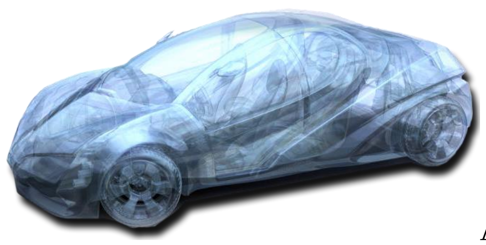


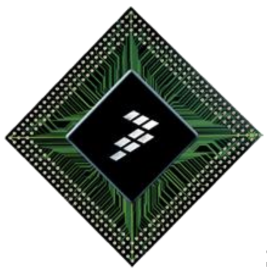
Fig 2.1 SoC Architecture [1]

2.1 SoC DESIGN FLOW

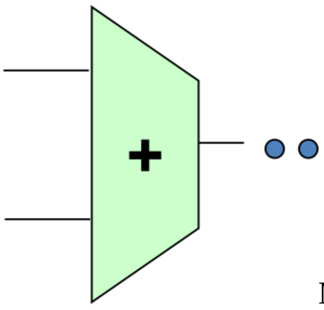
It is a strategy that can successfully realize a SoC from concept to silicon or a set of steps required to be executed in a particular sequence to ensure that you get functional silicon which behaves in the desired fashion.



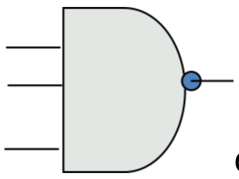
APPLICATION



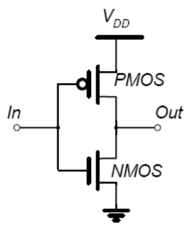
SYSTEM



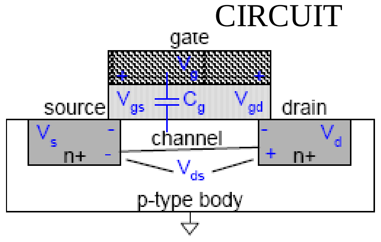
MODULE



GATE



CIRCUIT



DEVICE

Fig 2.2 Levels of Architecture [1]

2.1.1 SPECIFICATION

A product is born out of necessity or a requirement. A potential customer identifies a set of requirements endorsed by the marketing team. Then the Architecture team brings out the architecture of the product based on these requirements. Once these requirements are frozen, a specific document is released which tells the detailed functionality of the ASIC, every IP that is being used and protocols if any.

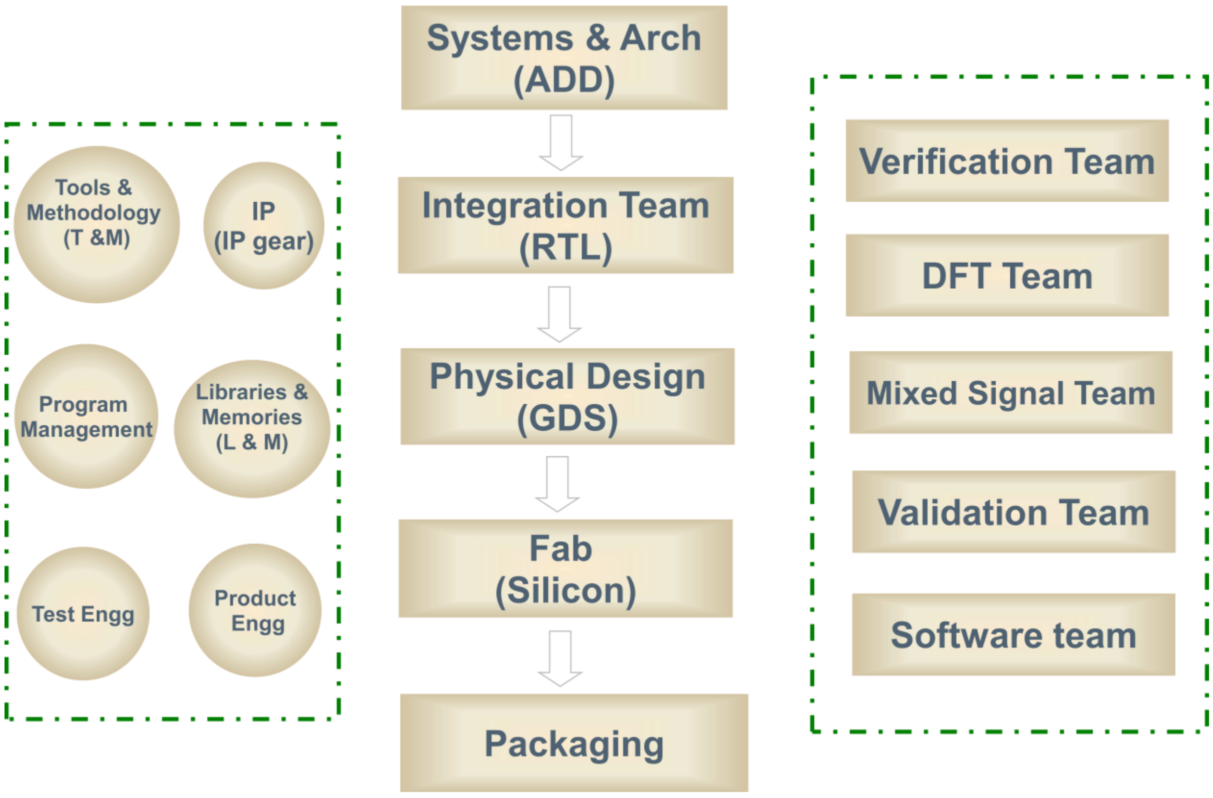


Fig 2.3 SoC Design Flow [1]

2.1.2 IP DELIVERY

An IP is essentially an independent unit catering to certain predefined functionality. Therefore, an IP can be reused in multiple SoCs that require the same functionality to be satisfied. An IP owner is responsible for delivering completely verified IPs (in form of RTL/gates) along with data such as stub netlists (consisting of input/output information in verilog format), timing exceptions, verification patterns, timing constraints etc.

2.1.3 FRONT END INTEGRATION

Front end integration is the process of integrating the design database, based on the design specification of the chip. The process involves generation of top-level chip netlist and providing a testbench in which the module owners can start verification of their modules. Front-end integration is not completed until and unless clearance is given by the verification team on the connectivity of all the modules.

2.1.4 FUNCTIONAL VERIFICATION

Verification is a process of verifying the design for any incompatibility with the desired specifications. The process includes, checking for the basic functionality, checking for the inter-module interaction, checking for the interaction with the cores and the memories, checking for the interaction between the module and the external chip-sets etc. Verification of a chip requires, the design database (RTLs, platforms, top-level netlist, SDFs etc.), stimulus files (written in verilog, C, C++, Vera, SystemC etc.) and a verification environment.

2.1.5 LOGIC SYNTHESIS

The next step is to carry out the logic synthesis. Logic synthesis is the process of converting a design description written in a hardware description language such as Verilog (mostly) or VHDL into an optimized gate-level netlist mapped to a specific technology library. Basically the Synthesis process consists of first translating the RTL code into a valid logic and then optimizing the translated logic.

Synthesis = Translation + Optimization.

Synthesis process basically consists of following:

- Mapping which is just the translation of RTL into a valid logic. Here the combinational logic is best tried for optimization.
- Sequential mapping which normally consists of the mapping of the sequential elements. Here we try to see whether the combinational logic attached to the sequential elements can be reduced to a single sequential element or not. Hence the optimization goes hand-in-hand.
- Design Rule Fixing, which normally consists of fixing of the design rule violations.

2.1.6 DESIGN FOR TEST

With design size growing at exponential rates and product time reducing, there should be some way to test design for functionality, manufacturing defects, AC/DC parameters, and speed performance, once it is translated into chip. With Design For Test we introduce some logic into the design in addition to functional requirement. In return we ensure that design we are producing is testable.

At present design include digital logic, Memories, PLL, debugger and analog circuitry. During manufacturing due to contamination open circuit occurs and because of extra/improper metal following fault arises.

Types of Faults:

- Open /shut fault. (Condition where signal get permanently tied to logic zero/one)
- Delay defect. (Delay between the nodes is not as per requirement).
- Transition defect. (Signal is not able to switch in required time.)

- Bridging. (Two signals shorted because of metal bridge between them)

Through different DFT techniques we ensure all of them are testable and check the quality of manufactured device.

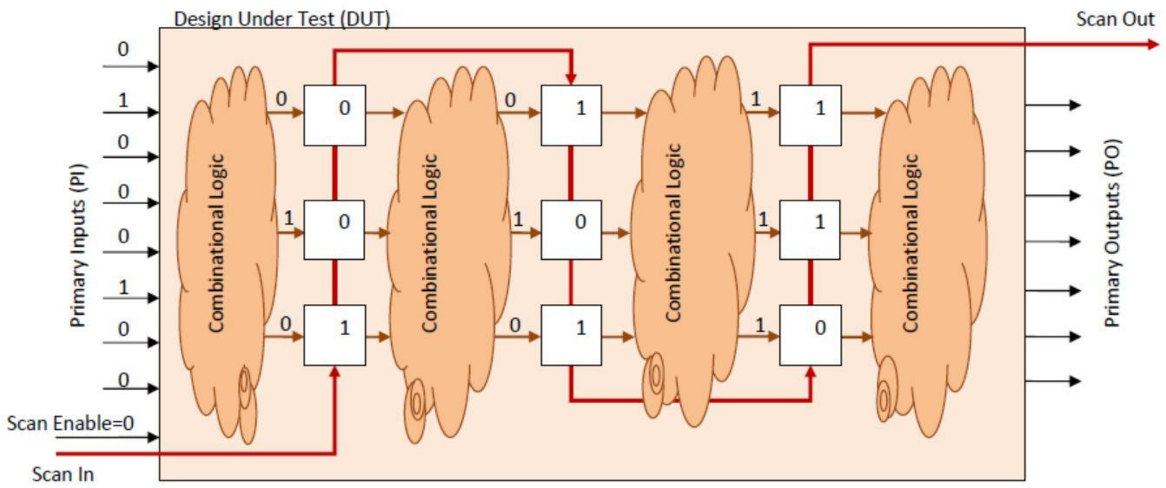


Fig 2.4 Scan Process

2.1.7 STATIC TIMING ANALYSIS

Static Timing Analysis is a method of validating the timing performance of a design by checking all possible paths for timing violations.

It is much faster than dynamic simulation because it is not necessary to simulate the logical operation of the design. It is done both pre and post layout. SDF is a product of post layout STA.

Sequential mapping which normally consists of the mapping of the sequential elements. Here we try to see whether the combinational logic attached to the sequential elements can be reduced to a single sequential element or not. Hence the optimization goes hand-in-hand.

Design Rule Fixing, which normally consists of fixing of the design rule violations.

2.1.8 PHYSICAL DESIGN

In deep submicron designs, interconnect delays are becoming more dominant than cell delays. Special care needs to be given for the interconnect planning. DSM (Deep Sub-Micron) effects are also gaining importance because of shrinking device geometries. DSM effects like signal integrity, electro-migration, power integrity etc. can only be analyzed once design is physically laid-out, estimations done at pre-layout do not hold good for these effects. APR (Automatic Placement and Routing) flow includes clock tree generation, power and signal routing, extraction, timing, signal integrity closure and physical verification. All DSM checks are analyzed and corrected at this stage.

2.1.9 BACKEND INTEGRATION

The very name implies putting together the physical data. In the SoC flow, where different blocks (soft or custom or memories) get delivered by different teams across the globe, backend integration of the chip involves putting together all those blocks in a common ‘platform’ connect those blocks, apply different verification/validation criteria to make sure that everything that has been integrated and the way those things have been integrated, meet the chip specifications and finally release the physical masks to the fabrication unit for mask generation.

The methodology used to integrate the chip with all the different blocks in parallel to the block development process forms the basis for various steps and activities in the backend integration flow and process.

The complete backend integration flow can be divided into 4 main categories

- Floor planning
- Routing
- Integration and physical verification
- Deep sub-micron checks

2.1.9.1 FLOORPLANNING

The objective here is to place the blocks appropriately on the chip so as to ensure a seamless integration process. Floorplanning involved creating/using physical models of the soft/hard blocks and place them on the chip in such a way that the timing and routability of the design are best optimized. The physical models (frames) generated for the soft modules are delivered to the APR team that in parallel works on the block design using those frames and develops the blocks. Similarly, the frames for the hard blocks get delivered by the individual custom block designer and integration team used those frame models to place those blocks on chip

2.1.9.2 ROUTING

It involves using connectivity data out of Floorplanning tool and some additional routing constraints and physically route the design as per logical connectivity. The power grid can either be laid out during floorplanning or during the initial steps of routing. With the power grid implemented, routing constraints are applied to the critical signals (include clocks, high frequency and timing sensitive signals) and design is partially routed. This can also be termed as pre-routing the design. The design with the pre-routes is then taken, global constraints applied to the rest of the signals and routability analysis done. If no major problems with the routability and no major signal integrity violations are encountered, the design is carried forward to the physical verification. If noise problems are found, various techniques are applied while doing signal routing to ensure that noise problems are minimized.

2.1.9.3 INTEGRATION AND VERIFICATION

This stage begins once routing activity is converged to. The routed design is exported and the physical models (frames) used thus far are replaced by the actual blocks. This is referred to as physical integration. The integrated design is then taken thorough the various physical verification checks, foremost among which are LVS and DRC. LVS checks for logical to physical data correlation and DRC checks for any fabrication design rule violations in the design.

2.1.9.4 DEEP SUBMICRON CHECKS

The DSM checks are applied to the design to ensure that the design meets the electrical specifications thereby ensuring complete functionality and proper yield. The deep submicron checks being carried out currently include floating node analysis, IR drop analysis and noise analysis (functional noise analysis). All the checks are thoroughly verified and only once they pass, the design is taken for further final finishing like tiling and seal ring insertion and made ready for the mask shop.

The methodology used to integrate the chip with all the different blocks in parallel to the block development process forms the basis for various steps and activities in the backend integration flow and process.

2.1.10 VERIFICATION

Verification is a process used to demonstrate the functional correctness of a design. It is a process of verifying the design for any incompatibility with the desired specifications. The process includes, checking for the basic functionality, checking for the inter-module interaction, checking for the interaction with the cores and the memories, checking for the interaction between the module and the external chip-sets etc. Verification of a chip requires, the design database (RTLs, platforms, top-level netlist, SDFs etc.), stimulus files (written in verilog, C, C++, Vera, SystemC etc.) and a verification environment.

CHAPTER 3
STATIC TIMING ANALYSIS

Timing and delay information is required because it instills a level of realism into the circuit.

STA is a method of validating the timing performance of a design by checking all possible paths for timing violations. It is much faster than dynamic simulation because it is not necessary to simulate the logical operation of the design. - It computes expected timing of the circuit without performing exact simulation. It is capable of verifying every path. It can also detect glitches, slow paths and skew

Static Timing analysis is preferred because it is much faster than its dynamic counterpart and thus saves a lot of time. Pessimistic measures ensure that the static results provide an accurate sign-off of the chip. Timing analysis is input independent. Worst case delay is found across all input combinations thus ensuring all the possible scenarios in which the SoC may be made to work.

STA requires no vectors, does not miss paths, runtime linear to circuit size, results are conservative, simple library (delay depends on input slew and output load) , easy to extend in incremental o/p for optimisation.

However, it cannot easily handle within die correlation, need many corner to cover all possible cases, too pessimistic if random variations.

In Static Timing Analysis (STA) static delays such as gate delay and net delays are considered in each path and these delays are compared against their required maximum and minimum values. Circuit to be analyzed is broken into different timing paths constituting of gates, flip flops and their interconnections. Each timing path has to process the data within a clock period which is determined by the maximum frequency of operation. Cell delays are available in the corresponding technology libraries. Cell delay values are tabulated based on input transition and fanout load which are characterized by SPICE simulation. Net delays are calculated based on the Wire Load Models (WLM) or extracted resistance R and capacitance C. Wire Load Models (WLM) are available in the Technology File. These values are Table Look Up (TLU) values calculated based on the net fanout length.

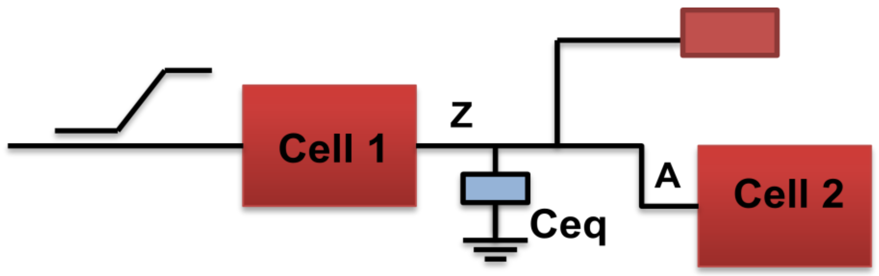


Fig 3.1 Delay dependencies of a cell [2]

Look up tables are present in the library , from which delay information is obtained:

Sample of a look-up table:

```
lu_table_template(table_1){
  index_1(" 0.011, 0.04035, 0.0807, 0.1614, 0.3228, 0.6456, 1.2912, 2.69 ");
  index_2(" 0.0027, 0.004306, 0.005912, 0.009124, 0.015548, 0.028396, 0.054092, 0.083 ");
  variable_1 : input_net_transition;
  variable_2 : total_output_net_capacitance;
 }

cell_fall(table_1){
    values("0.231350, 0.244700, 0.256620, 0.277960, 0.315520, 0.382970, 0.509850, 0.650100",\
      "0.253300, 0.266640, 0.278570, 0.299930, 0.337500, 0.404960, 0.531860, 0.672070",\
      "0.283930, 0.297270, 0.309190, 0.330560, 0.368200, 0.435680, 0.562560, 0.702930",\
      "0.342790, 0.356240, 0.368270, 0.389790, 0.427610, 0.495260, 0.622250, 0.762640",\
      "0.448280, 0.462200, 0.474570, 0.496590, 0.535030, 0.603280, 0.730630, 0.870880",\
      "0.629350, 0.644310, 0.657480, 0.680650, 0.720430, 0.789990, 0.918090, 1.058600",\
      "0.936150, 0.953190, 0.967910, 0.993280, 1.035700, 1.107800, 1.237500, 1.378300",\
      "1.497300, 1.518600, 1.536700, 1.566800, 1.615000, 1.692400, 1.825600, 1.967400");
   }
```

Table 3.1 look up table of a cell in a lib

3.1 Types of Checks Covered

The static timing analyzer will report the following checks (or it can do following analysis):

- Register to Register timing path checks
- Input port to register timing path checks
- Register to output port timing path checks
- Input port to output port timing path checks
- Clock gating checks
- Data to data checks

The wide spread use of STA can be attributed to several factors :

1) The basic STA algorithm is linear in runtime with circuit size, allowing analysis of designs in excess of 10 million instances.
2) The basic STA analysis is conservative in the sense that it will over-estimate the delay of long paths in the circuit and under-estimate the delay of short paths in the circuit. This makes the analysis ”safe” by introducing pessimism in the design , guaranteeing

that the design will function at least as fast as predicted and will not suffer from hold-time violations.

3) The STA algorithms have become fairly mature, addressing critical timing issues such as interconnect analysis, accurate delay modelling, false or multi-cycle paths, etc.
4) Delay characterization for cell libraries is clearly defined, forms an effective interface between the foundry and the design team, and is readily available. In addition to this, the Static Timing Analysis (STA) does not require input vectors and has a runtime that is linear with the size of the circuit.

3.2 Advantages & Disadvantages of STA

3.2.1 Advantages of STA

- All timing paths are considered for the timing analysis. This is not the case in simulation.
- Analysis times are relatively short when compared with event and circuit simulation.
- Timing can be analyzed for worst case, best case simultaneously. This type of analysis is not possible in dynamic timing analysis.
- Static Timing Analysis (STA) works with timing models. STA has more pessimism and thus gives maximum delay of the design. DTA performs full timing simulation. The problem associated with DTA is the computational complexity involved in finding the input patterns (vectors) that produce maximum delay at the output and hence it is slow.

3.2.1Disadvantages of STA

- All paths in the design may not run always in worst case delay. Hence the analysis is pessimistic.
- Clock related all information has to be fed to the design in the form of constraints.
- Inconsistency or incorrectness of these constraints may lead to disastrous timing analysis.
- STA does not check for logical correctness of the design.
- STA is not suitable for asynchronous circuits.

3.3Timing Collaterals

In a design, following are the sources of information:

- Cell definitions and connectivity information comes from netlist (.v)
- Cell’s internal arc delay comes from the library and is dependent on input slew and output net capacitance.

- The Net delay information is derived SPEF (.spef)
- Constraints (.sdc) are delivered by the STA team.

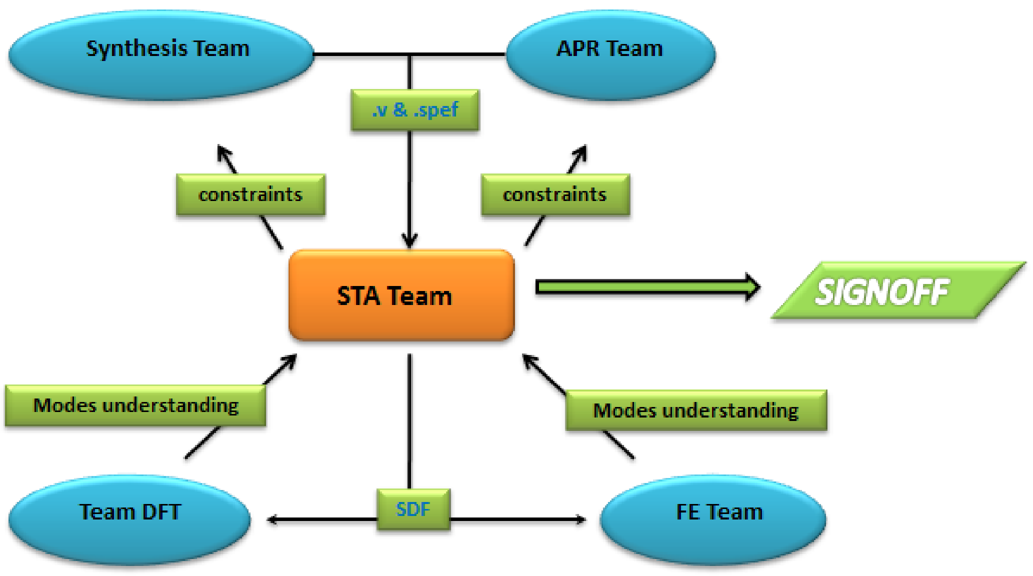


Fig 3.2 Inputs and Outputs w.r.t STA [2]

3.4 Principles of Timing Analysis

3.4.1 Clocking Schemes

Two common types:

- Single phase, used for edge triggered circuits.

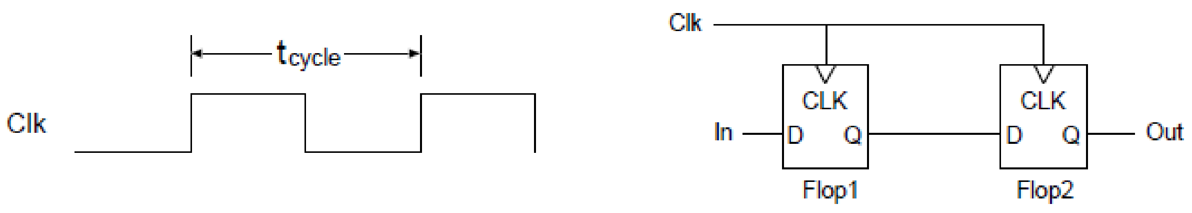


Fig 3.3. Single phase clock

- Two phase non-overlapping, used for edge triggered & level sensitive circuits.

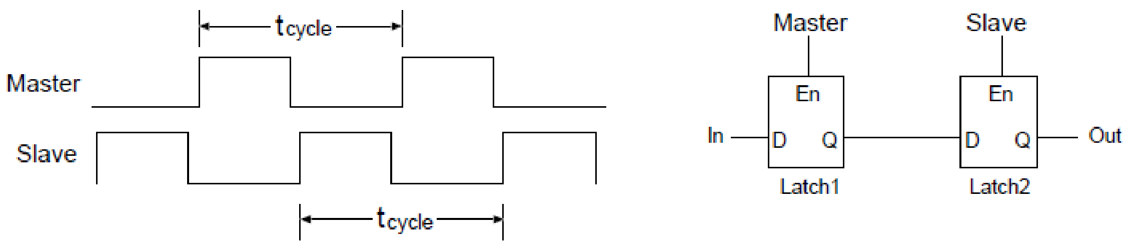
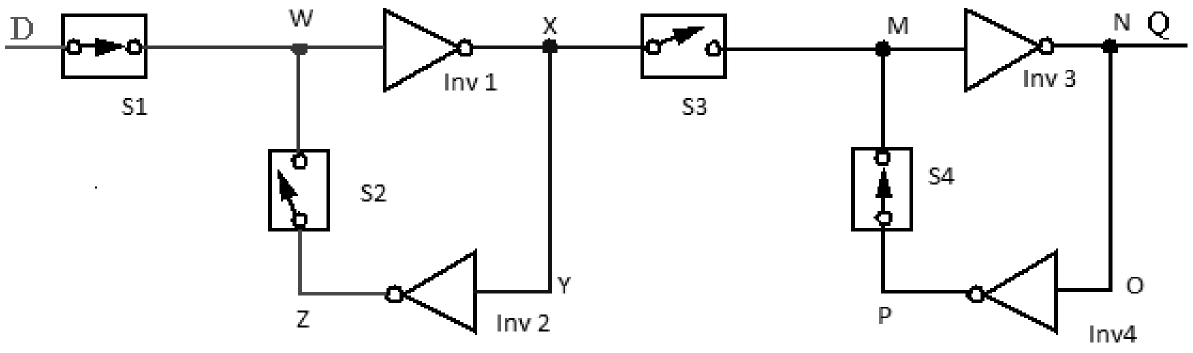


Fig3.4 Two phase non overlapping clock

3.4.2 D Flip Flop and Concept of Setup & Hold Time



.Fig3.5. D Flip flop structure

Operation of DFF :

- Initially assume D = 0 & clk is LOW. Hence W = 0, X = 1, Y = 1 & Z = 0.
- Now lets say that the clk goes HIGH (S1 off). This makes Tx gates DW & MP to switch OFF, and Tx gates ZW & XM to switch ON. Thus the LHS latching circuit is enabled and it latches a 0.
- Also this results in nodes M = 1, N = 0, O = 0, P = 1 and hence Q = 0 (which is what it should be for D = 0). Notice that the Q output arrives at the positive edge of the clk itself. Hence it's a positive edge triggered flop.
- When clk goes LOW, the RHS latching circuit is enabled & there is no change in the output.
- Also any change in D input gets reflected at node Z. (which will appear at the output at next posedge of clk).
- If D were to go logic HIGH at any time, then this change would reflect at node Z when clk is LOW, & would appear at the output when clk goes HIGH.
- Concept of SETUP TIME : Now it can be realized that before the switches S2 and S3 are closed (CLK goes high), the data needs to be at Z to be able to be retained in the inv1-inv2 loop. This extra time that the data needs to be able to reach from D-W-X-Y-Z, before the active edge of the clock, is called Setup Time.
- Concept of HOLD TIME : It is also known that any switch (say a transmission gate) will have a finite on and off time. Thus the switch S1 may take a finite time, after the CLK goes high (active clock edge) to be completely off, In this period, an invalid or undesired data may arrive at W and may get unintentionally stored in the loop. Thus data needs to be steady a certain time, equivalent to traversal time from D-W, after the active clock edge, to avoid this scenario. This time is called the Hold Time.

3.4.3 Setup and Hold Time

All latches & flip-flops should have setup & hold times specified with respect to the clock/enable pin.

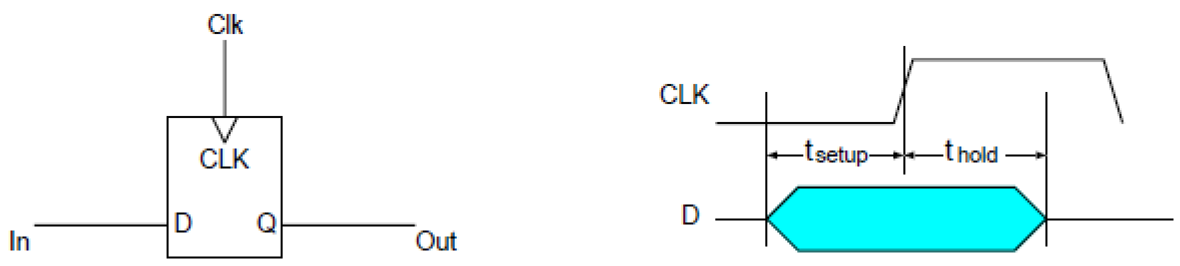
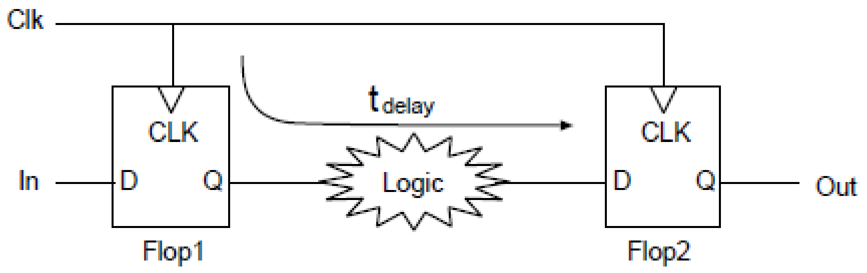


Fig 3.6. Setup and Hold times

3.4.3.1 Setup

- The length of time a signal needs to be valid before the capture event to ensure the value will be latched.
- Setup times are checked using Max signal delays, Min clock delays.



tdelay <= tcycle - Flop2 tsetup

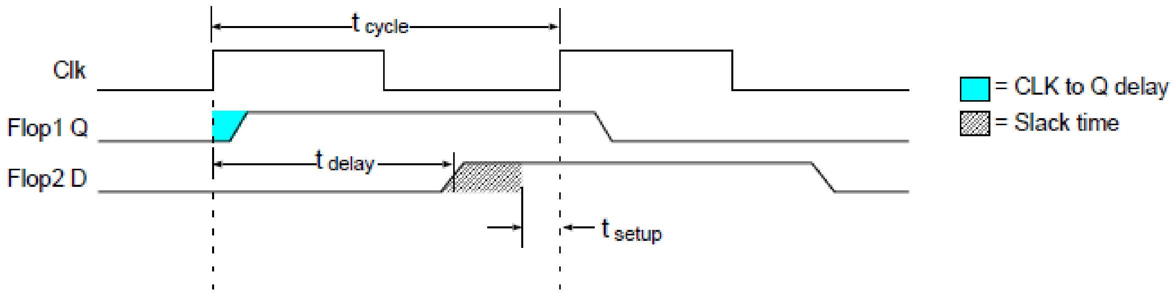


Fig 3.7. Timing Diagram for setup

3.4.3.2 Hold

- The length of time a signal needs to be valid after the capture event to prevent a race through occurring.
- Hold times are checked using Min signal delays, Max clock delays.

tdelay >= Flop2 thold (When data is getting launched at flop1 , flop2 is processing some data as well at same clock edge and needs some time to stabilise the data , i.e. its hold time therefore data processed at flop1 shouldn’t arrive at flop2 before this stabilising period or else meta-stability might occur leading to further delays and inaccurate processing **)**

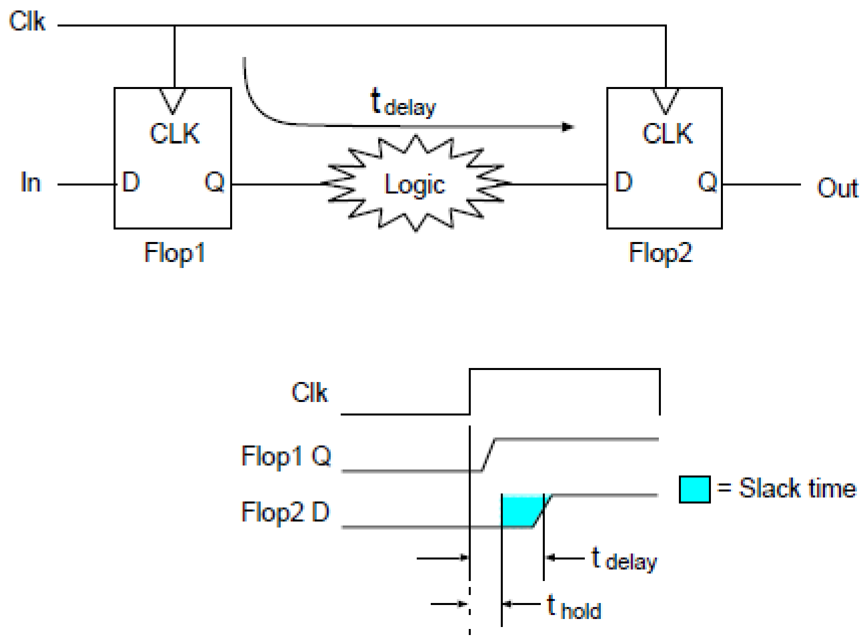


Fig 3.8. Timing Diagram for hold

3.4.4 Clock Skew and Effect on Timing checks

Skew is the difference in arrival of clock at two consecutive pins of a sequential element is called skew. Clock skew is the variation at arrival time of clock at destination points in the clock network. The difference in the arrival of clock signal at the clock pin of different flops.

Skew can be positive or negative. When data and clock are routed in same direction then it is **Positive skew.** When data and clock are routed in opposite direction then it is **negative skew.**

3.4.4.1 Positive Skew

If capture clock comes late than launch clock then it is called positive skew. Clock and data both travel in same direction. When data and clock are routed in same direction then it is **Positive skew.** Important thing to note is:

- positive skew can lead to hold violation
- Positive skew improves setup time.

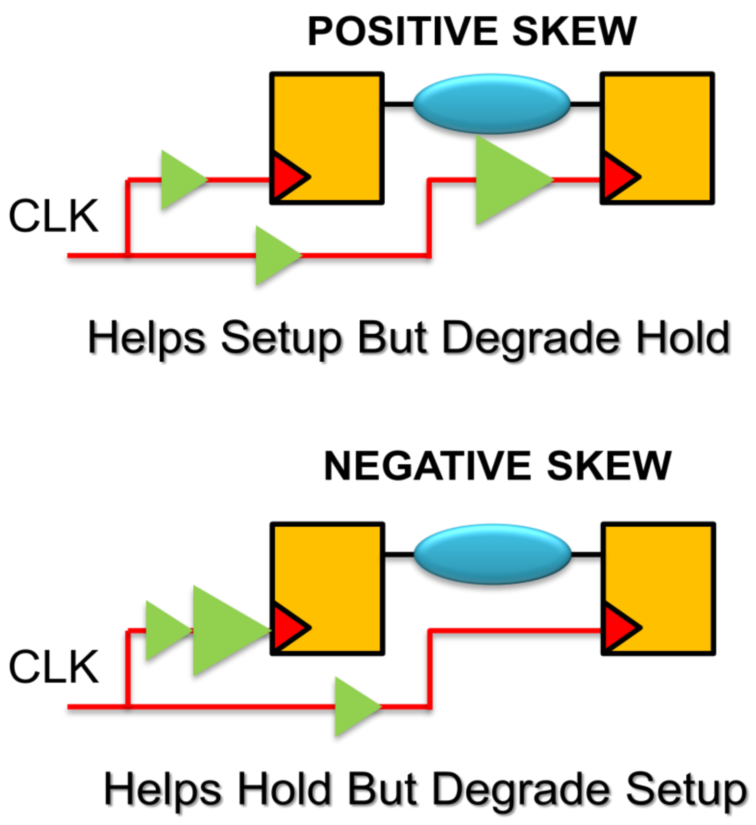


Fig 3.9. Positive and Negative skew

3.4.4.2 Negative Skew

If capture clock comes early than launch clock it is called –ve skew. Clock and data travel in opposite direction. When data and clock are routed in opposite then it is **negative skew.** Negative skew can lead to setup violation. -ve skew improves hold time. (Effects of skew on setup and hold will be discussed in detail in forthcoming articles)

Effect of Clock Skew on Setup Analysis

Setup time

- Positive clock skew allows the data delay allowance to increase.
- Negative clock skew decreases the data delay allowance.

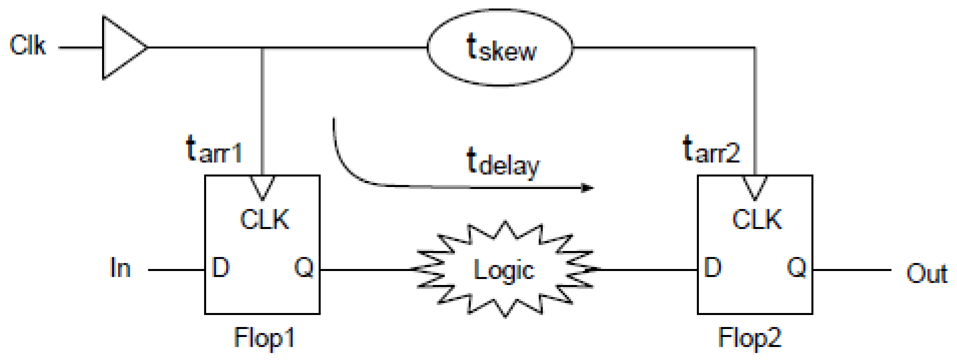


Fig 3.10 Analysing clock skew effect on setup

- Setup slack = tcycle + (tarr2 - tarr1) - tsetup – tdelay
- tskew = tarr2 – tarr1

Hold time:

- Positive clock skew increases the risk of hold time violations.
- Negative clock skew reduces the risk of hold time violations.
- Hold slack = tarrdata - thold – tskew

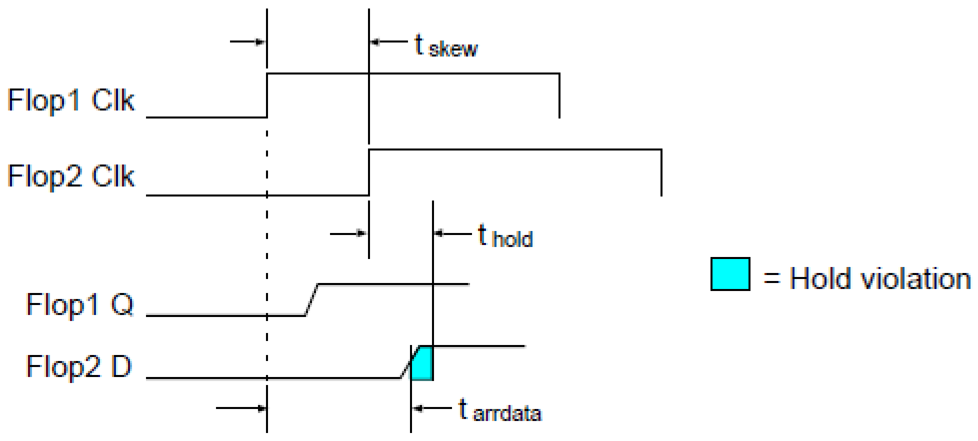


Fig 3.11. Analysing clock skew effect on hold

3.4.5 Minimum and Maximum Pulse Width

Apart from the setup, hold and clock gating checks, some other checks are also done by the tool carrying out timing analysis :

Minimum high pulse width: The amount of time, after the rising edge of a clock, that clock signal of a clocked device must remain stable

Minimum low pulse width: The amount of time, after the falling edge of a clock, that clock signal of a clocked device must remain stable

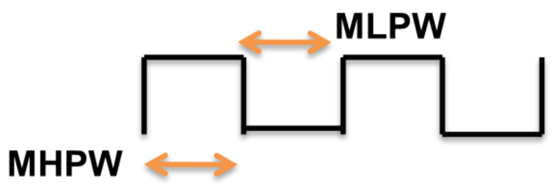


Fig 3.12 minimum and maximum pulse width [3]

3.4.6 Recovery and Removal Time

RECOVERY: The minimum time that an asynchronous control input pin must be stable after being de-asserted and before the next clock transition.

REMOVAL: The minimum time, that an asynchronous control input pin must be stable before being deasserted and after the previous clock transition

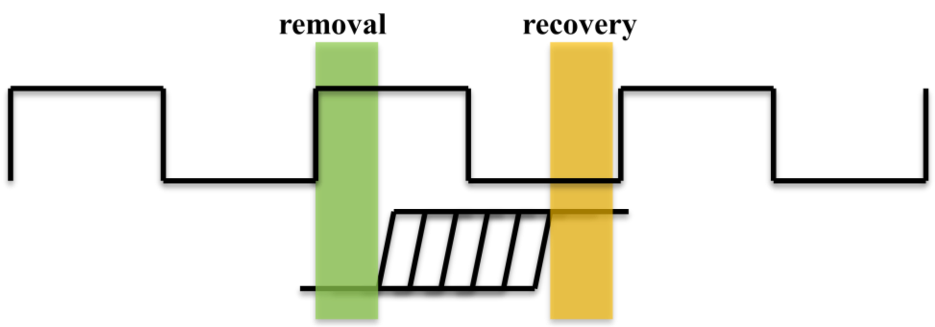


Fig 3.13 Recovery and Removal Time[4]

Basic Idea is to have all registers to come out of reset in the same clock cycle

3.4.7 Latches

Latches have two states; transparent & opaque.

Data is allowed to flow during the transparent state and cannot flow during the opaque state. The state is controlled by an enable signal. Latches are known as level sensitive devices.

This example shows an active high enable latch.

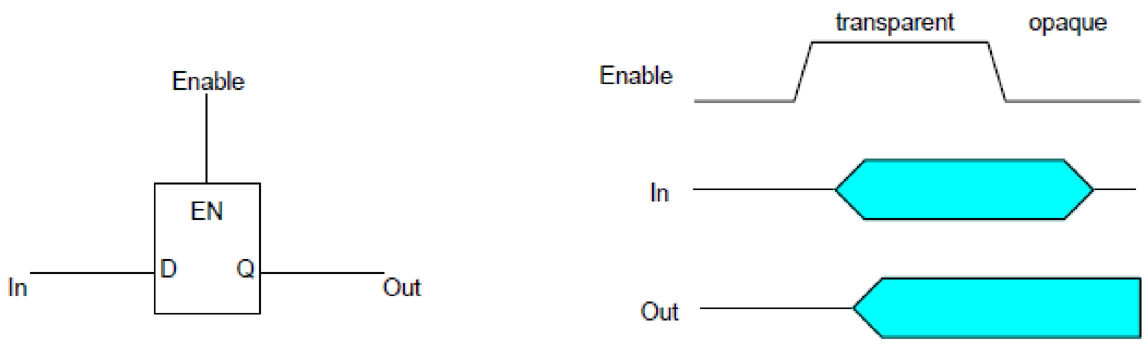


Fig 3.14. Latch

When performing timing analysis two delay paths must be considered:

- Data setup before Enable active edge – Enable to Q delay
- Data setup after Enable active edge – D to Q delay

The transparent state of latches allows time to be borrowed from the previous cycle.

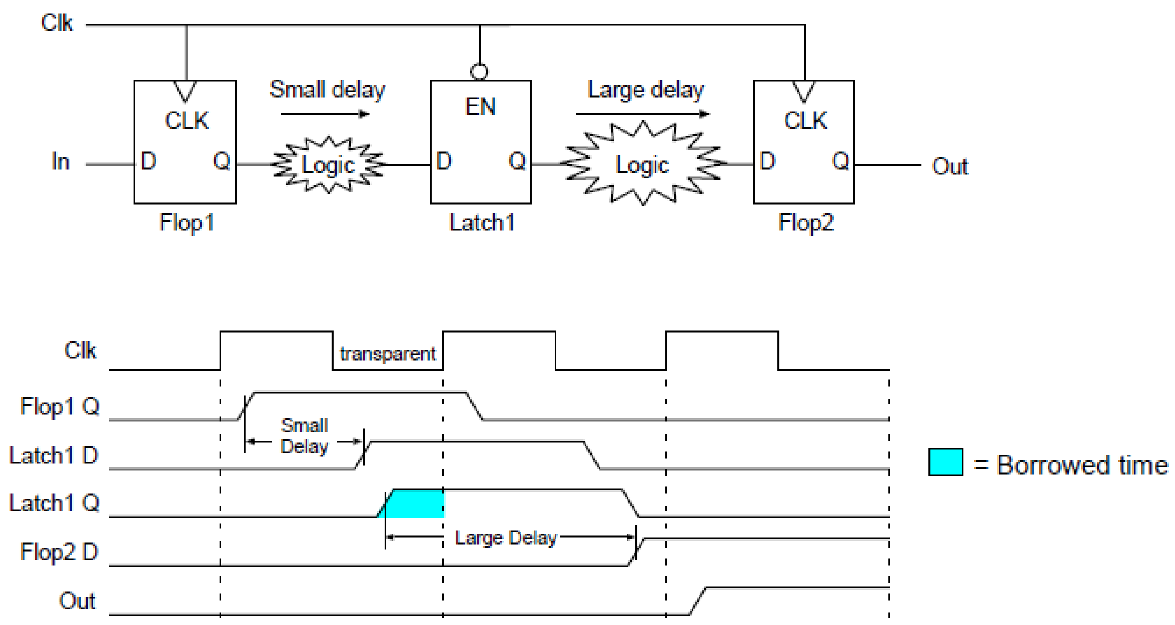


Fig 3.15 . Time Borrowing

3.4.8 Gated Clocks

Clock tree consume more than 50 % of dynamic power.
The components of this power are:
1) Power consumed by combinatorial logic whose values are changing on each clock edge
2) Power consumed by flip-flops and
3) The power consumed by the clock buffer tree in the design.

It is good design idea to turn off the clock when it is not needed. Automatic clock gating is supported by modern EDA tools. They identify the circuits where clock gating can be inserted.

RTL clock gating works by identifying groups of flip-flops which share a common enable control signal. Traditional methodologies use this enable term to control the select on a multiplexer connected to the D port of the flip-flop or to control the clock enable pin on a flip-flop with clock enable capabilities. RTL clock gating uses this enable signal to control a clock gating circuit which is connected to the clock ports of all of the flip-flops with the common enable term. Therefore, if a bank of flip-flops which share a common enable term have RTL clock gating implemented, the flip-flops will consume zero dynamic power as long as this enable signal is false.

There are two types of clock gating styles available. They are:

1) Latch-based clock gating
2) Latch-free clock gating.

3.4.8.1 Latch free clock gating

The latch-free clock gating style uses a simple AND or OR gate (depending on the edge on which flip-flops are triggered). Here if enable signal goes inactive in between the clock pulse or if it multiple times then gated clock output either can terminate prematurely or generate multiple clock pulses. This restriction makes the latch-free clock gating style inappropriate for our single-clock flip-flop based design.

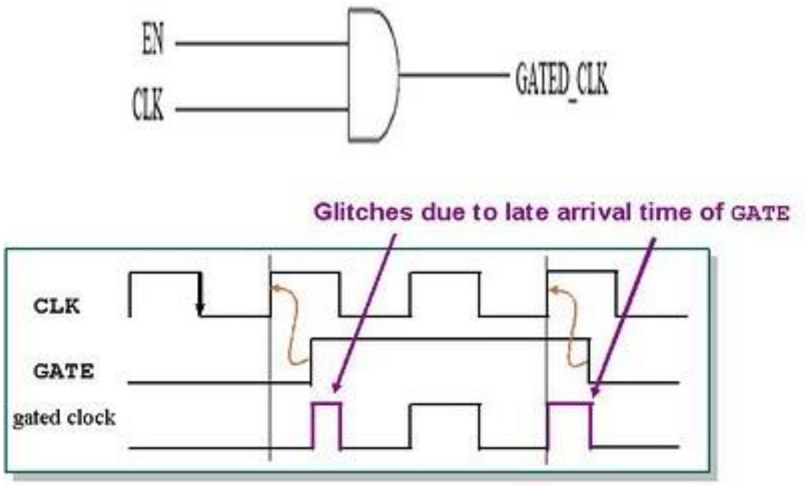


Fig 3.16. Latch free clock gating

AND type clock gating

Here it is necessary that the enable signal should be asserted only when the clock is low else glitches will appear. This is ensured by taking the enable signal from the output of a negative level-sensitive latch clocked by the clock to be gated.

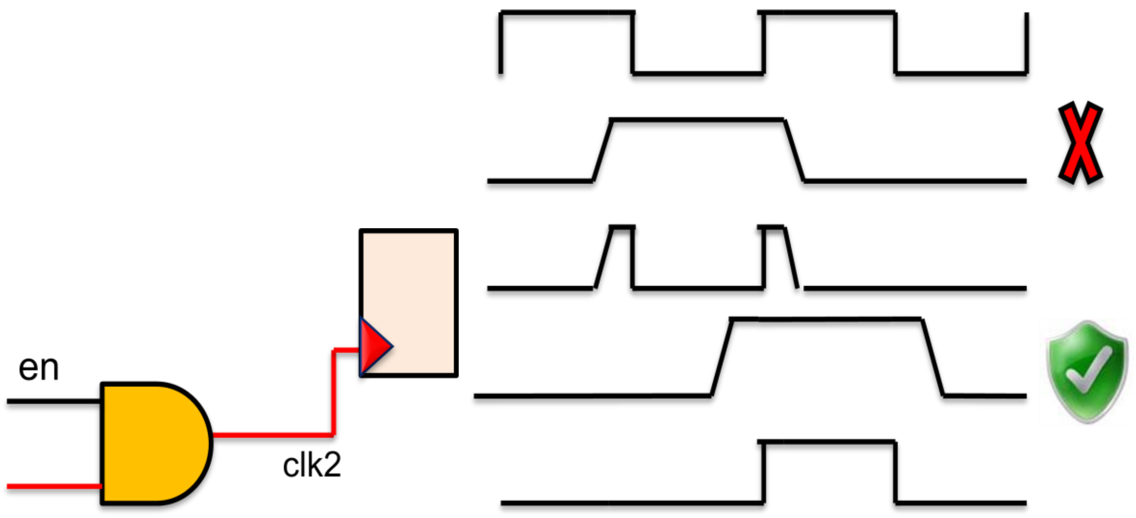


Fig 3.17 AND type clock gating [5]

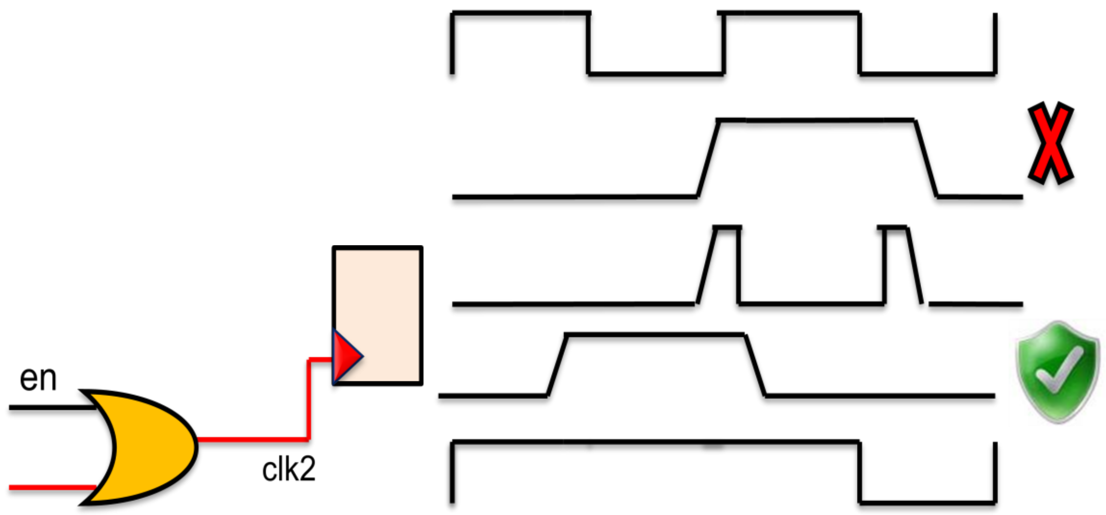


Fig 3.18 OR type clock gating [5]

OR type clock gating

Here it is necessary that the enable signal should be asserted only when the clock is high else glitches will appear. This is ensured by taking the enable signal from the output of a positive level-sensitive latch clocked by the clock to be gated.

3.4.8.2 Latch based clock gating

The latch-based clock gating style adds a level-sensitive latch to the design to hold the enable signal from the active edge of the clock until the inactive edge of the clock. Since the latch captures the state of the enable signal and holds it until the complete clock pulse has been generated, the enable signal need only be stable around the rising edge of the clock, just as in the traditional ungated design style.

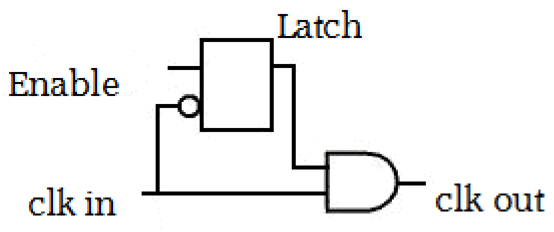


Fig 3.19. Latch based clock gating

Specific clock gating cells are required in library to be utilized by the synthesis tools. Availability of integrated clock gating cells and automatic insertion by the EDA tools makes it simpler method of low power technique. Advantage of this method is that clock gating does not require modifications to RTL description.

Glitchfree gating can be achieved by a latch

3.5. Timing Paths

Timing path is defined as the path between start point and end point where start point and end point is defined as follows:

- **Start Point:**
  All input ports or clock pins of a sequential element are considered as valid start point.
- **End Point:**
  All output port or D pin of sequential element is considered as End point.

For STA design is split into different timing path and each timing path delay is calculated based on gate delays and net delays. In timing path data gets launched and traverses through combinational elements and stops when it encounter a sequential element. In any timing path, in general (there are exceptions); delay requirements should be satisfied within a clock cycle. In a timing path wherein start point is sequential element and end point is sequential element, if these two sequential elements are triggered by two different clocks(i.e. asynchronous) then a common least common multiple (LCM) of these two different clock periods should be considered to find the launch edge and capture edge for setup and hold timing analysis.

3.5.1 Different Timing Paths

Any synchronous design is split into various timing paths and each timing path is verified for its timing requirements. In general four types of timing paths can be identified in a synchronous design. They are:

- Input to Register
- Input to Output
- Register to Register
- Register to Output

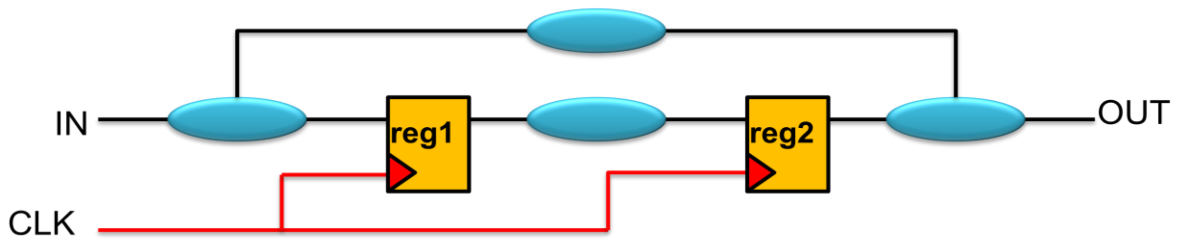


Fig 3.20. Timing Paths [6]

1) **Input to Output:**
   It starts at input port and ends at output port. This is pure combinational path. You can hardly find this in a synchronous design.

2) **Input to Register:**
   Semi synchronous; Register is controlled by the clock. Input data can come at any time.

3) **Register to Register:**

Purely sequential; both starting and ending flops are controlled by the clock.

4) **Register to Output:**
Data can come at any point of time.

Clock path
The path wherein clock traverses is known as clock path. Clock path can have only clock inverters and clock buffers as its element. Clock path may be passed trough a "gated element" to achieve additional advantages. In this case, characteristics and definitions of the clock change accordingly. We call this type of clock path as "gated clock path". The process of "clock gating" has main advantage of dynamic power saving.

Data path
The path wherein data traverses is known as data path. Data path is a pure combinational path. It can have any basic combinational gates or group of gates.

Launch path
Launch path is part of clock path. Launch path is launch clock path which is responsible for launching the data at launch flip flop. Launch path and data path together constitute arrival time of data at the input of capture register.

Capture path
Capture path is part of clock path. Capture path is capture clock path which is responsible for capturing the data at capture flip flop. Capture clock period and its path delay

3.6 CORNERS

Corners – A design needs to be qualified across various conditions specified as corners. It can made either pessimistic of optimistic conditions. Corners define differences due to process inaccuracies, temperature and other parameter variations.

3.6.1 PVT Corners

PVT corners consist of process voltage and temperature .

As processing technology migrates into sub-90 nm region, design performance can be affected by factors that were considered secondary before. One of such factors is the Inversed Temperature Dependence (ITD) effect . When a circuit is operating in low voltage, the propagation delay of a cell may decrease as the temperature increases . The reason behind ITD effect is due to the temperature effect on the threshold voltage, V_{TH}. As supply voltage (V_{DD}) scaled, the value of $|V_{GS} – V_{TH}|$, the absolute difference between transistor gate to source voltage and threshold voltage, decreases. The smaller $|V_{GS} – V_{TH}|$ makes saturation current more sensitive to change in V_{TH}, which decreases as the increase of temperature. The smaller V_{TH} incurs more current that makes the device switching faster. On the other hand, transition delay is also proportional to the electron mobility, which decreases as the temperature rises. Hence the device performance depends on the racing condition of electron mobility and V_{TH} together as temperature rises. Traditionally, timing is signed off at two extreme temperature corners, one representing the best case and the other representing the worst case. With ITD, the highest sign-off temperature can no longer guarantee the worst case, and vice versa. This poses a serious problem to the timing sign-off methodology, i.e. it is possible that the worst-case temperature occurs at some intermediate point and finding this point can be quite difficult. Due to the goal of having low power design, modern designs are implemented by standard cells with high V_{TH} extensively. Coupling with the ITD effect, it is necessary to understand the impact on sign-off methodology.

Thus until 90 nm technology there were only two pvt_corners worst and best

Worst corresponding to more delay with worst process, lower voltage and higher temperature

Best corresponding to less delay with best process, higher voltage and lower temperature.

But below 90nm temperature inversion plays a critical role and hence designs are checked in four PVT corners worst , worst_cold, best, best_hot.

3.6.2 RC Corners

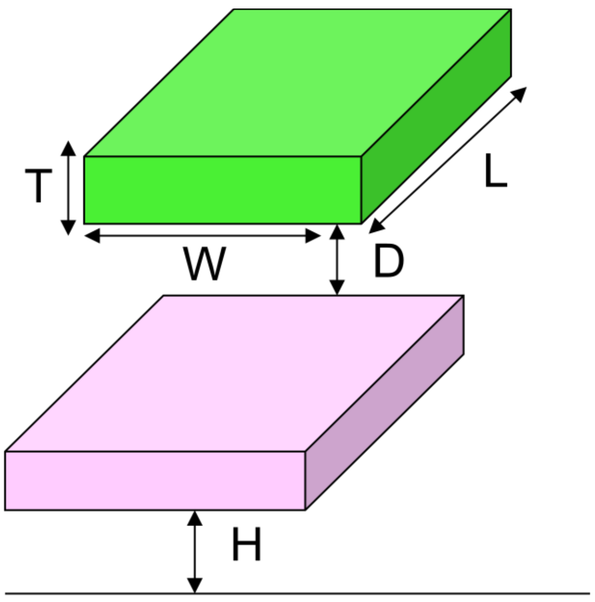


Fig 3.21 Parameters for RC corners

Process Variation parameter for RC corners

- Metal width variation (W)
- Metal thickness variation (T)
- IMD (inter/intra metal dielectric thickness) (D)

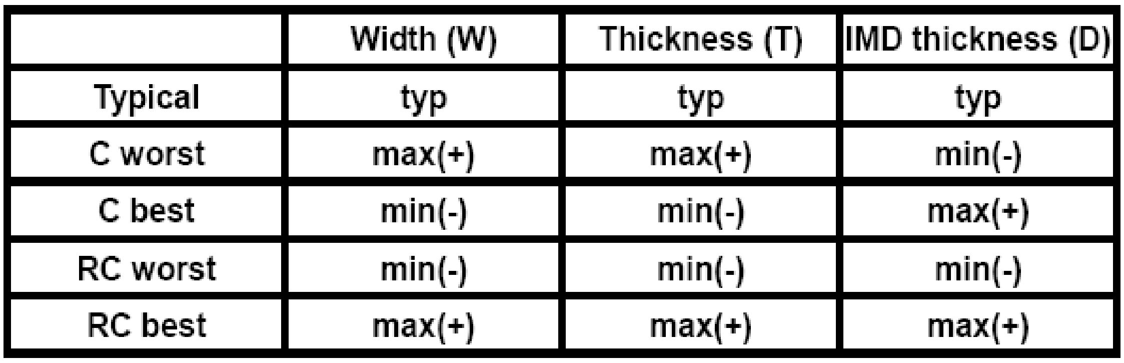
R Variation

- R varies with W and T

C Variation

- Cb varies with H, W and D
- Cc varies with W and T
    - Both Cc and Cb vary with Inter/Intra layer dielectric constant

Variations are captured with multiple process corners



	Width (W)	Thickness (T)	IMD thickness (D)
Typical	typ	typ	typ
C worst	max(+)	max(+)	min(-)
C best	min(-)	min(-)	max(+)
RC worst	min(-)	min(-)	min(-)
RC best	max(+)	max(+)	max(+)

Table 3.2 R,C dependence on device geometry

- Temperature is an additional variable in Extraction
  - CMAX – Max C, Min R (2 SPEF, 1 for 125 and 1 for -40C)
    This is a Setup critical corner.
  - CMIN – Min C, Max R (1 SPEF)
    This is a hold critical corner
  - XTLK – Max Cc, Min Cg, Min R (1 SPEF)
    Optimistic parasitic (R) is taken here. It is used to find violations where launch and capture path are sensitive to cell/interconnects differences.
  - DLY – Max Cg, Min Cc, Max R (1 SPEF)
    Pessimistic parasitic (R) is taken here. It is used to find violations where launch and capture path are sensitive to cell/interconnects differences. It can identify violations where long interconnects having a high R exceeds a drivers effective resistance

3.7 Constraints

Constraints mimic the timing requirements of a design. It defines the TIMING specification of the design. Constraints are used for correct design implementation and also for validating the predictable timing performance of the design.

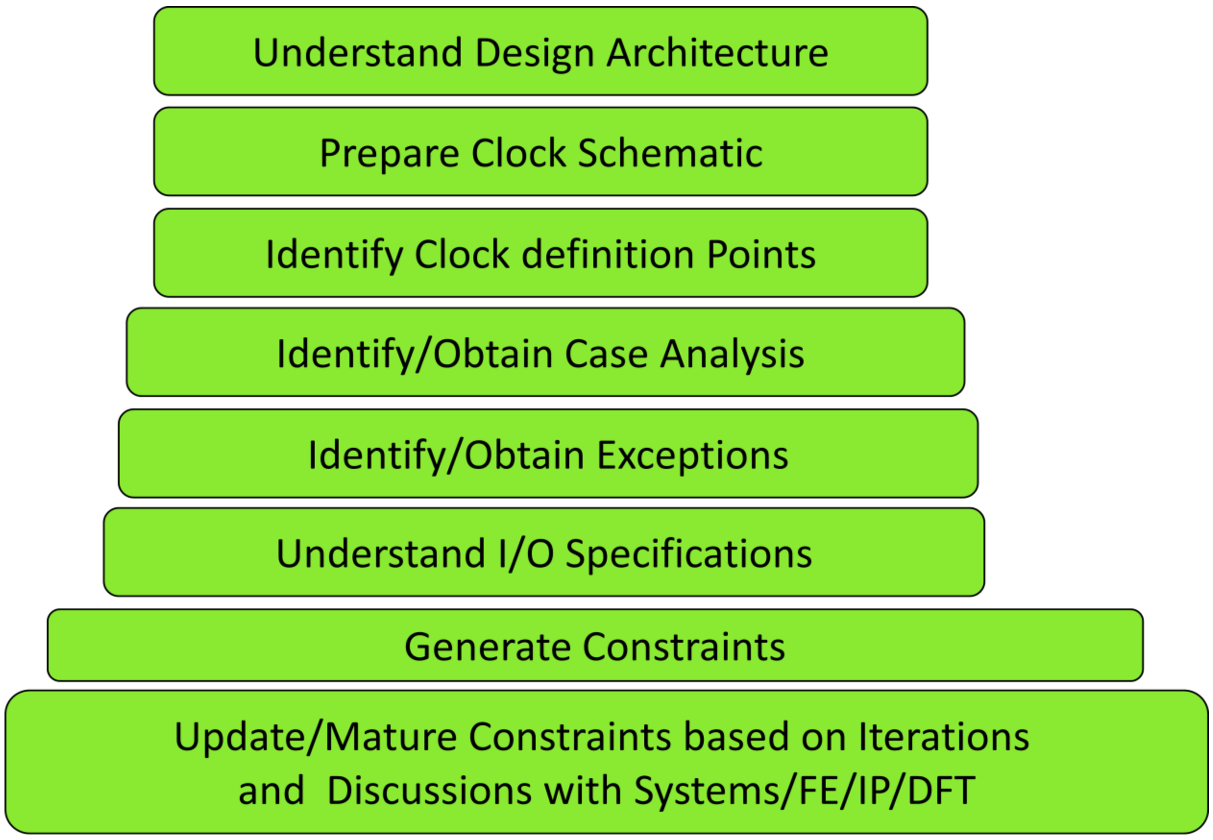

Understand Design Architecture
Prepare Clock Schematic
Identify Clock definition Points
Identify/Obtain Case Analysis
Identify/Obtain Exceptions
Understand I/O Specifications
Generate Constraints
Update/Mature Constraints based on Iterations and Discussions with Systems/FE/IP/DFT

Fig 3.22 Constraint Development Flow[3,7]

3.7.1 Constraint Development Process

Constraints development begins right from understanding the Architecture Definition Document (ADD). It involves understanding the overall design and the clock architecture is a must. Discussion are carried out with the System & Architecture team for overall design requirements, specially Clocking and I/O specifications(PES) for better clarity. Also required is the understanding of IP/Block details through Integration/Block Guide of the IPs. Discussions with FE/IP/DFT team are held and inputs are taken from Integration/Block/DFT guide for obtaining mode selection, clock definitions, case analysis, timing exceptions (false_path / multi-cycle path etc.)

The constraints gradually mature through iterations and further discussions/review.

3.7.2 Clock Definitions

Rising and falling edge of the clock

For a positive edge triggered design positive (or rising) edge is called ‘leading edge’ whereas negative (or falling) edge is called ‘trailing edge’.

For a negative edge triggered design negative (or falling) edge is called ‘leading edge’ whereas positive (or rising) edge is called ‘trailing edge’.

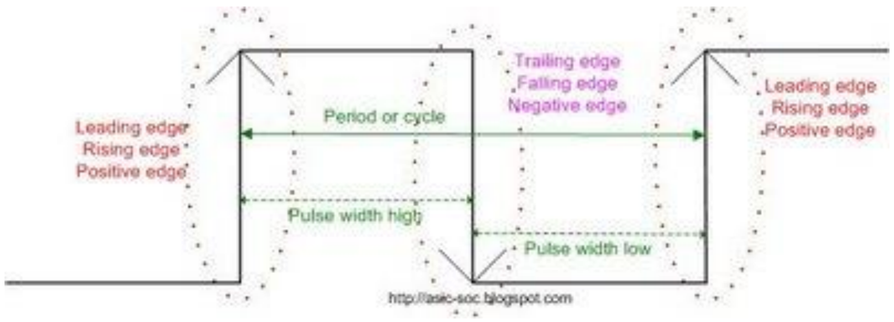


Fig 3.23. Basic Clock Pulse

Minimum pulse width of the clock can be checked in ETS by using commands given below:

set_min_pulse_width -high 2.5 [all_clocks]

set_min_pulse_width -low 2.0 [all_clocks]

These checks are generally carried out for post layout timing analysis. Once these commands are set, ETS checks for high and low pulse widths and reports any violations.

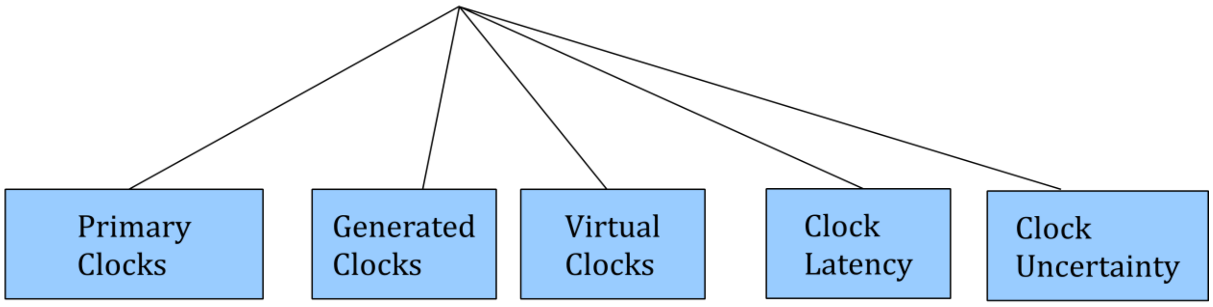


Fig 3.24 Components of clock definition

- **Capture Clock Edge**

  The edge of the clock for which data is detected is known as capture edge.

- **Launch Clock Edge**

  This is the edge of the clock wherein data is launched in previous flip flop and will be captured at this flip flop.

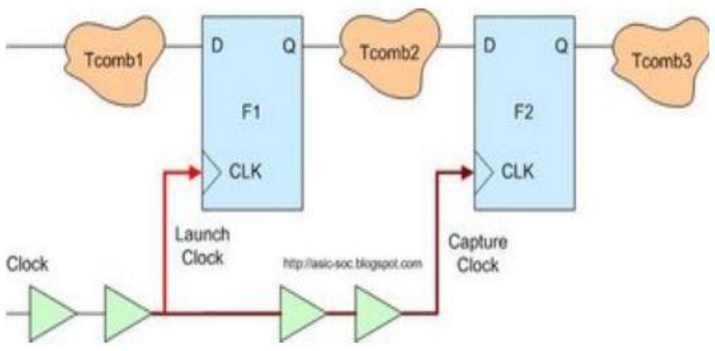


Fig 3.25. Launch Clock and Capture Clock

3.7.2.1 Uncertainty

Clock uncertainty is the time difference between the arrivals of clock signals at registers in one clock domain or between domains.

Pre-layout and Post-layout Uncertainty

Pre CTS uncertainty is clock skew, clock Jitter and margin. After CTS skew is calculated from the actual propagated value of the clock. We can have some margin of skew + Jitter [8, 9].

Uncertainties at various stages in design as below:

PreCTS
Uncertainty = Skew + Jitter + Tool margin + Ocv margin + Delay noise margin

Post CTS
Uncertainty = Jitter + Tool margin + Ocv margin + Delay noise margin

After OCV stage
Uncertainty = Jitter + Tool margin + Delay noise margin

Noise Analysis
Uncertainty = Jitter + Tool margin

3.7.2.2 Clock latency

Latency is the delay of the clock source and clock network delay. Clock source delay is the time taken to propagate from ideal waveform origin point to clock definition point. Clock network latency is the delay from clock definition point to register clock pin.

Pre CTS Latency and Post CTS Latency
Latency is the summation of the Source latency and the Network latency. Pre CTS estimated latency will be considered during the synthesis and after CTS propagated latency is considered.

Source Delay or Source Latency

It is known as source latency also. It is defined as "the delay from the clock origin point to the clock definition point in the design". Delay from clock source to beginning of clock tree (i.e. clock definition point).

The time a clock signal takes to propagate from its ideal waveform origin point to the clock definition point in the design.

Network Delay (latency) or **Insertion Delay**
It is also known as Insertion delay or Network latency. It is defined as "the delay from the clock definition point to the clock pin of the register". The time clock signal (rise or fall) takes to propagate from the clock definition point to a register clock pin.
Figure below shows example of latency for a design without PLL.

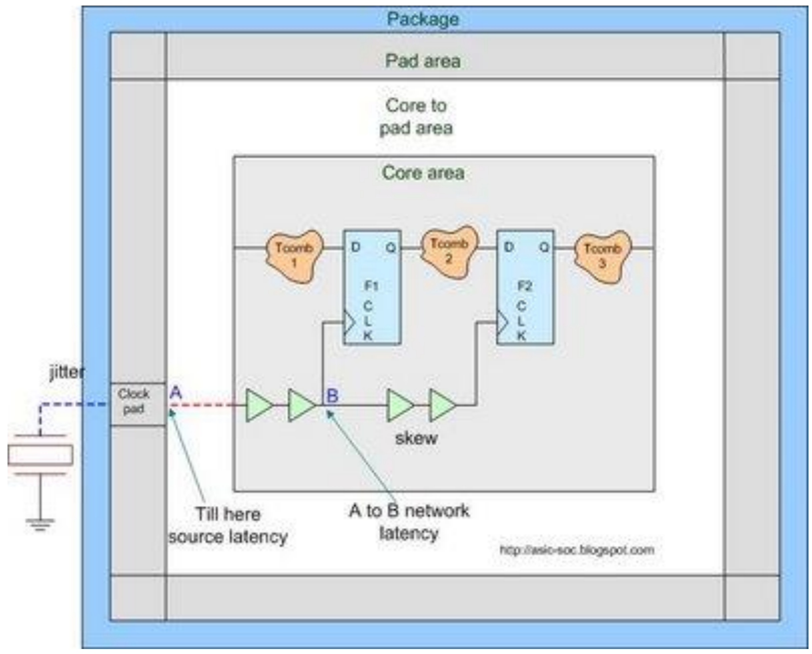


Fig 3.26. Latency for a design without PLL

The latency definitions for designs with PLL are slightly different.

Figure below shows latency specifications of such kind of designs.

Latency from the PLL output to the clock input of generated clock circuitry becomes source latency. From this point onwards till generated clock divides to flops is now known as network latency. Here we can observe that part of the network latency is clock to q delay of the flip flop (of divide by 2 circuit in the given example) is known value.

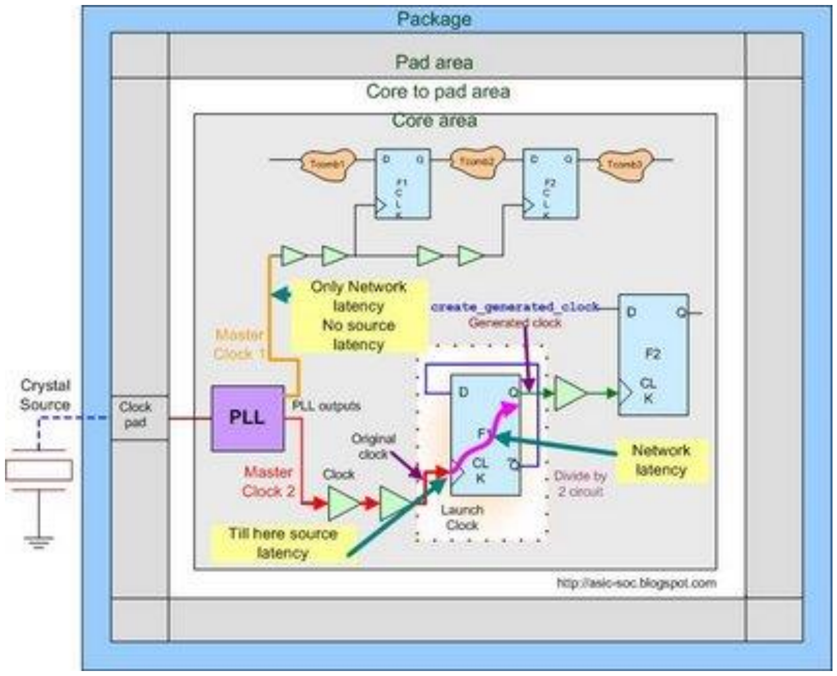


Fig 3.27. Latency for a design with PLL

Jitter

Jitter is the short-term variations of a signal with respect to its ideal position in time. Jitter is the variation of the clock period from edge to edge. It can vary +/- jitter value.

From cycle to cycle the period and duty cycle can change slightly due to the clock generation circuitry. Jitter can also be generated from PLL known as PLL jitter. Possible jitter values should be considered for proper PLL design. Jitter can be modeled by adding uncertainty regions around the rising and falling edges of the clock waveform.

Sources of Jitter

Common sources of jitter include:

- Internal circuitry of the phase-locked loop (PLL)
- Random thermal noise from a crystal
- Other resonating devices
- Random mechanical noise from crystal vibration
- Signal transmitters
- Traces and cables
- Connectors

- Receivers

Multiple Clocks

If more than one clock is used in a design, then they can be defined to have different waveforms and frequencies. These clocks are known as multiple clocks. The logics triggered by each individual clock are then known as "clock domain". If clocks have different frequencies there must be a base period over which all waveforms repeat.

Base period is the least common multiple (LCM) of all clock periods

Asynchronous Clocks

In multiple clock domains, if these clocks do not have a common base period then they are called as asynchronous clocks. Clocks generated from two different crystals, PLLs are asynchronous clocks. Different clocks having different frequencies generated from single crystal or PLL are not asynchronous clocks but they are synchronous clocks.

Generated clocks

Generated clocks are the clocks that are generated from other clocks by a circuit within the design such as divider/multiplier circuit. Static timing analysis tools such as ETS will automatically calculate the latency (delay) from the source clock to the generated clock if the source clock is propagated and you have not set source latency on the generated clock.

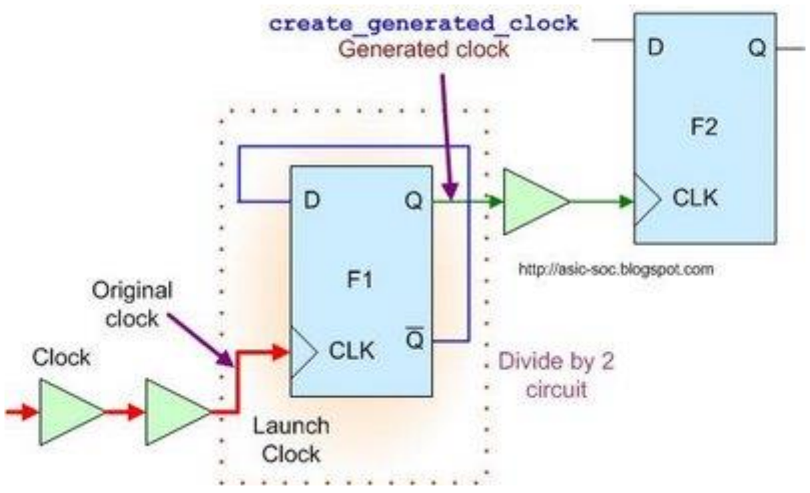
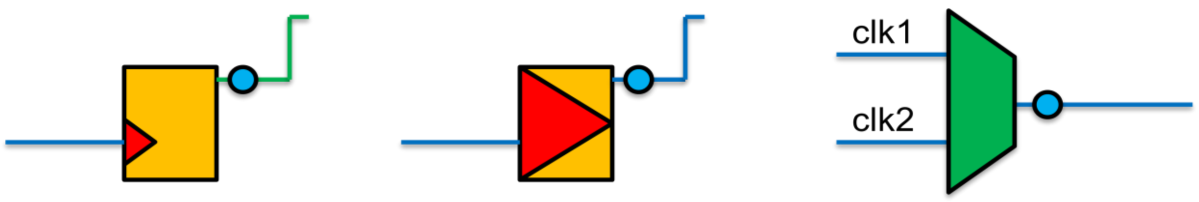


Fig 3.28. Generated clock

'Clock' is the master clock and new clock is generated from F1/Q output. Master clock is defined with the constraint 'create_clock'. Unless and until new generated clock is defined as 'generated clock' timing analysis tools won't consider it as generated clock. Hence to accomplish this requirement use "create_generated_clock" command. 'CLK' pin of F1 is

now treated as clock definition point for the new generated clock. Hence clock path delay till F1/CLK contributes source latency whereas delay from F1/CLK contributes network latency.
Generated Clocks: generated clocks are defined at divider registers, clock gating cells and multiplexers in the clock path.



Divider register clock gating cell multiplexers

Fig 3.29 Clock Generation points

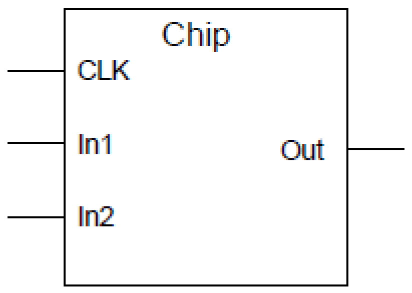
Virtual Clocks

Virtual clock is the clock which is logically not connected to any port of the design and physically doesn't exist. A virtual clock is used when a block does not contain a port for the clock that an I/O signal is coming from or going to. Virtual clocks are used during optimization; they do not really exist in the circuit.

Virtual clocks are clocks that exist in memory but are not part of a design. Virtual clocks are used as a reference for specifying input and output delays relative to a clock. This means there is no actual clock source in the design. Assume the block to be synthesized is "Block_A". The clock signal, "VCLK", would be a virtual clock. The input delay and output delay would be specified relative to the virtual clock.

3.7.3 I/O constraints

Used to define a timing environment around a chip or module. Chip constraints derived from specification. Module constraints derived from specification/analysis.
Defined in terms of input/output delays or input/output setup & hold times with respect to an input, internal or output clock.



In1 setup time w.r.t. CLK	=	5ns
In1 hold time w.r.t. CLK	=	1ns
In2 setup time w.r.t. CLK	=	7.5ns
In1 hold time w.r.t. CLK	=	0.5ns
Out max delay from CLK	=	4.0ns
Out min delay from CLK	=	1.5ns

Fig 3.30 pin constraints [10]

Global Constraints

For delay calculation of cells talking to ports following information is required.

1) Input _transition for input ports (set_input_transition command)

2) Load information for output port (set_load command)

I/O constraints

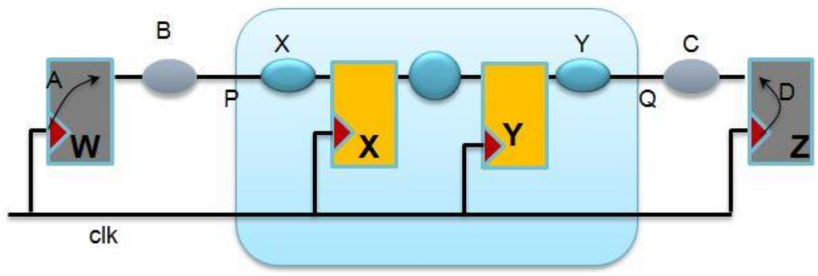


Fig 3.31 I/O Constraints [3]

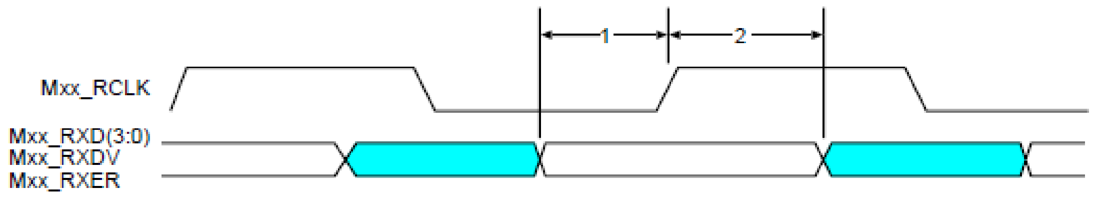
- ❑ Considering clocks as ideal

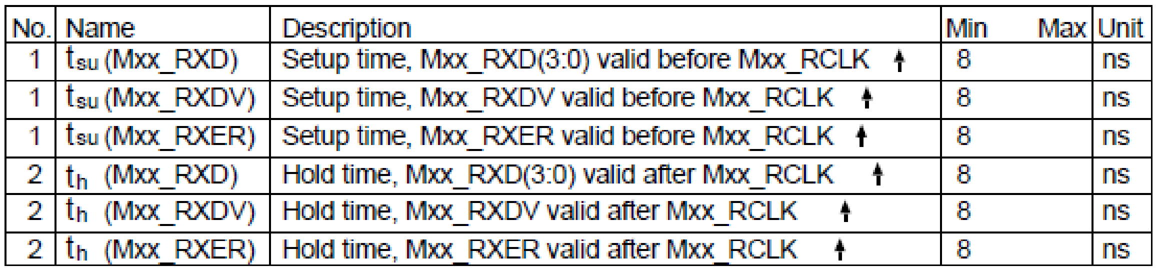
Input Delay (IPD) @ P @ clk = c2q + B

Output Delay (OPD) @ Q @ clk = C + su

Interpreting Timing Specifications

Timing specifications are captured using a waveform diagram and a table of timing parameters.





No.	Name	Description	Min	Max	Unit
1	t_{su} (Mxx_RXD)	Setup time, Mxx_RXD(3:0) valid before Mxx_RCLK ↑	8		ns
1	t_{su} (Mxx_RXDV)	Setup time, Mxx_RXDV valid before Mxx_RCLK ↑	8		ns
1	t_{su} (Mxx_RXER)	Setup time, Mxx_RXER valid before Mxx_RCLK ↑	8		ns
2	t_h (Mxx_RXD)	Hold time, Mxx_RXD(3:0) valid after Mxx_RCLK ↑	8		ns
2	t_h (Mxx_RXDV)	Hold time, Mxx_RXDV valid after Mxx_RCLK ↑	8		ns
2	t_h (Mxx_RXER)	Hold time, Mxx_RXER valid after Mxx_RCLK ↑	8		ns

Fig 3.32. Interpreting Input Timing waveforms

The input setup constraints indicate that the input data will be valid 8ns before the rising clock edge.

The clock MUST arrive at the flop after the data to ensure the data is captured in the flop.

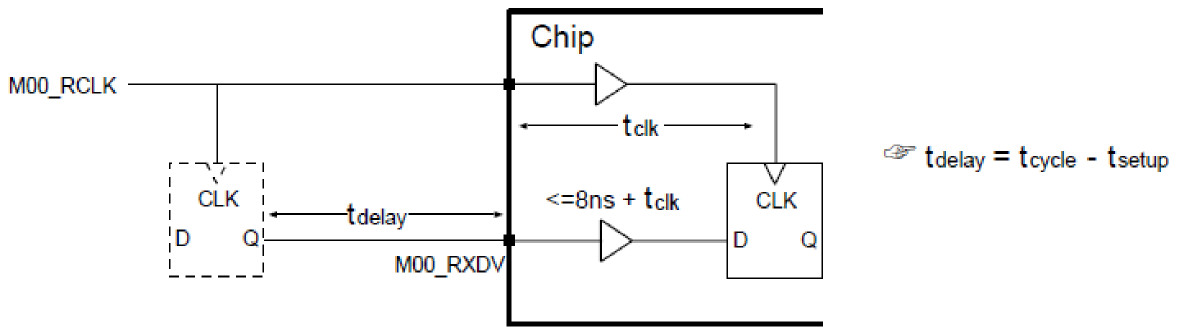


Fig 3.33. Timing Diagram for Input Constraint

Output timing parameters can be specified as delays rather than setup & hold times especially when their reference clock is an input.

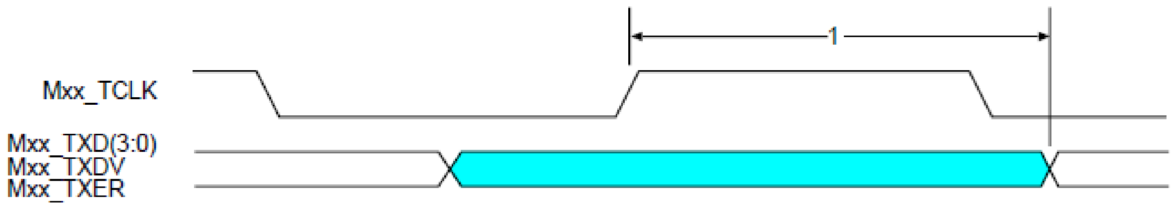


Fig 3.34 Interpreting output timing waveforms[11,12]

The max delay converts to a setup constraint: tsetup = tcycle – tmax.

The min delay converts to a hold constraint: thold = tmin.

The output delay times indicate a maximum of 15ns internal delay and a minimum (hold) of 5ns. When examining output delays the internal clock paths has to be considered.

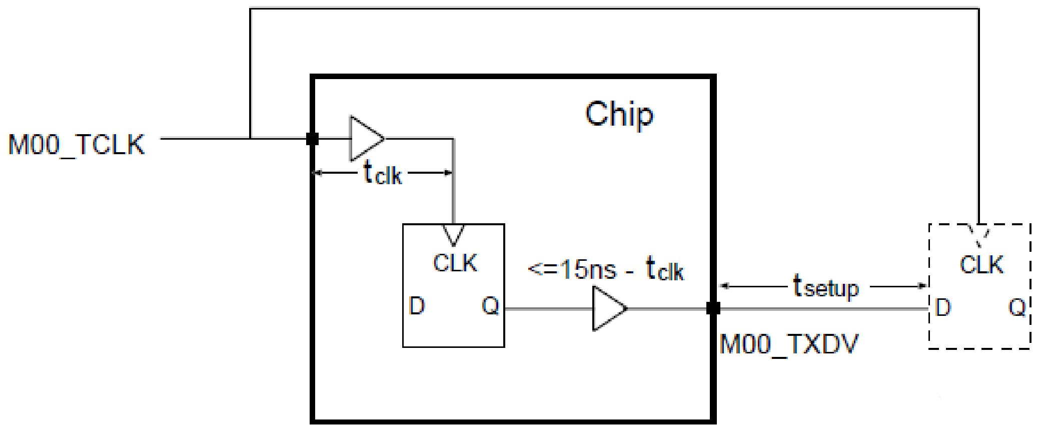


Fig 3.35 Timing diagram for Output Constraint

3.7.4 Timing Exceptions

Timing exceptions are required to model the actual design requirement and to prevent over designing by avoiding unnecessary design optimization efforts. However, caution must be exercised since incorrect exceptions can cause potential silicon failures.

Major Types

- False Path
- Multicycle Path

3.7.4.1 False Paths

A false path is a path that is of no interest. False path is used to educate the tool to mask or ignore certain paths for timing purpose.

Common reasons are:

- A path that cannot functionally occur.
- A path that is not exercised in a particular mode.

Application

- False path is used in any path that is not relevant to a circuit's operation in a specific mode
- It can also be used in a path which would not toggle, e.g. path through a static control register
- Another application of false path is Asynchronous clock domain crossing in merged modes.

•

Static timing analysis can report violations on false paths because there is no knowledge of circuit function.

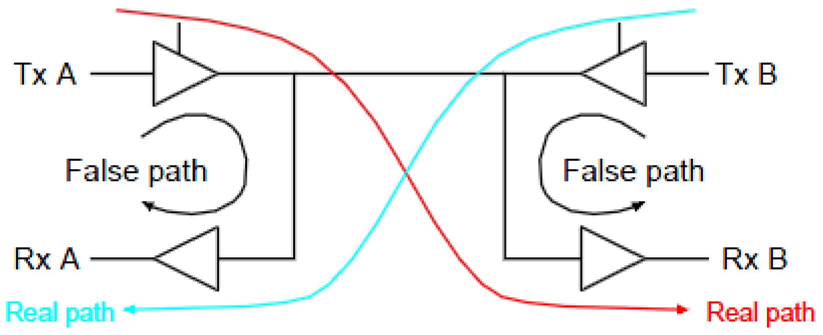


Fig 3.36. False Path [3]

3.7.4.2 Multicycle Paths

A multicycle path is a path takes more than 1 cycle to propagate. It is used to educate the tool about paths that by design take more than one clock cycle to become stable

- It is not a cure for a timing violation
- A genuine multicycle path implies a memory element & needs to be thoroughly justified.
  - A register may need to trigger or capture a signal every second or third rising clock edge because of synchronized enable generation logic.
  - A multicycle path between a multiplier's input registers and output register where the destination latches data on every other clock edge
-

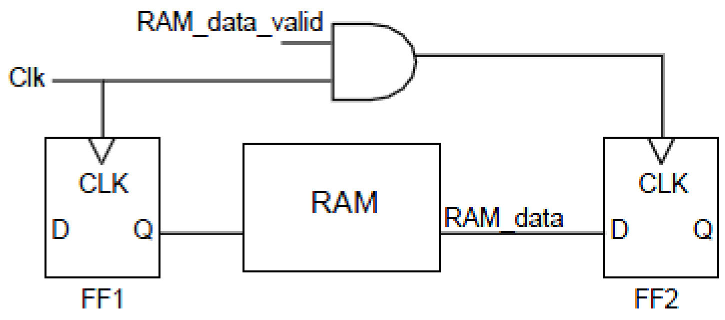


Fig 3.37. Multicycle Condition

The RAM takes 1.5 cycles to propagate and RAM_data_valid goes active 2 cycles after the RAM is accessed. A setup violation would be reported for FF2. FF1/Q to FF2/D is a multicyle path of 2 cycles.

3.7.5 Case Analysis

Need

It is used to set a specific device configuration towards a specific application

Application example

Configure Master or Slave mode of device for SPI interface

3.7.6 Disable Timing

Need

It is used to to disable a specific invalid timing arc in the design

Application

- Disable timing arcs through IPP_PUS/IPP_PUE in functional mode
- It is also used to break combinational timing loops in the design.

3.7.7 Disable Clock Gating Check

Need

It is needed to disable clock gating check on a particular pin or cell

Application

Invalid clock gating checks for observable clock logic or for clock going to data

3.7.8 Set Max/Min Delay

Need

It is needed to apply point to point delay between two points

Application

It is used to constrain feedthrough paths in the design

3.8 Synchronizers

Synchronisers are used to make asynchronous signals synchronous to a clocked design.
Consider this synchroniser:

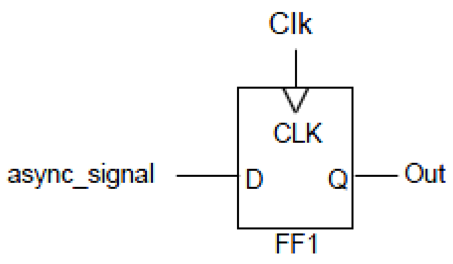


Fig 3.38 Standalone Synchronizer

Since the async_signal can change at any time, setup & hold times will not be met in all cases. Worst case is that the clk rises at the same time as the FF2/D pin changes, possibly leaving FF2 in a meta-stable state.
The meta-stable condition means that a flip-flop has not resolved to be either a 1 or 0. It is some where in between. The diagram shows the possible output transitions of a flop that samples meta-stable input data.

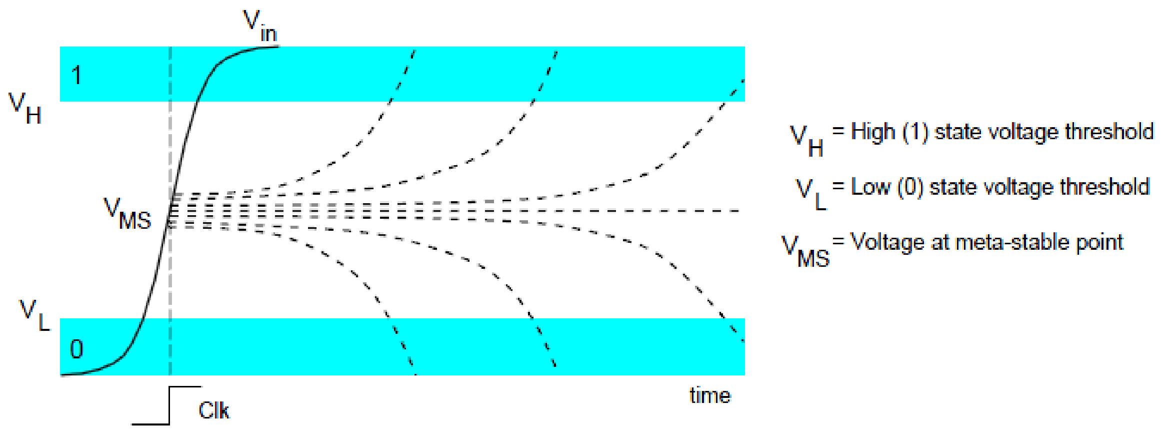


Fig 3.39 Meta stability

It can be seen that the closer to the centre of the meta-stable voltage range the longer it takes for the flop output to resolve. To reduce the chances of a meta-stable state being propagated another flip-flop can be added.

This now takes on the form of a multi-stage synchroniser.

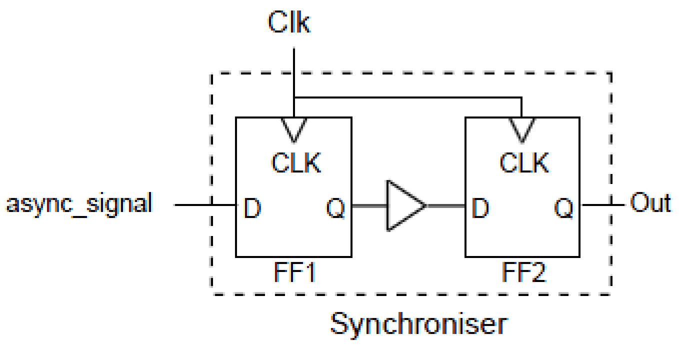


Fig 3.40 Multi-stage synchroniser

If the first flop goes meta-stable, it has a whole cycle to resolve into a 1 or 0. It is conceivable that one cycle may not be enough to resolve the meta-stable state. A third stage can be added to reduce this risk further, at the expense of synchronisation delay. Note, the buffer between the flops is to prevent hold violations.

3.9 Timing Margins

SignOff Margin

It is the margin considered for Final Timing Signoff

Implementation Margin

They are the margins considered during various stages (e.g. synthesis/placement/ cts/route/noise etc) of design flow

3.10 Modes

Any individual functionality of chip independent of other defines a unique mode.

Factors which define a mode:

- Clock source
- Pin muxing
- Case analysis

3.10.1 Need for Modes

Modes are required to model signoff margin scenario at each of stages.

If estimated margins for each stage comes within the defined margin value, then no extra violation should pop up during signoff provided timing is clean in each stage

Typical STA Modes defined for Timing Analysis of a Design

<u>FUNCTIONAL MODES</u>

(1) FUNC MODE

Based on design requirements/application and/or any other reasons FUNC mode can be split into different modes

e.g. Ethernet modes (gmii/mii/smii/rmii/tmii/sssmii etc.)

e.g. Master/Slave modes (dspi/sai etc.)

(2) RESET MODE

This mode is used during booting of the device to configure it properly.

<u>TEST MODES</u>

Various test modes are present for a device depending on whether the testing is to be done using scan or whether built-in self testing is being used.

1. SHIFT
2. BURNIN
3. MBIST
4. NVMBIST
5. ATSPEED SCAN
6. PLL
7. PLL BYPASS

8.STUCKAT SCAN
9.MEMBYPASS
10. MEMENABLE
11. Analog Expose Modes

3.10.2 Important factors for STA mode Selection

1. Design Functionality
2. Design Architecture
3. Multiple Clock Sources and Frequency Variations
4. Clock Edge Relationships
5. Clock Reconvergence
6. Application / Use case of the Design
7. Multiple Clock Sources and Clock Domain Crossing

3.10.3 Mode Merging

- Mode merging is actually the clubbing of multiple separate modes by redefining constraints and creating common constraints covering the requirements of those separate modes

Need for Mode Merging

- More number of STA Modes increases number of signoff scenarios since(sign-off scenarios = number of PVT corners x number of modes)
- More number of modes create difficulty in converging timing closure iterations across all scenarios
- Increased number of resources (machines, LSF, more file management, disc-space) are required for more modes
- Potential coverage issue is found if modes are analysed one at a time– Unintentional masking of a valid path due to inappropriate/incomplete selection criteria used for individual Mode selection

Demerits of Mode Merging

Mode merging increase the complexity and manual analysis efforts, clock definitions, timing exceptions, number of paths, tool runtime and memory footprint requirements. Better machines are required to handle mode merging.

Summary of constraint development

Constraints development, including Mode selection and merging, is an important process and should be judiciously done to ensure complete coverage of the design with proper timing verification[13].Increased number of STA Modes lead to better design coverage and easier timing analysis/verification, but they also increase the efforts involved in timing closure and signoff for all modes.Invalid paths should be taken care through proper timing exceptions to save unnecessary over design efforts. Valid scenarios should not get masked due to improper exceptions. Clock propagation should adhere to the functionality of the design.[14,15]

3.11 On-chip variation

OCV (On- Chip Variation) is taken care of by introducing extra pessimism in the design w.r.t setup and hold checks in the following manner:

OCV DERATE

Setup Timing

Speed Up Capture Path by X%

Slow Down Launch Path by X%

Speed down Data Path by X%

Hold Timing

Slow Down Capture Path by Y%

Speed up Launch Path by Y%

Speed up Data Path by Y%

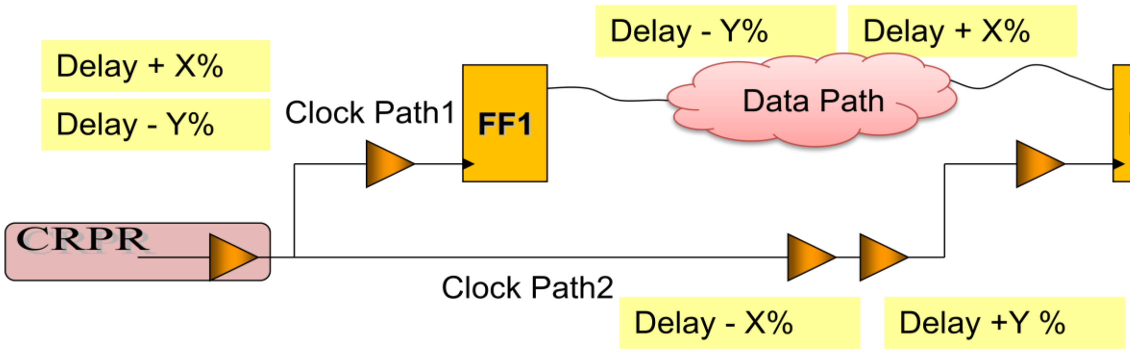


Fig 3.41 OCV Derates

3.12 Timing MODELS

3.12.1 **QTM (Quick Timing Model)** - .lib [16]

1) This is equivalent to constraints that are used.

2) Expectations are set at the interface.

3.12.2 **ILM (Interface Logic Model)** - .v and .spef

1) .v + .spef of interface part only (Most Accurate).

2) it can be used for Signoff.

3.12.3 **ETM (Extracted Timing Model)** - .lib

1) This is an extracted Timing Model, with internal pins defined

2) Here, actual delays are mapped in forms of Arc.

3.13 STA RUN FILE

- Firstly, the design data (A gate level netlist and associated technology .lib) is read.
- Constraining of the design by specifying the clock characteristics, input timing conditions and output timing requirements is carried out.
- Specification environment and analysis conditions such as operating conditions, case analysis settings, net delay modes and timing exceptions is provided.
- Next, checking is done of the design data and analysis setup (scripts) parameters for a full timing analysis.

Lastly a full timing analysis is performed and results are analysed.

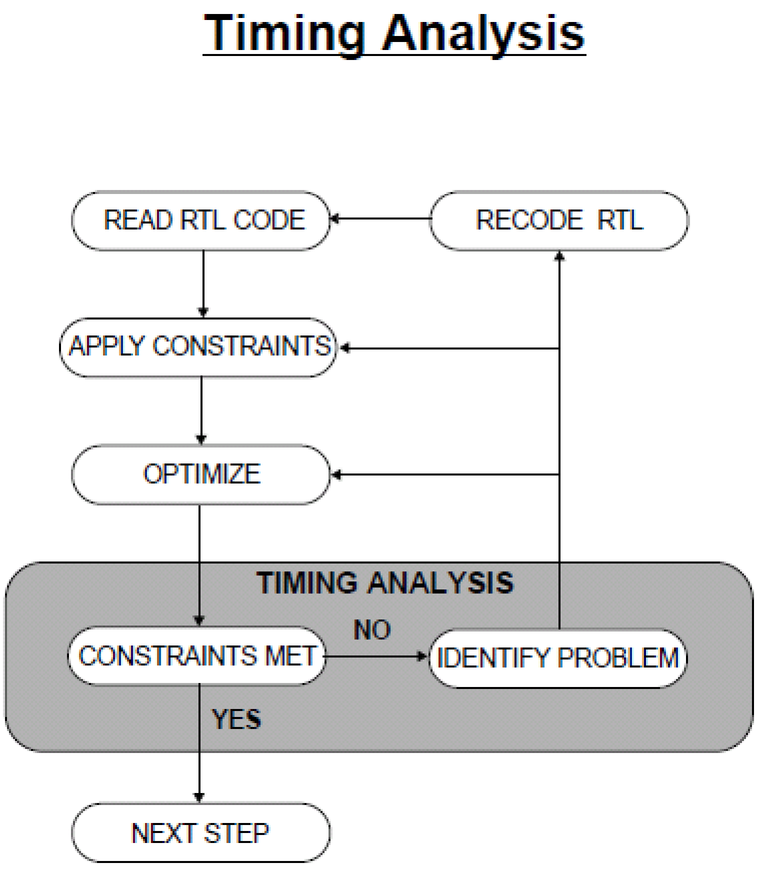


Fig 3.42 Timing analysis [17]

3.13.1 Steps to fix the violations

- The Timing Reports are checked for any violation.
- Annotations are reported.
- It is ensure that the path facing violation is valid.
- A check is done for skew if it is a problem.

- Identification of the critical time consuming element from the verbose report is carried out.
- Checks are done for high transition time and/or high loads.

The following figure shows the timing analysis flow at different stages:

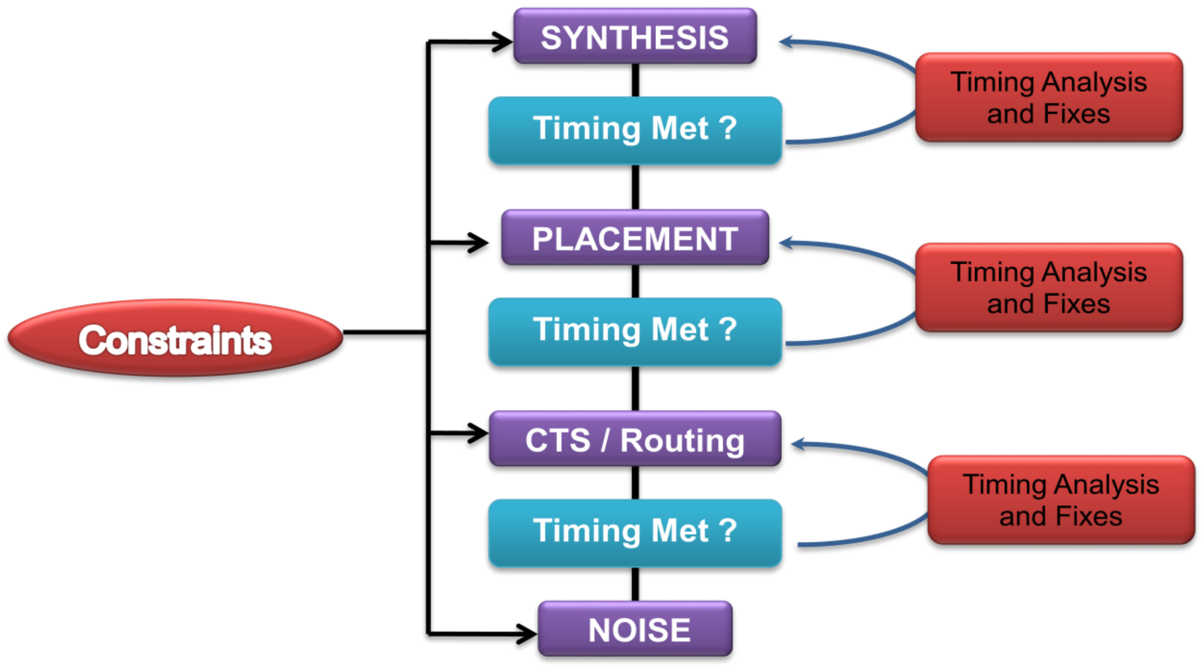


Fig 3.43 Timing Analysis across different stages [1,18]

CHAPTER 4
Project Work

4.1Project set-up

- First, the setup which was based on ETS 10.13 ver which used the PD Flow 3.0, was run. The setup involved populating the database. This used mainly script to fetch files from the database already user released.
- Secondly, all the constraints i.e. sdc files were populated in my area to get the constraints to be read in design. This involved using c shell commands mainly dssc.
- Thirdly, all the scripts were added to my target area which was used for generating different kinds of report or getting all the design reports properly read.
- After this, the SPEF's i.e. Standard Parasitics Exchange Format files were picked which provide the R and C values for the line or wires and help in calculating the delay or induced effects in the designs.
- The netlist is also added to the destination area which provides all the information about the design configuration. They provide all the information about the standard cells as well as the analog blocks which are being put on in the design. In the design there were netlist for the whole design including specific netlist for the sdram and adc.
- After getting all the files needed to run this design it was needed to set the environment variables to get the entire tools right version to be picked up according to the design requirements. This involved setting up of the destination folder for reports dumping, setting the tool version, setting the revision version to get the correct areas to pickup files.

The run gives an encounter shell to work on where various commands for different purposes can be used. For example, the command "calNegSlack" gives the worst negative slack, the total negative slack and the number of path violating.

To see a path command "report_timing" with to and from points to look at the timing of a particular path is used. The report generated is very much detailed and is generated in the form of a table which consists of various information like the fanout of a pin the slew, load, arrival time, incremental delay if noise is also taken into consideration. The report format is customizable using the "set_table_style" command. These report help us debug the path i.e. look for the problem and fix it by different methods like changing cell with better slew or insertion of buffer etc.

All modes consolidated results are prepared in form a table using a mode known as regression mode.

4.2 SDF Generation

“The Standard Delay Format (SDF) was designed to serve as a simple textual medium for communicating timing information and constraints between electronic design automation tools. The original version was designed by Rajit C. Chandra in 1990 while at Cadence Design Systems, and was intended as a means of communicating macrocell and interconnects delays from Gate Ensemble to Verilog-XL, Veritime and other stand-alone tools requiring timing data.”

The SDF was designed from the ground up to be an easy way to convey timing information to a simulator. The SDF file can furthermore be utilized by other design tools. It can be leveraged to convey design constraints identified during timing analysis to layout tools (forward annotation) and it can also be used for post layout timing analysis and simulation (back annotation). SDF is an interface between the STA and the GLS (Gate Level Simulation) Team. It is a copy of STA signoff design.
A SDF contains:-

- Interconnect delays
- Timing Arcs of the hard IPs
- Timing Arcs of the standard cells

Before the SDF generation the regression is run in single mode, then the results are seen and if any violation they are put in annotations for the wcs and bcs modes. Then the run is fired using the –write sdf tag. After the the sdf are generated they are checked in and the results seen for any violation present if no violation is there then the sdf are checked in.

4.3 SoC FA

Two blocks were analysed in SoC FA Block1 and Block2. This work was done as part of the training to understand and build the block-level constraints. The analysis was carried out for shift and stuckat modes. Only reg to reg paths were analysed.

4.3.1 Project Activities

[1] Defining the clocks and verifying from the check timing verbose report whether clock was reaching all desired pins or ports in the design or not. No unclocked flops remained after proper analysis and debugging.
[2] Checking for the check timing report to remove the occurrence of unconstrained endpoints. Only output clock ports remained after proper analysis and debugging.
[3] Building further constraints for the design such as case analysis on scan_mode , scan_enable, startpoint ports of asynchronous pins. (to ensure that they are de-asserted for shift mode).

[4] Ensured by proper analysis that no path to d-pins were present in shift mode except that to a DFT Lock-up Latch.
[5] Carrying out a regression run for func,shift and stuckat modes in worst, best, worst_cold, best_hot corners
[6] Consolidating the STA_Summary of the regression run across various modes and corners also obtaining the unique violation report in the process.

4.3.2 Results FA

4.3.2.1 Check_Timing Summary

```
+-------------------------------------------------------------------------------------+
|                        TIMING CHECK SUMMARY                                         |
|-------------------------------------------------------------------------------------|
|     Warning       |            Warning Description            | Number             |
|                   |                                           |   of               |
|                   |                                           | Warnings           |
|--------------------+---------------------------------------------+--------------    |
| ideal_clock_waveform |   Clock waveform is ideal                |    14 |
| uncons_endpoint      | Unconstrained signal arriving at end point|    7 |
+-------------------------------------------------------------------------------------+
```

4.3.2.2 FA Block1 sta_summary

Block1 STA Summary: STA Summary w/o Noise						
Tag Type: nsi_single_synth						
Tag Date: run_TO_normal_icd_b40070_Mar20_11:50						
CORNER	worst					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	801	-0.785	-469.425	34	-0.452	-2.303
shift	0	6.962	0	0	0.111	0
stuckat	40	-3.898	-123.269	14	-17.515	-62.282
CORNER	worst_cold					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	801	-1.326	-895.019	42	-0.565	-3.067
shift	0	6.962	0	0	0.111	0
stuckat	40	-3.814	-119.325	14	-17.602	-63.068

CORNER	best					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	20	-0.574	-4.118	35	-0.03	-0.737
shift	0	6.962	0	0	0.136	0
stuckat	48	-3.955	-130.404	6	-17.129	-59.054
CORNER	best_hot					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	20	-0.574	-4.122	27	-0.03	-0.433
shift	0	6.962	0	0	0.136	0
stuckat	48	-3.966	-130.226	6	-17.15	-59.078

Table 4.1 FA Block1 sta_summary

4.3.2.3. FA Block1 unique violation summary

Block1_iso_wrap Unique Violations Summary						
Tag Type: nsi_single_synth						
Tag Date: run_TO_normal_icd_b40070_Mar20_11:50						
CHECK_TYPE	SETUP			HOLD		
CORNER	VIOLATIONS	WNS(ns)	TNS(ns)	VIOLATIONS	WNS(ns)	TNS(ns)
worst	841	-3.898	-592.959	48	-17.515	-64.58
worst_cold	841	-3.814	-1014.27	56	-17.602	-66.147
best	68	-3.955	-134.538	41	-17.129	-59.798
best_hot	68	-3.966	-134.339	33	-17.15	-59.52
All Corner's	849	-3.966	-1025.49	64	-17.602	-66.677

Table 4.2 FA Block1 unique violation summary

4.3.2.4 FA Block2 STA SUMMARY

Block2 STA Summary: STA Summary w/o Noise						
Tag Type: nsi_single_synth						
Tag Date: run_TO_normal_icd_b40070_Mar14_14:26						
CORNER	worst					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	19211	-1.031	-4837.38	391	-2.197	-89.782
shift	0	8.635	0	2	-0.06	-0.083
stuckat	294	-15.207	-1606.97	222	-11.016	-453.456
CORNER	worst_cold					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	7831	-0.873	-2309.97	413	-2.295	-97.239
shift	0	8.635	0	2	-0.06	-0.083
stuckat	294	-15.206	-1745.95	236	-11.023	-457.366
CORNER	best					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	282	-0.274	-12.035	120	-1.099	-30.672
shift	0	8.635	0	1	-0.035	-0.035
stuckat	258	-15.122	-1370.83	193	-10.991	-445.06
CORNER	best_hot					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	349	-0.415	-15.6	112	-1.148	-32.318
shift	0	8.635	0	1	-0.035	-0.035
stuckat	264	-15.137	-1375.56	193	-10.991	-445.06

Table 4.3 FA Block2 STA SUMMARY

4.3.2.5 FA Block2 unique violation summary

Block2 Unique Violations Summary						
Tag Type: nsi_single_synth						
Tag Date: run_TO_normal_icd_b40070_Mar14_12:14						
CHECK_TYPE	SETUP			HOLD		
CORNER	VIOLATIONS	WNS(ns)	TNS(ns)	VIOLATIONS	WNS(ns)	TNS(ns)
worst	19446	-15.207	-6416.72	517	-11.016	-537.888
worst_cold	293	-15.206	-1743.94	238	-11.023	-457.414
best	258	-15.122	-1370.81	194	-10.991	-445.095
best_hot	264	-15.137	-1375.57	194	-10.991	-445.095
All Corner's	19447	-15.207	-6557.69	531	-11.023	-541.77

Table 4.4 FA Block2 unique violation summary

4.4 Project PH

It is from 32-bit MCUs family of devices and in 90nm Thin Film Storage (TFS) embedded Flash technology. These devices are primarily focused to serve the metering markets.

4.4.1 Operating Characteristics

- Voltage range 1.71V - 3.6V
- Flash programming voltage from 1.71V to 3.6V
- iRTC battery supply range 1.71V to 3.65V
- Temperature range (TA) -40°C to 85°C
- Flexible modes of operation
- High Performance ARM Cortex M0+ Core
- Upto 50 MHz of Core Clock Frequency

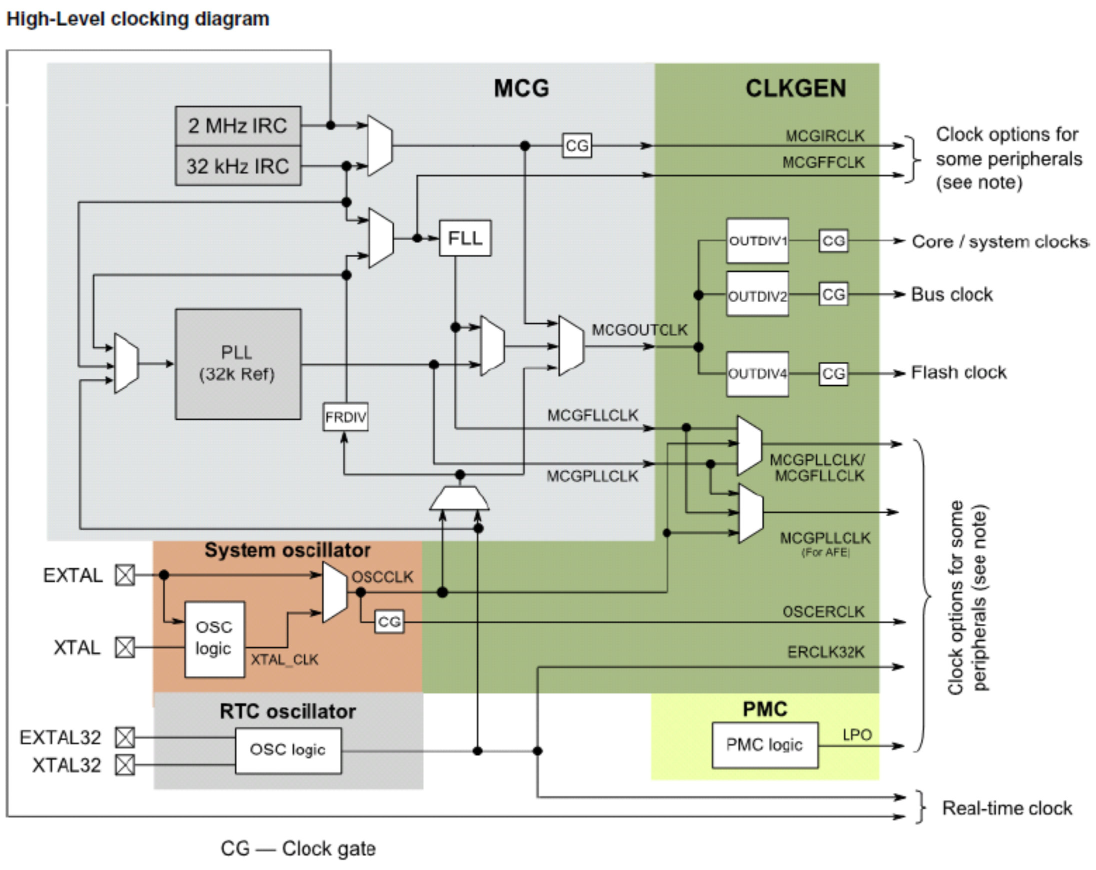


Fig 4.1 High Level clocking diagram of PH

4.4.2 Clocks

- MHz Oscillator
  - Mid Range: 1 MHz to 8 MHz
  - High Range: 8 MHz to 32 MHz
- 32.768 kHz crystal oscillator in iRTC power domain

- Two internal trim-able clock references
  - 32 kHz
  - 4 MHz
- Internal 1 kHz low power oscillator

- PLL to generate clocks for AFE (Analog Front End)
  - Input range: 31.25 kHz to 39.0625 kHz
  - Output range: 11.72 MHz to 14.65 MHz
- FLL to generate core, system & flash clocks
  - Input range: 31.25 kHz to 39.0625 kHz
  - Output Range: 20 MHz to 50 MHz
- Clock ratio1
  - Core:Bus:Flash = 2:1:1 for core clock > 25 MHz
  - Core:Bus:Flash = 1:1:1 for core clock <= 25 MHz

4.4.3 Project Activities and Results

The project activities for PH included

4.4.3.1 Constraint development for shift and stuckat modes for PH

The constraints were taken from the xls provided by the DFT team and after proper analysis , these were placed to configure the Design in the scan mode and stuckat mode.

The case analysis were placed including the global scan mode.

There after clocks were defined in the shift and stuckat mode until all the required flops were clocked. Temporary clock definitions needed to be placed to be able to start working with a still – to -more mature design.

Thereafter path exceptions and disable timing (to break combinational loops) were placed to emulate the design scenario.

Ph.shift.clocks.tcl

```
##-----------------------------------------
## Generated Clock:IRTC_SHIFT_CLK will be removed after bypassing through a mux
TEMPORARY DEFINITION flop present in clock path should have been bypassed in test
mode, but left somehow. Therefore temporary clocks defined in shift mode so that flops in
irtc don't remain unclocked##-----------------------------------------cause: unclocked flops in
irtc in check_timing report , bypassing to be done by DFT team
proc_create_generated_clock -name IRTC_SHIFT_CLK -pin
clkgen/clkgen_cp/count_irtc_div_reg_2_/q  -master_clock SHIFT_CLK -source
clkgen/clkgen_cp/count_irtc_div_reg_2_/ck -divide_by 1


#creating OSC_SHIFT_CLK since multiple clock sources yet to be defined for the common
clock SHIFT_CLK by dft team TEMPORARY DEFINITION leading to unclocked flops in
irtc
Cause : unclocked flops in irtc in check_timing report , clock to be defined by DFT team
```

proc_create_clock -name OSC_SHIFT_CLK -period $SHIFT_CLK_PERIOD -rise $SHIFT_CLK_RISE_EDGE -fall $SHIFT_CLK_FALL_EDGE -pin irtc/a_ip_osc_3v32kvlp_nn_c90lp/clk_32kvlp_3v

Ph_top.shift.case.tcl

#below analysis applied to select d3 pin of the corresponding mux TEMPORARY EXCEPTION PLACED until the issue is resolved, clock was coming to the d3 pin of the mux , if d1 pin was allowed to be selected 119 flops in adc were remaining unclocked in shift mode. Cause: unclocked flops in adc due to selection of d1 pin rather than d3 pin on mux in clock path
set_case_analysis 1 [get_pins adc0/adc_adcksel/rc_mux_CPD_adc0_div_clk_erclk_clk_lv_adc0_asclk_bus_clk_pll_0__H__clk_mux_STA/sl1]

#below bits as per discussion with sandeep, on REL14, are made zero , as paths to rb pins of lcd and paths to anl pins thru xbar were observed. Cause : observations of paths to rb / sb pins (async pins) in shift mode
##analog expose mode
set_case_analysis 0 [get_pins tcu/scan_control_reg/update_register_reg_4_/$output]
#Mbist_reset
set_case_analysis 0 [get_pins tcu/scan_control_reg/update_register_reg_6_/$output]

##Below is temporary exception applied , as this mux that selects between clock path from EDT or from RGPIO, was unconstrained therefore launch and capture clocks may go through different paths leading to increase in skew. Constraining it to 1 as it will now pick only the path thru EDT.TEMORARY EXCEPTION : REL14 cause : unconstrained mux in clock path allowing selection of different paths for launch and capture clocks
set_case_analysis 1 [get_pins clkgen/clkgen_cp/p214748365A/sl0]

##Temporarily constraing in REL14 the below bit 20 "ipt_se_sync" to zero, to control rb violations on AFE wrapper.This would be constrolled from next release onwards. cause : observation of paths to rb pins in afe wrapper
set_case_analysis 0 [get_pins tcu/scan_control_reg/update_register_reg_20_/$output]

Ph_top.shift.exceptions.tcl

#temporary exceptions added by Babul post dbmay04
##
#these paths should be half-cycle since starting from lockup_reg, will be corrected in next netlist cause: since common clock is present in shift mode no concept of clock exceptions for async domains , therefore to ease hold time-check an extra half cycle path traversal time is given by placing lock-up latches along with flops in pos_edge-neg_level-pas_edge or neg_edge-pos_level-neg_edge manner

set_false_path -from [get_pins padring/pti_top/edt_comp_photon/photon_edt/photon_edt_decompressor_i/lfsm_vec_lockup_reg_13_/ck] -to [get_pins photon_dgo/lptimer/lptimer_counter/timer_trigger_neg_reg/sdi]
set_false_path -from [get_pins padring/pti_top/edt_comp_photon/photon_edt/photon_edt_decompressor_i/lfsm_vec_lockup_reg_17_/ck] -to [get_pins photon_dgo/lptimer/lptimer_counter/timer_trigger_neg_reg/sdi]
set_false_path -from [get_pins padring/pti_top/edt_comp_photon/photon_edt/photon_edt_decompressor_i/lfsm_vec_lockup_reg_22_/ck] -to [get_pins photon_dgo/lptimer/lptimer_counter/timer_trigger_neg_reg/sdi]

#placed after discussion with Sandeep, paths from tcu/scan_control_reg and tcu/scan_control_reg1 are invalid in shift and stuckat modes cause: paths observed with high hold violation from scan_control_registers
set_false_path -from [get_pins tcu/scan_control_reg1/update_register_reg_3_/ck] -to [get_pins photon_dgo/lptimer/lptimer_counter/timer_trigger_neg_reg/sdi]
set_false_path -from [get_pins tcu/scan_control_reg/update_register_reg_2_/ck] -to [get_pins photon_dgo/lptimer/lptimer_counter/timer_trigger_neg_reg/sdi]

#path to d pins of lock_up latch is a valid path in shift mode, but should have been half-cycle paths cause : path to d pins in shift mode are usually invalid paths EXCEPT when they are at d pins of lock_up registers. BUT these paths should have been half-cycle paths.
set_false_path -from [get_pins rnga/rnga_logic/rnga_core/rnga_clk_gen/rnga_oscillators/osc_clk_1_reg/ck] -to [get_pins rnga/rnga_logic/rnga_core/rnga_clk_gen/rnga_oscillators/DFT_lockup_latch_en_lo_top_bus_25m_chain_47__H__g11/d]
set_false_path -from [get_pins rnga/rnga_logic/rnga_core/rnga_clk_gen/rnga_oscillators/osc_clk_2_reg/ck] -to [get_pins rnga/rnga_logic/rnga_core/rnga_clk_gen/rnga_oscillators/DFT_lockup_latch_en_lo_top_bus_25m_chain_47__H__g13/d]

Ph_top.stuckat.exceptions.tcl
NOTE : Stuckat is done to check manufacturing defects . So ideally no exceptions should be there.
##### pte[7] acting both as data port and clock port for the following in2reg paths ################ a port can’t act as data port and clock port simultaneously for an in2reg path, therefore these paths are set false
set_false_path -from [get_ports pte[7]] -to [get_pins rgpioc_pctl_pte/PCTL_REG_7_rgpioc_pctl_filter/ipp_ind_pad_sync_reg/d]
set_false_path -from [get_ports pte[7]] -to [get_pins rgpioc_pctl_pte/ipp_ind_sync_reg_7_/d]
set_false_path -from [get_ports pte[7]] -to [get_pins sci2/sci_pi/pi_ind_int_reg_1_/d]

###### False path added since through_pin (clock generation point) bus_divider, itself was acting as launching point ###### cause: bus divider mux is a clock generation point , So this is again a case of both data and clock for a path coming from the same point , to be confirmed from FE/IP team

```
set_false_path -from [get_pins
clkgen/clkgen_dividers_photon_c90lp/bus_divider/divider_with_sync/mux_div_clk/mux_sig
_out_80_9_sta/x] -to [get_pins da_sglcd_top/lcd_digit/fault_det/samp_clk_d1_reg/d]
```

4.4.3.2 Check-timing analysis for PH

The check timing analysis was done mostly to ensure that
[1] All the required flops were clocked . Some flops may be out of scan in which case they don’t get clocks in a particular mode. Also clock-gating cells may have been bypassed in scan-mode in which the ckb pin of clock-gating cell remain unclocked. Master clock not reaching the generated clock source can again lead to unclocked flops in the design which needs to be fixed.

[2] No timing loops remain in the design. Set_disable_timing was used to break the loops by removing the arcs.

<u>SCRIPT to find out the fanin startpoints of unclocked clockpins and the clock/constant info</u>

```
rm -rf /mot/b40070/babulfiles/outfiles/check_timing.rpt
set_alias
nos
check_timing -verbose > /mot/b40070/babulfiles/outfiles/check_timing.rpt

vim /mot/b40070/babulfiles/outfiles/check_timing.rpt -c
":v/Clock.*not.*found.*where.*clock.*is.*expected/d" -c ":1d" -c ":wq"
set file1 [open /mot/b40070/babulfiles/outfiles/check_timing.rpt r+]
#set file2 [open /mot/b40070/babulfiles/outfiles/babul_check_timing.rpt w+]
set outfile2 [open /mot/b40070/babulfiles/outfiles/babul_check_timing_out3.csv w+]
#set outfile [open /mot/b40070/babulfiles/outfiles/test.csv w+]
puts $outfile2 "PREPARED BY BABUL ANUNAY B40070 "
puts $outfile2 "ORDER : POINT WHERE CLOCK IS NOT FOUND, IF ANY CONSTANT
ON THE POINT,\n STARTPOINT PINS AND PORTS, WHICH STARTPOINTS HAVE
ANY CLOCK
OR CONSTANT ON IT"
set file_data_vio [read $file1]
set data_vio [split $file_data_vio "\n"]
  foreach line $data_vio {
     if { $line != "" } {
      set value [lindex $line 0]
 #    puts $testfile "$value"
      set pinconst [get_property [get_pins -quiet $value] constant]

set e [get_ports -quiet [all_fanin -to $value -startpoints_only]]
            if {[sizeof_collection $e] != 0 } {

              set mn [get_object_name $e]
               for {set i 0} {$i < [llength $mn]} {incr i} {
                 puts $outfile2 "$value,   $pinconst   ,FANIN PORT NO.$i [lindex $mn $i]"
               }
```

```
          foreach_in_collection ports $e {
           set portclocks [get_property [get_ports -quiet [get_object_name $ports]] clocks]
           set portcons [get_property [get_ports -quiet [get_object_name $ports]] constant]
                if {[sizeof_collection $portclocks] != 0} {
            puts $outfile2 "\t\tSTARTPOINT_INFO,[get_object_name
$ports],[get_object_name $portclocks]"
             }
              if {[regexp {[^NA]} $portcons]} {
              puts $outfile2 "\t\tSTARTPOINT_INFO,[get_object_name $ports], CONSTANT=
$portcons"
                          }
               }

          }

         set f [get_pins -quiet [all_fanin -to $value -startpoints_only]]
          if {[sizeof_collection $f] != 0 } {


          set mnm [get_object_name $f]
            for {set i 0} {$i < [llength $mnm]} {incr i} {
              puts $outfile2 "$value,   $pinconst   ,FANIN PIN NO.$i [lindex $mnm $i]"
             }





          foreach_in_collection pins $f {
           set pinclocks [get_property [get_pins -quiet [get_object_name $pins]] clocks]
           set pincons [get_property [get_pins -quiet [get_object_name $pins]] constant]
           if {[sizeof_collection $pinclocks] != 0} {
           puts $outfile2 "\t\tSTARTPOINT_INFO,[get_object_name $pins],[get_object_name
$pinclocks]"
           }
           if {[regexp {[^NA]} $pincons]} {
           puts $outfile2 "\t\tSTARTPOINT_INFO,[get_object_name $pins], CONSTANT=
$pincons"
              }
           }

          }

puts $outfile2 "\n"
     #puts $file2 "$value, $pincons"
    }
  }
```

```
puts "output file is /mot/b40070/babulfiles/outfiles/babul_check_timing_out3.csv"
#puts "output file pins is /mot/b40070/babulfiles/outfiles/test.csv"
close $outfile2
close $file1
#close $outfile
#close $file3
```

4.4.3.3 Lib Parser for PH

The analog libraries in the design may have an timing arc missing or may have the library max-transition or max-delay limit more than the index value in which case the tool extrapolates and timing may be inaccurate. The report_lib_info command reports the arc missing information. This has to be done for all the analog libraries in the design.
A script was developed which reported the i/p transition limits on the input pins and load limits on the output pins of all analog libraries in the design.

<u>SCRIPT to report the input transition on input pins and output load for output pins of an Analog lib</u>

```
#proc aip_info {} {

set outfile2 [open /mot/b40070/babulfiles/photonoutfiles/babul_lib_parsel_out3.csv w+]
set aip_in_pins [get_pins -quiet -of_objects [get_cells -hier * -filter "is_hierarchical == false
&& (ref_lib_cell_name =~ *a_ip* ||
ref_lib_cell_name =~ *A_IP* || ref_lib_cell_name =~ *adc*)"] -filter "direction == in"]
set aip_out_pins [get_pins -quiet -of_objects [get_cells -hier * -filter "is_hierarchical == false
&& (ref_lib_cell_name =~ *a_ip* ||
ref_lib_cell_name =~ *A_IP* || ref_lib_cell_name =~ *adc*)"] -filter "direction == out"]
puts $outfile2 "INPUT_PIN ,lib_cell_name, slew max fall, slew max rise, slack_max,
slack_min"
append_to_collection aip_in_pins  [get_pins -quiet -of_objects [get_cells -hier * -filter
"is_black_box == true"] -filter "direction == in"] -unique
append_to_collection aip_out_pins  [get_pins -quiet -of_objects [get_cells -hier * -filter
"is_black_box == true"] -filter "direction == out"] -unique
foreach_in_collection in_pin $aip_in_pins {
set m [get_cells -of_objects [get_pins $in_pin]]
set n [get_property [get_cells -quiet $m] ref_lib_cell_name]
set c [get_property $in_pin slew_max_fall]
set d [get_property $in_pin slew_max_rise]
set smin [get_property $in_pin slack_min]
set smax [get_property $in_pin slack_max]
puts $outfile2 "[get_object_name $in_pin],$n,$c,$d, $smax,$smin "
}
puts $outfile2 "\n"
puts $outfile2 "OUTPUT_PIN ,lib_cell_name,connected
net,pin_capacitance_max_fall,pin_capacitance_max_rise, slack_max, slack_min "

foreach_in_collection out_pin $aip_out_pins {
set g [get_property $out_pin net_name]
set p [get_cells -of_objects [get_pins $out_pin]]
set q [get_property [get_cells -quiet $p] ref_lib_cell_name]
set asmin [get_property $out_pin slack_min]
set asmax [get_property $out_pin slack_max]

set h [get_property [get_nets -quiet $g] pin_capacitance_max_fall]
set i [get_property [get_nets -quiet $g] pin_capacitance_max_rise]
puts $outfile2 "[get_object_name $out_pin],$q,$g,$h,$i,$asmax,$asmin "

}
puts $outfile2 "\n"
puts "find outputfile at /mot/b40070/babulfiles/photonoutfiles/babul_lib_parsel_out3.csv"
close $outfile2
#}
```

4.4.3.4 DRV Analysis for PH

The DRV analysis consists of analysing and fixing the transition violation (when the transition on a pin is greater than the maximum transition limit defined in the library) and the capacitance violation (when the load on a pin is greater than the maximum capacitance limit defined in the library).

Transition violation are fixed usually by upsizing the driver cells and load violations can be fixed by adding buffers before the load-pins of the output net.

<u>Script to add buffers before the load pins to fix Transition violations</u>

```
set outfile1 [open /mot/b40070/babulfiles/photonoutfiles/drv/mydrv_outcap.csv w+]
set outfile2 [open /mot/b40070/babulfiles/photonoutfiles/cap_buf_list.csv w+]

#cp /mot/b40070/babulfiles/photonoutfiles/drv/june5_2/cap_photon_drv.rpt
cap_photon_drv_copy.rpt
global pvt_corner
cp /mot/b40070/babulfiles/photonoutfiles/drv/june7route/cap_${pvt_corner}_earliest_run.rpt
cap_photon_drv_copy.rpt

vim cap_photon_drv_copy.rpt -c ":1,8d" -c ":wq"

set file [open cap_photon_drv_copy.rpt r+]
set file_data [read $file]
set data [split $file_data "\n"]
foreach drvpoint_col $data {
   if { $drvpoint_col != "" } {



      set drvpoint [lindex $drvpoint_col 0]

#puts $outfile1 "[drvcap $drvpoint 16]"
set net [all_connected [get_pins $drvpoint]]
set b [get_property [get_nets $net] load_pins]
set mn [get_cells -of_objects [get_pins $b]]
set gh [gpcells $mn ref_lib_cell_name]
if {[regexp dgo $gh] == 0} {
```

```
add_repeater -term [get_object_name $b] -cell buf_hvt_16
reportCapViolation -outfile final_cap.rpt
puts $outfile2 "add_repeater -term [get_object_name $b] -cell buf_hvt_16"

}

if {[regexp dgo $gh]} {

add_repeater -term [get_object_name $b] -cell buf_dgo65_8
reportCapViolation -outfile final_cap.rpt
puts $outfile2 "add_repeater -term [get_object_name $b] -cell buf_dgo65_8"

}



puts "\n find the detailed  outputfile at
/mot/b40070/babulfiles/photonoutfiles/drv/mydrv_outcap.csv"
puts "find the command summary at /mot/b40070/babulfiles/photonoutfiles/cap_buf_list.csv
"
puts "outfile is final_cap.rpt"



}
}


puts "\n find the detailed  outputfile at
/mot/b40070/babulfiles/photonoutfiles/drv/mydrv_outcap.csv \n"
puts "\n find the command summary at
/mot/b40070/babulfiles/photonoutfiles/cap_buf_list.csv\n "
close $outfile1
close $file
close $outfile2
```

4.4.3.5 Clock-Gating Analysis for PH

Clock gating checks present on primitive cells are diagnosed for their validity by checking whether the clock-gating are catering to any clock-pins or not or whether the enable generation logic is glitch-free or not .

<u>SCRIPT to check whether any clock-pins are present in the fanout of the clock-gating cell's output pin</u>

```
#rm -rf /mot/b40070/babulfiles/photonoutfiles/myckcheck.csv
set outfile1 [open /mot/b40070/babulfiles/photonoutfiles/myckcheck2.csv w+]
set abc [report_timing -check_type clock_gating_hold -max_points 200000 -collection]

foreach_in_collection ab $abc {
   if {[sizeof_collection $ab] != 0 } {
    set cap [get_object_name [get_property $ab capturing_point]]
    set slack1 [get_property $ab slack]
           if {$slack1 < 0} {
           set b [get_pins -quiet [all_fanout -from [get_pins -quiet $cap]]]
            foreach_in_collection c $b {
            #set d [get_property $c is_clock]]

                  if {[sizeof_collection $c] != 0 } {

                  set mn [all_registers -clock_pins]
                  foreach_in_collection e $mn {
                  if {[string compare [get_object_name $c] [get_object_name $e]] == 0} {



                  set net [get_object_name [all_connected $c]]
                      if {[sizeof_collection $net] != 0 } {

                       set d [dbIsNetClock $net]
                          if {$d == 1} {
                            puts $outfile1 "$cap , [get_object_name $c] ,$d , $slack1 "
                          }
                      }
                 }
               }
              }
           }
       }
   }
}
close $outfile1
puts "find the outputfile at /mot/b40070/babulfiles/photonoutfiles/myckcheck2.csv"
g /mot/b40070/babulfiles/photonoutfiles/myckcheck.csv
```

4.4.3.6 Analysis of PH timing on post-CTS db

Finally , the timing summary was taken out for various modes and various corners in which the design is required to be configured. This can be done at different stages of the design . The violations are then analysed across different corners and different fixes are generated and optimisation is done by all involved teams to ensure that timing is met across different modes and corners at the signoff.

Table 4.5 and 4.6 PH STA SUMMARY MAY17 and JUNE 19

ph_top STA Summary: STA Summary w/o Noise Tag Type: nsi_ocv_derate_cts Tag Date: run_cts_db_may11_ics_hier_normal_icd_b40070_May17_16:56						
CORNER	**cmax_hot_worst**					
CHECK_TYPE	**SETUP**			**HOLD**		
MODE	**VIOLATIONS**	**WNS (ns)**	**TNS (ns)**	**VIOLATIONS**	**WNS (ns)**	**TNS (ns)**
func	365	-0.631	-35.909	28	-1.034	-6.323
shift	0	4.49	0	85	-0.284	-6.378
stuckat	0	0	0	7	-0.012	-0.032
atspeed	70	-0.631	-6.957	177	-0.7	-41.064
mbist	1	-0.085	-0.085	48	-0.387	-11.155
nvmbist	0	0.349	0	6	-0.314	-0.987
CORNER	**cmin_cold_best**					
CHECK_TYPE	**SETUP**			**HOLD**		
MODE	**VIOLATIONS**	**WNS (ns)**	**TNS (ns)**	**VIOLATIONS**	**WNS (ns)**	**TNS (ns)**
func	0	0	0	378	-1.269	-16.021
shift	0	18.627	0	475	-0.291	-15.405
stuckat	0	0	0	36	-0.399	-2.319
atspeed	0	0	0	470	-0.399	-25.282
mbist	0	7.932	0	245	-0.163	-8.274
nvmbist	0	7.495	0	182	-0.089	-3.707

ph_top STA Summary: STA Summary w/o Noise						
Tag Type: nsi_ocv_derate_route						
Tag Date: run_db_june18_fixes_ics_hier_normal_icd_b21591_Jun19_08:42						
CORNER	**cmax_hot_worst**					
CHECK_TYPE	**SETUP**			**HOLD**		
MODE	**VIOLATIONS**	**WNS (ns)**	**TNS (ns)**	**VIOLATIONS**	**WNS (ns)**	**TNS (ns)**
func	4	-0.979	-2.035	2	-0.069	-0.078
atspeed	7	-1.572	-3.144	7	-0.466	-1.954
shift	0	0.375	0	0	0.027	0
stuckat	4	-0.063	-0.131	2	-0.113	-0.145
mbist	0	0	0	0	0.081	0
nvmbist	1	-0.157	-0.157	0	0.079	0

CORNER	cmin_cold_best					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	18	-0.336	-4.81	3	-0.059	-0.103
atspeed	18	-0.363	-5.12	8	-0.166	-0.693
shift	0	1.741	0	0	0.003	0
stuckat	18	-0.332	-5.013	2	-0.327	-0.441
mbist	0	5.944	0	0	0.008	0
nvmbist	19	-0.43	-5.24	0	0.008	0
CORNER	cmin_hot_best_hot					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	18	-0.298	-3.778	4	-0.444	-0.898
atspeed	18	-0.329	-4.14	7	-0.187	-0.703
shift	0	1.853	0	0	0.009	0
stuckat	18	-0.294	-4.042	2	-0.098	-0.108
mbist	0	5.372	0	0	0.016	0
nvmbist	19	-0.345	-4.122	0	0.016	0
CORNER	cmax_cold_worst_cold					
CHECK_TYPE	SETUP			HOLD		
MODE	VIOLATIONS	WNS (ns)	TNS (ns)	VIOLATIONS	WNS (ns)	TNS (ns)
func	5	-0.362	-0.952	12	-0.178	-0.619
atspeed	8	-1.01	-2.021	17	-0.491	-2.539
shift	0	2.155	0	0	0.021	0
stuckat	5	-0.118	-0.303	12	-0.367	-0.857
mbist	0	0.037	0	0	0.047	0
nvmbist	3	-0.205	-0.265	0	0.047	0

CHAPTER 5
CONCLUSIONS AND FUTURE SCOPE OF WORK

5.1 REVIEW

During my project in Freescale Semiconductors India Pvt. Ltd., I was assigned the task related to timing. Including that, I learned and understood the SoC design flow and the need, contribution of TIMING ANALYSIS in the design flow. I learned and understood the flow used for timing checks and SDF generation.

I developed interactive scripts, and worked in the area of fixing violation as well as rigorous debugging. I understood many methods of debugging and taking lead in a problem. Working under intensive situation helped me understand the need of different procedures involved for each and every task.

I used the information during my project to debug the solution in the following project like PH. This involved understanding the need for each and every file in the flow and take on understanding the need and function of it.

.In addition, the project helped me understand and strengthen within me, the roles of

- **Teamwork** is the concept of people working together cooperatively, as in a sports team. Projects require that people work together, so teamwork has become an important concept in organizations. Effective teams are an intermediary goal towards getting good, sustainable results. Industry has seen increasing efforts through project and cross-project to help people to work together more effectively and to accomplish shared goals.
- **Co-operation,** is the practice of individuals or larger societal entities working in common with mutually agreed-upon goals and possibly methods, instead of working separately in competition, and in which the success of one is dependent and contingent upon the success of another.
- **Co-ordination,** making different people or things work together for a goal or effect.
- **Effective Communication** keeps people up-to-date about the happenings around. This is very essential for a company where many teams work simultaneously.

5.2 CONCLUSION

However, there still is much more to timing than meets the eye. As mentioned before, timing takes away a lot of the total resources in a typical chip design cycle. Chip design flow requires timing at each level of abstraction beginning from architecture to silicon prototyping. The challenge for engineers here is to verify spec-adherence for multiple abstractions. The timing process depends on various parameters such as the methodology employed, and the complexity of SoCs in terms of size, functionality and application.

Statistical Static Timing Analysis (SSTA) ,is a procedure that is becoming increasingly necessary to handle the complexities of process and environmental variations in integrated circuits.

There are a plethora of new tools, methodologies and languages in the semiconductor industry today. Each user segment has it's own preference. The combination of languages, tools, IPs and methodologies has morphed traditional timing into a "Hybrid Model" process. Since time-to-verify has become a major concern, moving forward, verification engineers may now have to depend on the "Hybrid Timing Model," that addresses all parameters of the timing process.

The future of design timing does not belong to any particular tool, language or methodology, but a combination of them all.

REFERENCES

[1] Chip Design Flow –Amit Jindal FREESCALE TRAINING PROGRAM

[2] FREESCALE MSG-UNIVERSITY PROGRAM - STA Basics – Ateet Mishra and Amol Agarwal

[3] FREESCALE MSG-UNIVERSITY PROGRAM STA-Constraint Development and STA Mode Selection – Neha Mathur and Ashish Maitra

[4] FREESCALE MSG-UNIVERSITY PROGRAM STA- Understanding Clock and Reset from Timing Perspective – Pawan Deep Gandhi and Siddharth Taneja

[5] FREESCALE MSG-UNIVERSITY PROGRAM STA – Glitch Analysis –Saurabh Dhingra , Naveen S , Dipti Gupta

[6] R. B. Hitchcock, "Timing verification and the timing analysis problem", *ACM Design Automation Conference*, pp.594-604, 1982.

[7] Robert B.Hitchcock, Sr, Gordon L. Smith, David D. Cheng, "Timing Analysis of Computer Hardware," IBM Journal, vol. 26, no. 1, Jan 1981.

[8] Wortmann, A.; Simon, S.; Bergholz, W.; Muller, M.; Mader, D.; Static timing analysis with rigorous exploitation of setup time margins,Circuits and Systems, 2003. MWSCAS '03. Proceedings of the 46th IEEE International Midwest Symposium Volume 3, 27-30 Dec. 2003 Page(s):1396 - 1399 Vol. 3

[9] X. Bai, C. Visweswariah, P. Strenski, D. Hathaway,"Uncertainty-aware circuit optimization", ACM Design Automation Conference, 2002.

[10]Anirudh Devgan ,IBM Research and Chandramouli Kashyap IBM Microelectronics Block-based Static Timing Analysis with Uncertainty, http://www.eecs.berkeley.edu/~alanmi/research/timing/papers/stat_iccad03.pdf

[11] PrimeTime Static Timing Analysis

bass.gmu.edu/courses/ECE545/projects_F06/PrimeTime.pdf

[12] A.Gattiker, S. Nassif, R. Dinaker, C. Long, "Timing yield estimation from static timing analysis," Proceedings ISQED, 2001.

[13] McWilliams, T.M. (1980). "Verification of timing constraints on large digital systems". Design Automation, 1980. 17th Conference on. IEEE. pp. 139--147.

[14] R. Chadha and J. Bhasker, Static Timing Analysis for Nanometer Designs, by, ISBN 978-0-387-93819-6, Springer, 2009

[15] N. Jouppi, "Timing Analysis for nMOS VLSI", *ACM Design Automation Conference*, 1983, pp. 411-418.

[16] Electronic Design Automation for Integrated Circuits Handbook, by Lavagno, Martin, and Scheffer, ISBN 0-8493-3096-3 A survey of the field. Volume II, Chapter 8, 'Static Timing Analysis' by Sachin Sapatnekar

[17] http://en.wikipedia.org/wiki/Static_timing_analysis

[18] http://eetimes.com/