A project report on

Duty Cycle Distortion (DCD)

Submitted in partial fulfillment for the award of the degree of

Master of Technology

In

VLSI Design

by

SHREYAS S BAGI (21MVD0086)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING (SENSE)
VELLORE INSTITUTE OF TECHNOLOGY
VELLORE - 632 014.

May, 2023

Duty Cycle Distortion (DCD)

Submitted in partial fulfillment for the award of the degree of

Master of Technology

In

VLSI Design

by

SHREYAS S BAGI (21MVD0086)

SCHOOL OF ELECTRONICS ENGINEERING (SENSE)
VELLORE INSTITUTE OF TECHNOLOGY
VELLORE - 632 014.

May, 2023

DECLARATION

I hereby declare that the thesis entitled “Duty Cycle Distortion” submitted by me, for the award of the degree of *Master of Technology in VLSI Design* to VIT Vellore is a record of bona fide work carried out by me under the supervision of *Mr. Haricharan Kotagiri (External Guide).*

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 29/05/2023 **Signature of the Candidate**

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Date : 29/05/2023

To whomsoever it may concern

This is to certify that Mr. SHREYAS S BAGI of M.Tech in VLSI Design bearing the Registration Number 21MVD0086 have carried out the work entitled Duty Cycle Distortion (DCD) and Spice Simulation.

Under my supervision during the period of September 5th 2022 to June 26th 2023.

Name : Haricharan Kotagiri
Designation : Engineer, Senior Staff/Manager
Emp ID : 133309

Signature of Guide

Signature of the HoD :
Seal and Signature of the School Dean :

ABSTRACT

Synchronous circuits dominate the electronic world because clocking eases the design of circuits compared to asynchronous circuits. At the same time, clocking also introduces its share of challenges to overcome. No wonder, a tremendous amount of time and effort have been spent over the years on developing and implementing various types of clock distribution networks. A lot of time has also been spent on analyzing and addressing clock jitter due to power supply. And at the design level, a lot of thought goes into choosing the clock duty cycle when designing a circuit.

In terms of accuracy, SPICE simulations have always been held as the gold standard. But SPICE simulations are compute-time intensive and typically run on just small portions of a design. Instead, gate level simulation was used as the default signoff tool for chips until the turn of the 21st century. This worked well as most of the designs then were not very large or complex and the process nodes in use were 250nm or larger. As process nodes advanced and design size and complexity started growing, gate level simulation as a signoff tool started getting strained. A 7nm SoC can have 10 billion transistors, and to meet the power spec there are many clock domains, and multi-voltage power domains; resulting in aging issues like jitter, duty cycle distortion, insertion delay, reduced design margins, and increased process variation. To predict the impact of transistor aging requires knowing the circuit topology, switching activity, voltages and even temperature – a complex goal.

Static timing analysis (STA) took over as the default signoff tool and has worked well for the last two decades. But today's advanced-process-based designs are facing chip-signoff challenges due to limitations of STA and duty cycle distortions (DCD). While the intrinsic limitations of STA were always present, they did not pose practical issues when it came to signoff on less advanced process nodes. And while duty cycle distortions go hand in hand with clocking, they were either corrected with a DCD correcting circuit or were not serious enough to impact the proper functioning of a design.

We're entering an era where STA needs to be augmented for addressing DCD and increasing verification coverage for high confidence at chip signoff. Wouldn't it be great if overnight simulation runs on multi-millions of gates can deliver SPICE level accurate results? Infinisim ClockEdge tool explains how their analysis tools and methodology can deliver all of the above.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to my mentor Haricharan Kotagiri, Senior Staff Design Engineer, Qualcomm India Pvt. Ltd. for introducing me this area of Duty Cycle Distortion. His constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Static Timing Analysis (STA).

I would like to express my gratitude to Dr. G. VISWANATHAN, SANKAR VISWANATHAN, Dr. SEKAR VISWANATHAN, G V SELVAM, DR. RAMBABU KODALI, DR. PARTHA SHARATHI MALLICK and Dr. Sivanantham S, School of Electronics Engineering (SENSE), for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Vellore

Date: 29/05/2023 **SHREYAS S BAGI**

CONTENTS

CHAPTER 3

STA Concepts

Chapter 4

Delay

Chapter 5

Timing Checks

Chapter 6

Signal Integrity

Chapter 7

Duty Cycle Distortion

Chapter 8

Conclusion

LIST OF FIGURES

Chapter 1
Introduction

1.1 Nanometer Technology

In semiconductor devices, metal interconnect traces are typically used to make the connections between various portions of the circuitry to realize the design. As the process technology shrinks, these interconnect traces have been known to affect the performance of a design. For deep submicron or nanometer process technologies, the coupling in the interconnect induces noise and crosstalk - either of which can limit the operating speed of a design. The physical design should consider the effect of crosstalk and noise and the design verification should then include the effects of crosstalk and noise.

1.2 What is Static Timing Analysis?

Static Timing Analysis (STA) is method used to verify the timing of a digital design. An alternate approach used to verify the timing simulation which can verify the functionality as well as the timing of the design. The Timing analysis is composed of two methods – static timing analysis and dynamic timing analysis.

The STA is static since the analysis of the design is carried out statically and does not depend upon the data values being applied at the input pins. This is in contrast to simulation-based timing analysis where a stimulus is applied on input signals, resulting behaviour is observed and verified, then time is advanced with new input stimulus applied, and the new behaviour is observed and verified and so on.

The STA is used to validate if the design can operate at the rated speed i.e., the design can operate safety at the specified frequency of the clocks without any timing violations. Figure 1.1 shows the basic functionality of static timing analysis. The DUA is the design under analysis. The timing checks are setup and hold checks. The setup check is to ensure the data can arrive at a flip-flop within given Clock period (Tclk). The Hold check is to ensure

flip-flop capture the intended data correctly. The setup and hold check ensure that the proper data is captured and latched in for new state.

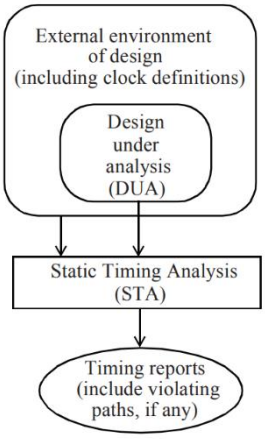


Figure 1.1: Static timing analysis

The DUA is specified using Verilog or VHDL language. The external environment is specified using SDC (constraints file). The .sdc file provides timing constraints, environment conditions and multi-voltage-based design. The SDC stands Synopsys Design Constraints, is a timing constraint specification language.

1.3 Why Static Timing Analysis ?

Static timing analysis is a complete and exhaustive verification of all timing checks of a design. To simulate and verify all timing conditions of a design with 10-100 million gates is very slow with timing simulation and the timing cannot be verified completely. Thus, it is very difficult to do exhaustive verification through simulation.

Static timing analysis on the other hand provides a faster and simpler way of checking and analysing all the timing paths in a design for any timing violations. Given the complexity of present-day ASICs , which may contain 10 to 100 million gates, the static timing analysis has become a necessity to exhaustively verify the timing of a design

The design functionality and its performance can be limited by noise. The noise impact can limit the frequency of operation of the design and it can also cause functional failures.

Verification based upon logic simulation cannot handle the effects of crosstalk, noise and on-chip variation. The STA can consider the crosstalk and noise effect for analysis to verify the design.

1.4 STA at various Design Phases

Once a design at the RTL level has been synthesized to the gate level, the STA is used to verify the timing of the design. STA can also be run prior to performing logic optimization - the goal is to identify the worst or critical timing paths. STA can be rerun after logic optimization to see whether there are failing paths still remaining that need to be optimized, or to identify the critical paths.

At the start of the physical design, clock trees are considered as ideal, that is, they have zero delay. Once the physical design starts and after clock trees are built, STA can be performed to check the timing again. In physical implementation, the logic cells are connected by interconnect metal traces. The parasitic RC (Resistance and Capacitance) of the metal traces impact the signal path delay through these traces. Thus, any analysis of the design should evaluate the impact of the interconnect on the performance characteristics (speed, power, etc.).

An extraction tool is used to extract the detailed parasitics (RC values) from a routed design. In final verification during which very accurate RC values are extracted with a larger runtime. In Figure 1.2 shows that STA at various stages in Design flow.

At the logical level (gate-level, no physical design yet), STA can be carried out using:

i. Ideal interconnect or interconnect based on wireload model.
ii. Ideal clocks with estimates for latencies and jitter.

During the physical design phase, in addition to the above modes, STA can be performed using:

i. Interconnect - which can range from global routing estimates, real routes with approximate extraction, or real routes with signoff accuracy extraction.
ii. Clock trees - real clock trees.
iii. With and without including the effect of crosstalk.

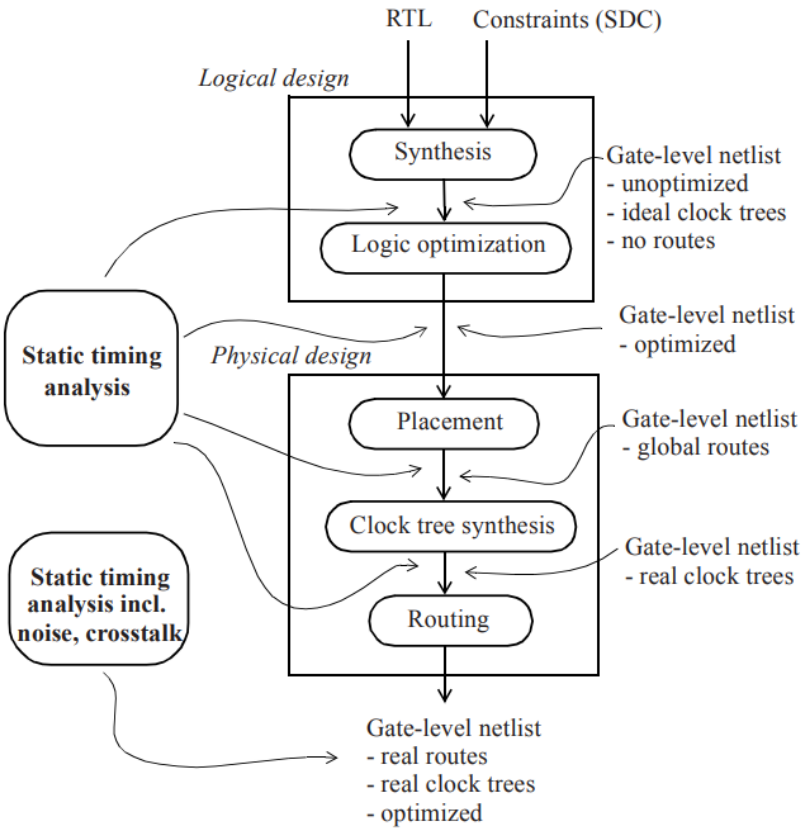


Figure 1.2: Design Flow

1.5 Limitations of STA

i. Reset Sequence:

This cannot be checked. Initial value on signals is not synthesized. These can be only verified during simulation or simulation-based timing analysis should be used.

ii. X-handling:

The Glitch analysis, Propagation delays and Noise analysis this can be verified by STA but Logic X or unknow logic can't be verified using STA. This kind of logic verified by Simulation based timing.

iii. PLL:

The PLL configuration are loaded during boot-load sequence. So, this cannot be verified.

iv. Asynchronous Clock domain crossings:
STA cannot check if correct clock synchronizers are being used.

v. IO interface timing:
It is not possible to specify the IO interface requirements in terms of STA constraints.

vi. Interface between analog and digital blocks:
STA is not used to verify analog blocks. If there is some connectivity between analog and digital block, then it cannot be verified using STA.

vii. False Path:
The STA verifies that timing constraints are met by all logic paths and reports if any logical path doesn't meet required specification.

There are some logical paths based on some condition only the logic is propagated through the path. The paths are mutually contradictory conditions are used during sensitization of the path. These timing paths are referred as false paths.

The designer needs to specify the false path in his constraints file (.sdc or set_false_path -help command). The quality of STA results is better when false path is specified.

viii. FIFO pointers out of sync :

The delays in the circuit can cause the FSM to be out of sync with other, because one FSM comes out of reset sooner than other. This situation cannot be detected in STA.

ix. Clock Synchronization logic :

STA cannot detect the problem of clock generation logic not matching the clock definition. STA assumes that clock generator provides clock clean and without any

delay. Due, to some optimization errors there is a large logic block added at Clock generator circuit this leads to addition of delay or insertion delay on the paths.
This insertion delay may change the duty cycle of the clock (DCD). The STA cannot detect these problems.

x. Functional behaviour across clock cycles:

STA cannot verify or model or simulate functional behaviour that changes across clock cycles.

Chapter 2

STA Flow

In this section we describe about STA Flow used for design. The STA signoff toll needs to be run separately for all the timing corners after synthesizing and doing final PnR netlist. This ensures that timing is met across all corners, including those that haven't been checked in synthesis or the PnR flow. The flow for running STA is shown in Figure 2.1.

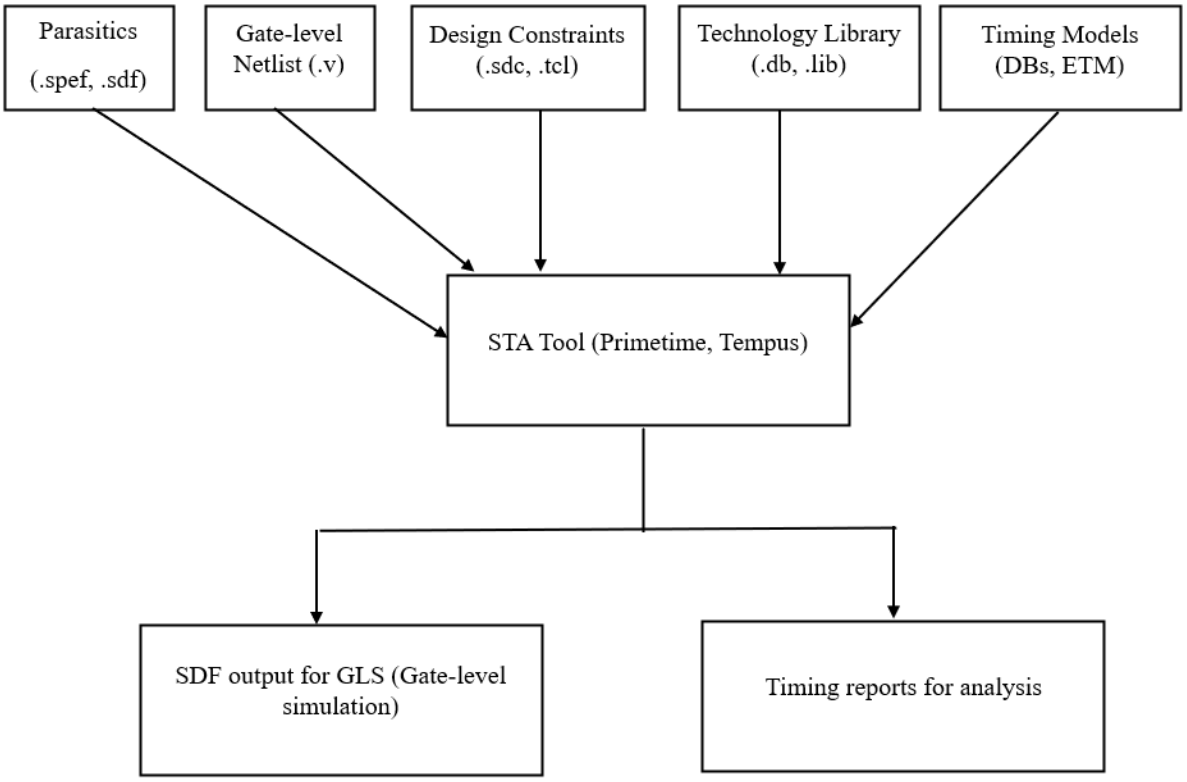


Figure 2.1: STA Environment with Inputs and Outputs

2.1 STA Inputs

The STA Inputs is comprising of following types:

- Netlist (verilog/VHDL): Synthesis is the process of converting RTL to technology specific gate level netlist. The gate level circuit description in verilog or VHDL language.

- Constraints: Sets of constraints to enable timing checks of the paths that dictate the desired performance of the design.
- Technology library: Technology file defines basic characteristic of cell library pertaining to a particular technology node. The technology library includes timing information and contains several other attributes.
- Parasitics: The nets have parasitic resistance and capacitance that are captured by SPEF.
- Timing Models: These are known as Process, Voltage and temperature variations. The derate models the effects of varying operating conditions.

2.2 STA Outputs

The STA Outputs is comprising of following types:

- Sanity Check Logs: These logs are intended to validate quality of the Inputs.
- Timing window report: Setup, Hold and SI reports.
- DRC reports: Max Transition, Max capacitance checks are involved.
- SDF (Standard Delay Format): This is output for GLS (Gate level simulation).

2.3 Netlist

The gate level circuit description as an example is shown in Figure 2.2.

```
module mux_1 ( i0, i1, s, y );
  input [7:0] i0;
  input [7:0] i1;
  output [7:0] y;
  input s;
  wire   n1, n2;

  SAEDRVT14_BUF_U_0P5 U1 ( .A(s), .X(n1) );
  SAEDRVT14_INV_0P5 U2 ( .A(n1), .X(n2) );
  SAEDRVT14_MUXI2_U_0P5 U3 ( .D0(n2), .D1(n1), .S(i0[0]), .X(y[0]) );
  SAEDRVT14_MUXI2_U_0P5 U4 ( .D0(n2), .D1(n1), .S(i0[1]), .X(y[1]) );
  SAEDRVT14_MUXI2_U_0P5 U5 ( .D0(n2), .D1(n1), .S(i0[2]), .X(y[2]) );
  SAEDRVT14_MUXI2_U_0P5 U6 ( .D0(n2), .D1(n1), .S(i0[3]), .X(y[3]) );
  SAEDRVT14_MUXI2_U_0P5 U7 ( .D0(n2), .D1(n1), .S(i0[4]), .X(y[4]) );
  SAEDRVT14_MUXI2_U_0P5 U8 ( .D0(n2), .D1(n1), .S(i0[5]), .X(y[5]) );
  SAEDRVT14_MUXI2_U_0P5 U9 ( .D0(n2), .D1(n1), .S(i0[6]), .X(y[6]) );
  SAEDRVT14_MUXI2_U_0P5 U10 ( .D0(n2), .D1(n1), .S(i0[7]), .X(y[7]) );
endmodule


module ha ( s, c, a, b );
  input a, b;
  output s, c;


  SAEDRVT14_AN2_MM_0P5 U1 ( .A1(b), .A2(a), .X(c) );
  SAEDRVT14_OA21B_1 U2 ( .A1(b), .A2(a), .B(c), .X(s) );
endmodule
```

Figure 2.2: Gate level circuit description for Multiplexer (8:1) and Half adder circuit.

2.4 Design Constraints

The Synopsys Design Constraints (SDC) format is used to specify the design intent, including timing, power and area constraints of the design. These constraints are used to enable timing checks of the paths for the desired performance of the design. These consists of set of commands or syntax that define particular property or attribute for particular timing check for the path.

The Design constraints consists of

- Clock definitions
- Input delay
- Output delay
- Max and Min delay
- Multi-cycle path
- False path
- Disable timing
- Case analysis

2.5 Technology Library

The Technology file defines characteristic of cell library pertaining to a particular technology node. In addition to timing information, the library cell description contains several other attributes.

The library has below information :

- Units
- Operating conditions
- NLDM and CCS delay models
- Capacitance values
- Noise voltage and current waveforms CCSN
- Internal Cell Power

The Figure 2.3 shows an example of technology library comprised of the above listed things.

```
library("my_cell_library") {
 voltage_unit : "1V";

 time_unit : "1ns";
 capacitive_load_unit (1.000000, pf);
 current_unit : 1mA;
 pulling_resistance_unit : "1kohm";
 . . .
}

operating_conditions("BCCOM"){
 process : 1;
 temperature : 0;
 voltage : 1.1;
 tree_type : "balanced_tree";
}


 cell_leakage_power : 0.70;
 leakage_power() {
  when : "!I";
  value : 1.17;
 }

 leakage_power() {
  when : "I";
  value : 0.23;
 }
```

```
pin (OUT) {
 . . .
 timing () {
  related_pin : "IN1";
  . . .
  ccsn_first_stage() {
   /* IN1 to internal node between stages */
   . . .
  }
  ccsn_last_stage() { /* Internal node to output */
   . . .
  }
  . . .
 }
```

```
pin (OUT) {
 . . .
 timing () {
  related_pin : "IN"-;
  . . .
  output_current_fall () {
   vector ("LOOKUP_TABLE_1x1x5") {
    reference_time : 5.06; /* Time of input crossing
     threshold */
    index_1("0.040"); /* Input transition */
    index_2("0.900"); /* Output capacitance */
    index_3("5.079e+00, 5.093e+00, 5.152e+00,
            5.170e+00, 5.352e+00");/* Time values */
    /* Output charging current: */
    values("-5.784e-02, -5.980e-02, -5.417e-02,
           -4.257e-02, -2.184e-03");
   }
```

Figure 2.3: Technology Library information.

2.6 Parasitics

In digital designs, a wire connecting pins of standard cells and blocks is referred to as a net. A net typically has only one driver while it can drive a number of fanout cells or blocks. After physical implementation, the net can travel on multiple metal layers of the chip. Various metal layers can have different resistance and capacitance values. For equivalent electrical representation, a net is typically broken up into segments with each segment represented by equivalent parasitics.

There are two types of parasitics:

i. Wire-load based: The wire load models can be used to estimate capacitance, resistance, and the area overhead due to interconnect. The wireload model is used to estimate the length of a net based upon the number of its fanouts. The wireload model depends upon the area of the block, and designs with different areas may choose different wireload models.

ii. Extracted Parasitics : The parasitics extracted from a layout can be present in three formats i.e., Detailed Standard Parasitic Format (DSPF), Reduced Standard Parasitic Format (RSPF) and Standard Parasitic Extraction Format (SPEF). The SPEF is compact and complete representation of parasitics. The SPEF is used in industry.

2.7 Timing Models

The design is becoming highly complex and large in nature. In a design once a block has met its timing constraints at the gate level, the detailed internal timing of the block is not needed for the chip-level timing analysis.

The timing models involved are :

i. Extracted Timing Models (ETM)
ii. Interface Logic Models (ILM)

(a) Extracted Timing Model (ETM)

As shown in Figure 2.4, ETMs use abstraction to minimize the amount of data while attempting to preserve accuracy. ETMs replace respective blocks in hierarchical timing analysis, which significantly speed-up analysis and reduce the memory footprint for the full-chip analysis.

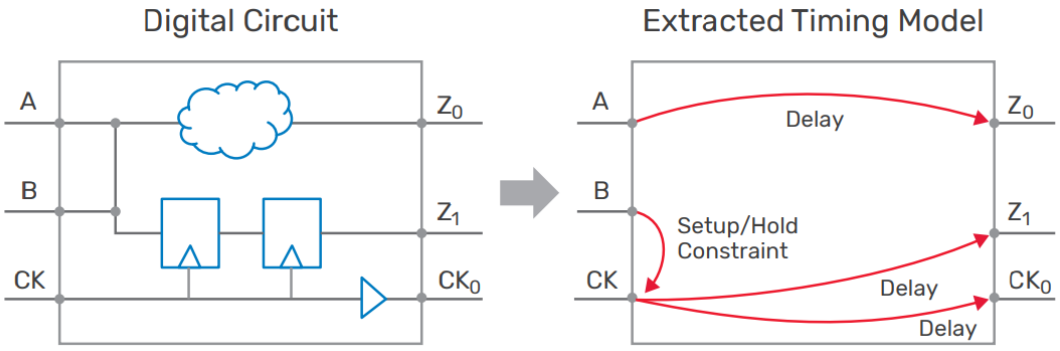


Figure 2.4: ETM. (Source: Cadence White Paper)

The ETM has many advantages like:

- Enable IP reuse and interchange of timing models in EDA tools.
- Black-box timing model can be done using ETMs.

- The timing arcs between external pins and internal pins only for generated or internal clocks.
- These are context independent.

(b) Interface Logic Model

ILMs remove the register-to-register logic and preserve the rest of the interface logic in the model as shown in Figure 2.5. The components within an ILM include a netlist, parasitic loading, constraints, and aggressor information pertinent to the preserved logic inside the ILM. ILMs are highly accurate and can also speed up analysis considerably, while reducing the memory footprint.

Because ILMs preserve the interface logic exactly the same way as the original netlist, they can deliver exactly the same timing for interface paths as a flat analysis. Contextual impacts to the interface logic are observed by merging the ILM netlist, parasitic, and constraints into the chip-level components. ILM limitations include over-the-block routing, constraint mismatches, data management, arrival pessimism, latch-based designs, specific flows, and special stitching.

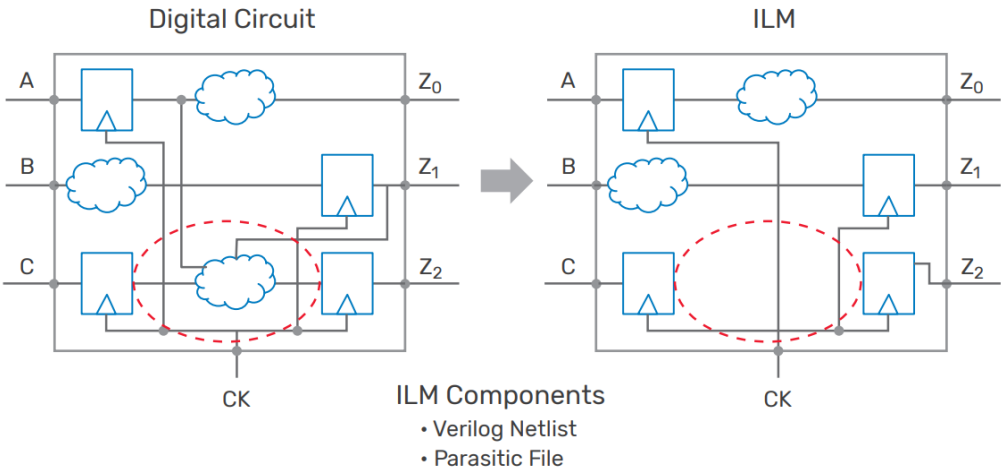


Figure 2.5: ILM components. (Source: Cadence White Paper)

2.8 Sanity Checks

To ensure that the input received from the library team and synthesis team is correct or not. If we are not doing these checks then it creates problems in later stages of design.

The following input files are checked :

i. Design and Library check : Check if current design is consistent or not. The library check involves validate the library i.e., it checks the consistency between logical and physical libraries.

ii. SPEF annotation

iii. Constraint validation: This involves SDC checks like check unconstrained endpoints, any ports are missing input/output delay, port missing slew/load constraints or check clock is reaching to all the flops.

2.9 Timing Window report

The timing window report involves setup time check, hold time check, multicycle path check, asynchronous timing check. These checks are discussed in detail in Chapter 5.

2.10 DRC Checks

The Design Rule Checking (DRC) verifies as to whether a specific design meets the constraints imposed by the process technology to be used for its manufacturing. DRC checking is an essential part of the physical design flow and ensures the design meets manufacturing requirements and will not result in a chip failure. The process technology rules are provided by process engineers and/or fabrication facility.

The DRC checks generally involves following checks:

- Minimum width
- Minimum spacing
- Minimum area
- Wide metal jog
- Misaligned via wire
- Special notch spacing
- End of line spacing

Chapter 3

STA Concepts

This section describes the standard cells, propagation delays, timing models, slew rate, skew, clock uncertainty concepts needed to understand Static timing analysis.

3.1 Standard Cells

The chips are designed using basic building blocks which implement simple logic functions such as and, or, nand, nor, and-or-invert, or-and-invert and flip-flop. These basic building blocks are pre-designed and referred to as standard cells. The functionality and timing of the standard cells is pre-characterised and available to the designer.

The Cell Library is large collection of standard cells with detailed information about the cell. The library cell or cell logic model consists of :

i. Timing parameters.
ii. Cell area.
iii. Functionality of cell.
iv. Power parameters.
v. Cell name, pins, pin directions.

In the Figure 3.1 shows the Cell library or Cell model importance in Digital design flow (Reference : Synopsys Inc.)

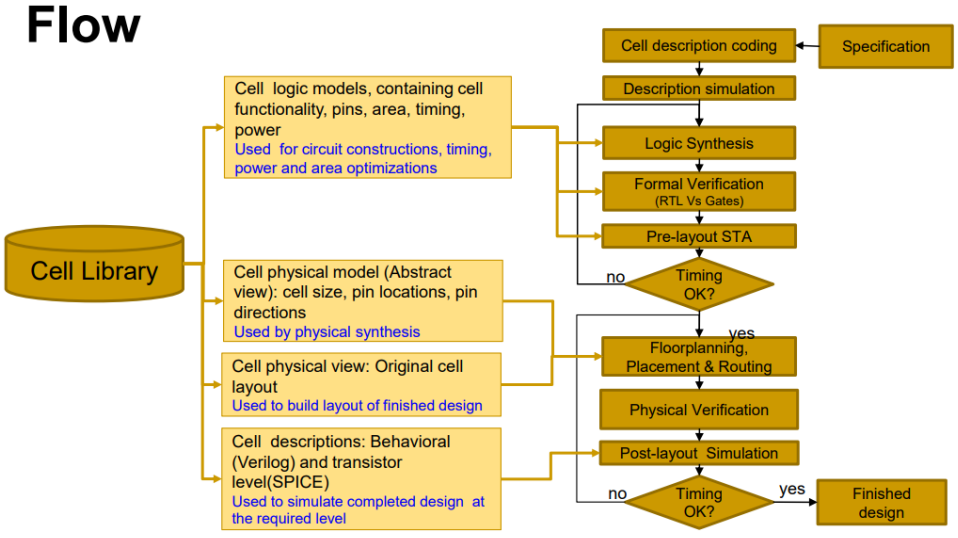


Figure 3.1: The Cell Library or Cell model importance in Digital design flow. (Source : Synopsys Inc.)

In Synopsys the library information is stored in Liberty file structure (.lib), which contains cell description, Environment description and Technology related information. In Figure 3.2 shows the liberty .lib File structure.

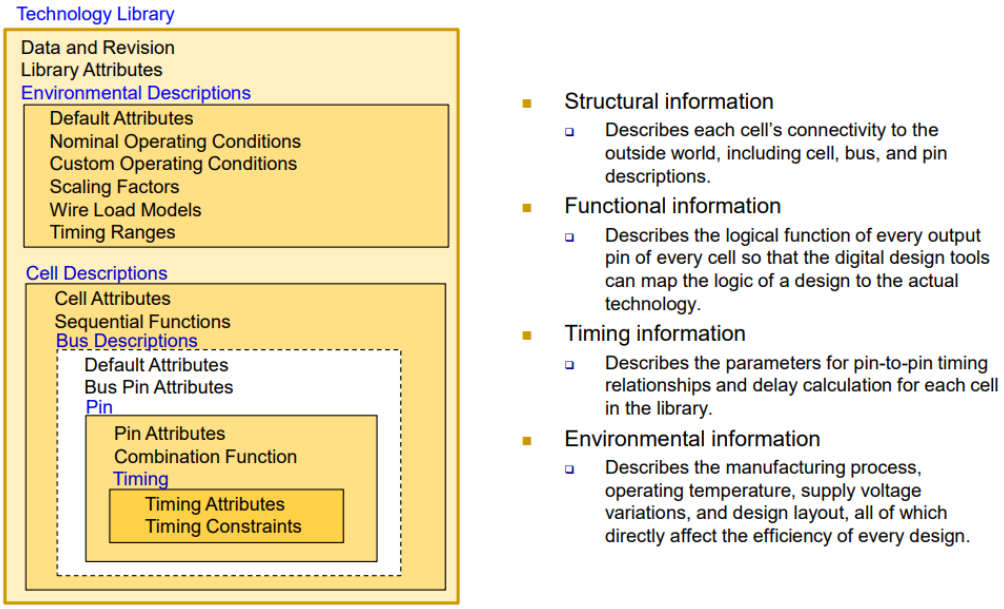


Figure 3.2: The Liberty .lib file structure. (Source: Synopsys Inc.)

3.2 Modelling CMOS

If a cell output pin drives multiple fanout cells, the total capacitance on the output pin of the cell is the sum of all the input capacitances of the cells that it is driving plus the sum of the capacitance of all the wire segments that comprise the net plus the output capacitance of the driving cell.

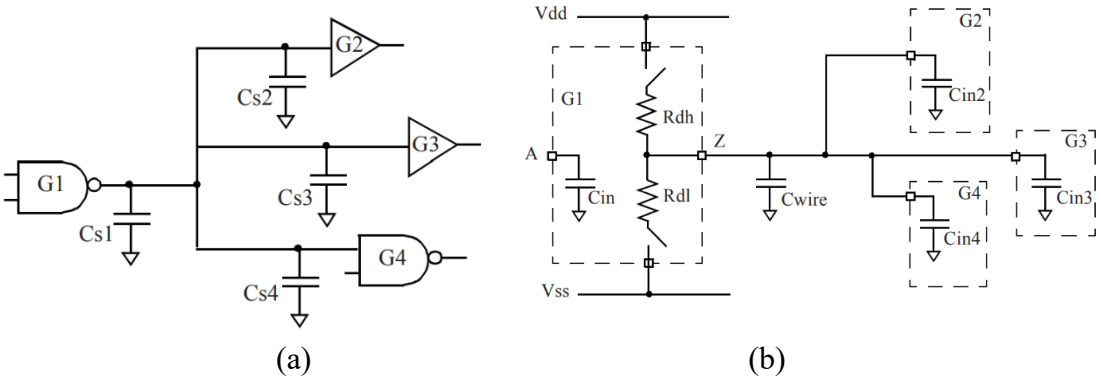


Figure 3.3 Modelling CMOS (a) Capacitance of net (b) Net with CMOS equivalent models.

The CMOS cell switches states, the speed of the switching is dependent on how fast the capacitance on the output net can be charged or discharged. The capacitance can be charged by pull-up and pull-down path separately.

$C_{wire} = C_{s1} + C_{s2} + C_{s3} + C_{s4}$.

Output charging delay = $C_{wire} * R_{out}$

The capacitance values of standard cell are described in cell library. The library description of the cell, the capacitance of cell, the capacitance for cell inputs only specified and output capacitance is set to zero. In Figure 3.4 shows an example capacitance description of a pin.

```
pin (INP1) {
  capacitance: 0.5;
  rise_capacitance: 0.5;
  rise_capacitance_range: (0.48, 0.52);
  fall_capacitance: 0.45;
  fall_capacitance_range: (0.435, 0.46);
  . . .
}
```

Figure 3.4: The pin INP1 capacitance description.

3.3 Propagation Delays

The delay between the input threshold point to output threshold point. The propagation delay (t_{pd}) means delay between the two edges (threshold points). The threshold points value is part of liberty files.

For example, in inverter cell and the waveforms at its pins shown in Figure 3.5. The propagation delays are represented as :

(i) Output fall delay (T_f).

(ii) Output rise delay (T_r).

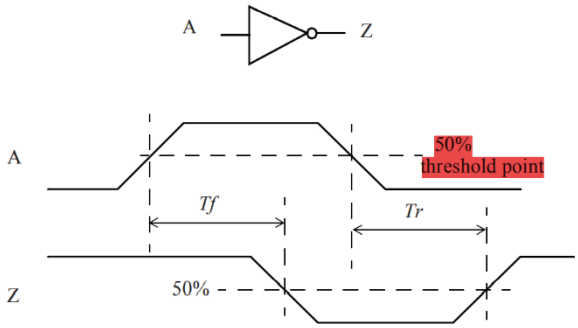


Figure 3.5: Propagation Delay.

The delay of the cell depends on :

i. Output load (C_L).

ii. Input signal Transition time. (T_{trans_in})

The delay is directly proportional with load capacitance and input transition time. The relation of Delay is shown below :

Delay (D or t_d) = $F(C_L, T_{trans\_in})$.

In Figure 3.6 shows delay dependences of an inverter cell (Source : Synopsys Inc.).

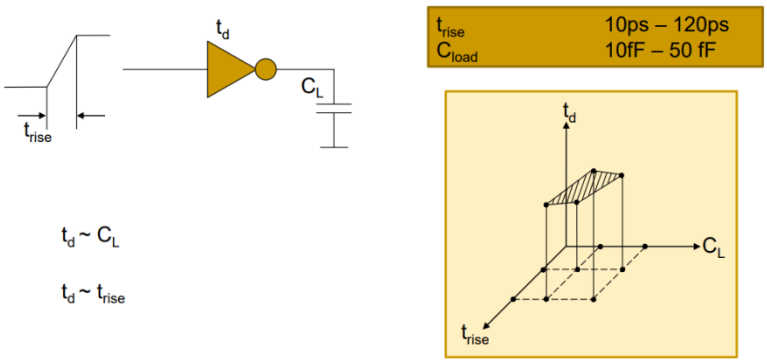


Figure 3.6 : Delay dependencies of an Inverter cell. (Source : Synopsys Inc.).

3.4 Timing Delay Model

The models are obtained from detailed circuit simulations of the cell to model actual working of the cell in various scenarios. The models are specified for each timing arc of the cell.

The delays are measured based upon the threshold points defined in a cell library. By default, it is set to 50% Vdd. For example, timing arc delay for Inverter cell is shown in Figure 3.7. In this figure we can see Output rise delay (T_r) and Output fall delay (T_f).

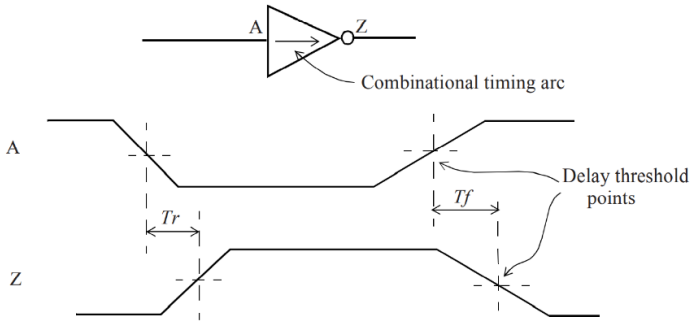


Figure 3.7: Delay of an Inverter cell.

3.4.1 Linear Delay Model

In linear delay model, the delay and Output transition time both are function of Load capacitance and Input signal transition time. The mathematical model is given as follows :

$$D = k_1 + \left(k_2 * T_{trans\_in}\right) + (k_3 * C_L).$$

The k_1, k_2 and k_3 are constant values. The linear delay models are not used because they are inaccurate over the range of T_{trans_in} and C_L for submicron technology.

3.4.2 Non-linear Delay Model (NLDM)

In this model cell libraries include table model or Look-up table with value. The table is used to specify delays and timing checks for various timing arcs of cell. The table used for delay, output slew, or other timing checks. The NLDM model for delay is presented in a 2D form, with two independent variables being input transition time and output load capacitance and entries in table correspond to delay. In Figure 3.8 : shows an example a part of NLDM for a pin of the cell.

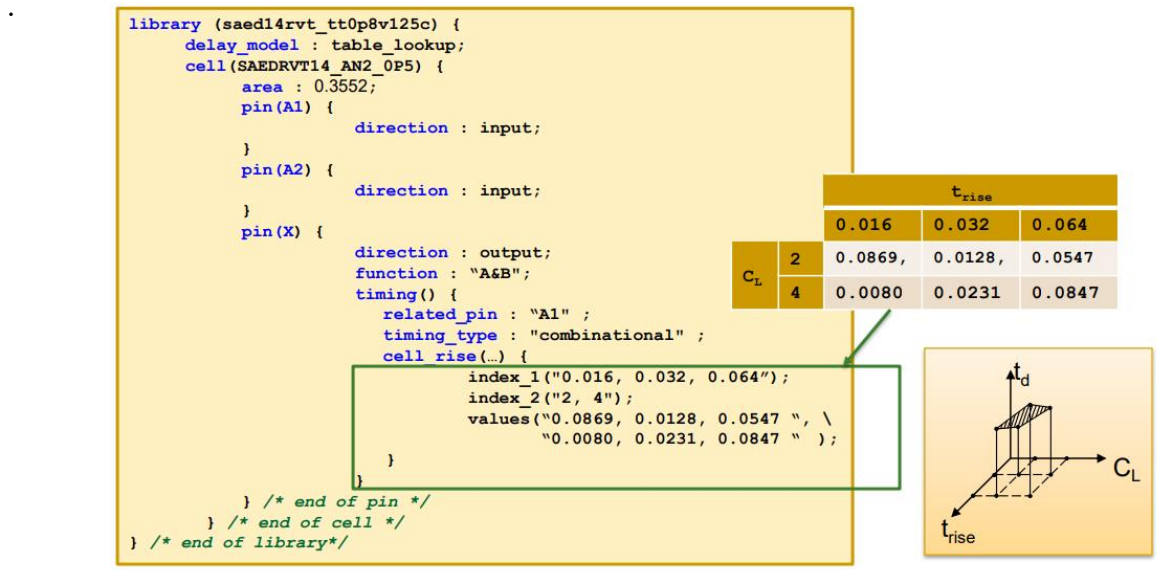


Figure 3.8: Synopsys Liberty Format (.lib) of and cell.

3.5 Slew rate

A slew rate is defined as a rate of change. In static timing analysis, the rising or falling waveforms are measured in terms of whether the transition is slow or fast. The slew is typically measured in terms of the transition time i.e., the time it takes for a signal to transition between two specific levels. The transition time is defined w.r.t to specific threshold levels. For example, the slew threshold values are :

Falling edge thresholds :

slew_lower_threshold_fall : 30.0 ;

slew_upper_threshold_fall: 70.0 ;

Rising edge thresholds :

slew_lower_threshold_rise: 30.0 ;

slew_upper_threshold_rise: 70.0;

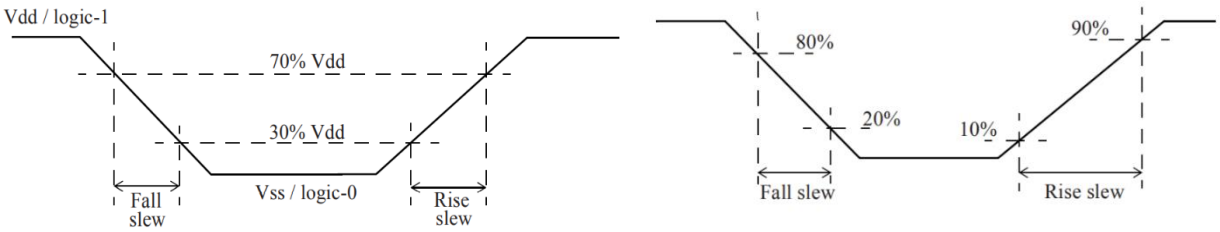


Figure 3.9: Rise and Fall transition time.

3.6 Skew in different signals

Skew is the difference in timing between two or more signals, maybe data, clock or both. The beginning point of a clock tree typically is a node where a clock is defined. The end points of a clock tree are typically clock pins of synchronous elements such as flip-flops. The Clock Latency is total time from clock source to an end point. The Clock Skew is the difference in arrival time at the end points of the clock tree. In Figure 3.10 shows the Clock skew and Clock latency definitions.

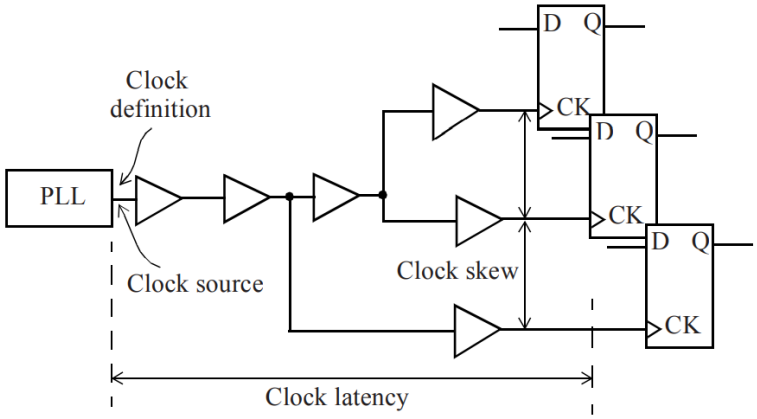


Figure 3.10: Clock tree, Clock skew and Clock latency.

3.7 Timing arcs

Every cell has multiple timing arcs. For combinational logic, has timing arcs from each input pins to output pin of the cell. In sequential logic, such as flip-flops have timing arcs from the clock to the outputs and timing constraints for the data pins w.r.t clock.

The timing arcs are categorized into three types :

(i) Positive unate arc.

(ii) Negative unate arc.

(iii)Non-unate arc.

The timing arc is positive unate is a rising transition on an input causes the output to rise. The and gate, or gate follow positive unate arc. In negative unate is a falling transition on an input causes the output to fall. The nand gate, nor gate follow negative unate arc. If the output transition cannot be determined solely from the direction of change of an input but also depends upon the state of the other inputs these are termed as non-unate arcs. The xor gate, xnor gate follow non-unate arcs.

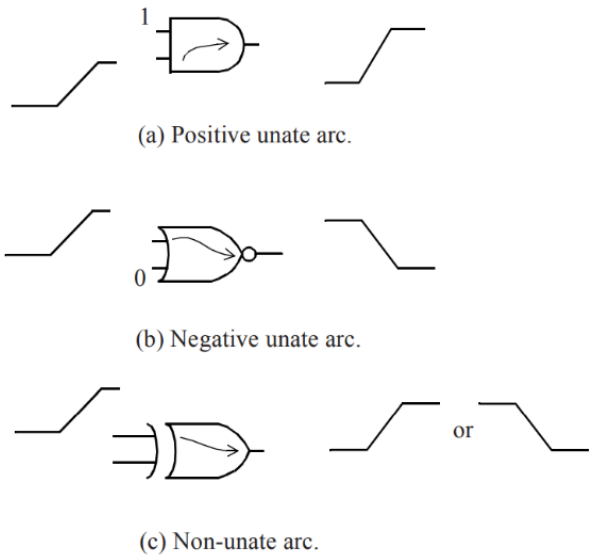


Figure 3.11: Timing arcs of different gates.

3.8 Path delay

The path delay is defined as sum of the delays through the various logic cells and nets along the paths. In general, there are multiple paths through which the logic can propagate to the required destination point. The actual path taken depends upon the state of the other inputs along the logic path.

In Figure 3.12 we observe there are multiple paths to the destination, the maximum and minimum timing to the destination points can be obtained. The max path is the longest delay between two end points. Similarly, a min path is the smallest delay between two end points.

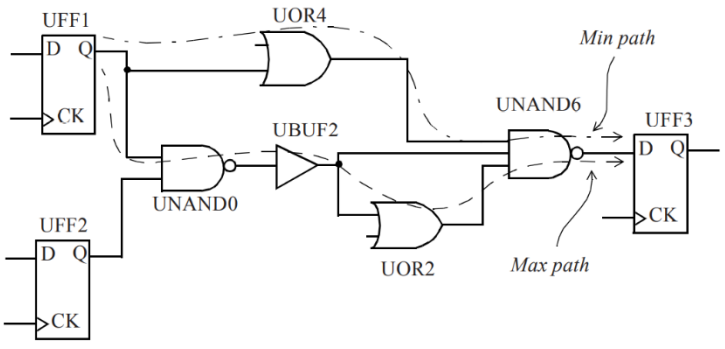


Figure 3.12 : Max and Min timing paths.

In Figure 3.13 is an example showing calculation of Path delay.

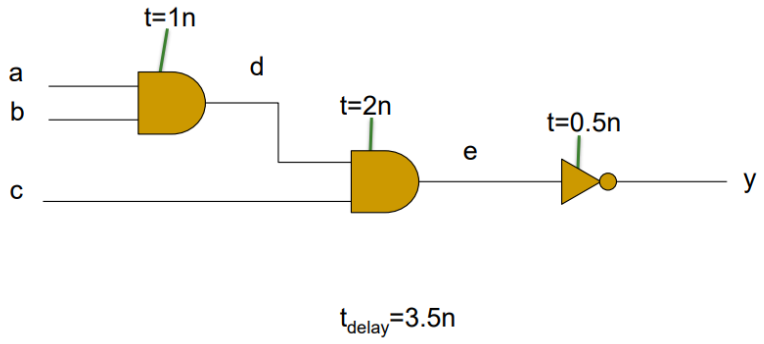


Figure 3.13 : The path delay calculation of the circuit.

3.9 Clock domain

In a chip, there are many clocks defined for the design. The clock feeds a number of flip-flops. The set of flip-flops fed by one clock is called its clock domain. In chips, there are numerous clock domain. In Figure 3.14 we see some flip-flops are clocked by USBCLK and other flip-flops are clocked by MEMCLK.

The clock domain can be independent or related of each other. If there no paths between the two clock domains then they are independent of each other. This means that there is no timing path that starts from one clock domain and ends in other clock domain.

If there are data paths that cross between clock domains, a decision has to be made as to whether the paths are real or not. In Figure 3.14 a clock domain crossing can occur both ways, from USBCLK clock domain to MMCLK clock domain, and from MEMCLK clock domain to USBCLK clock domain. Both scenarios need to be understood and handled properly in STA.

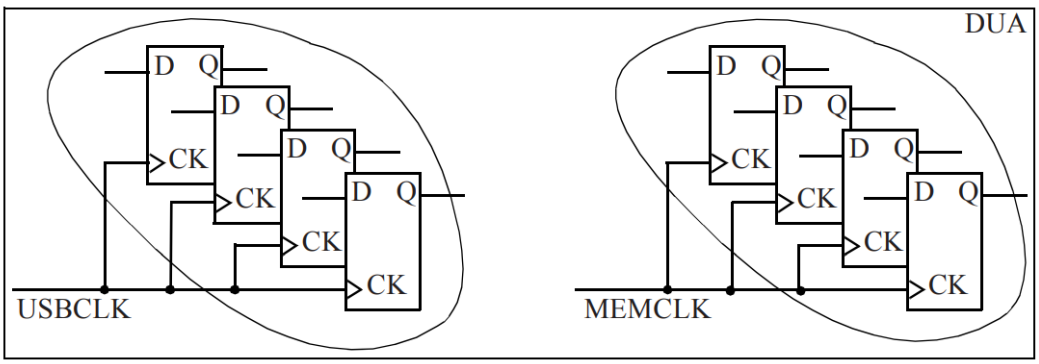


Figure 3.14: Clock domain crossing.

3.10 Clock Uncertainty

The clock uncertainty is defined as sum of clock jitter, clock skew and additional pessimism. These are used for timing verification.

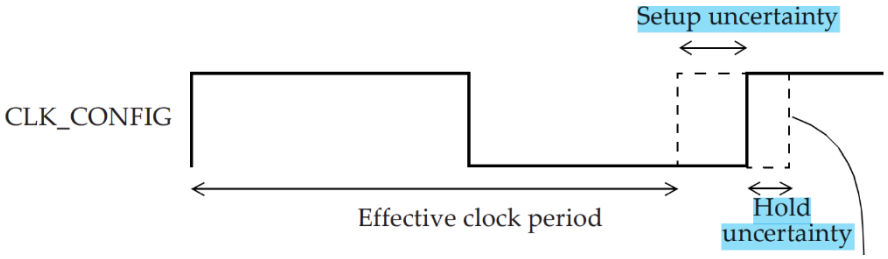


Figure 3.15: Specifying the clock uncertainty.

3.11 Operating conditions

The operating condition is defined as a combination of Process, Voltage and Temperature (PVT). The cell delays and interconnect delays are computed based upon the specifying operating condition. The three kinds of manufacturing process are slow, typical and fast process models. The slow and fast process models represent the extreme corners of the manufacturing process of a foundry.

In Figure 3.16 depicts delay variation with PVT.

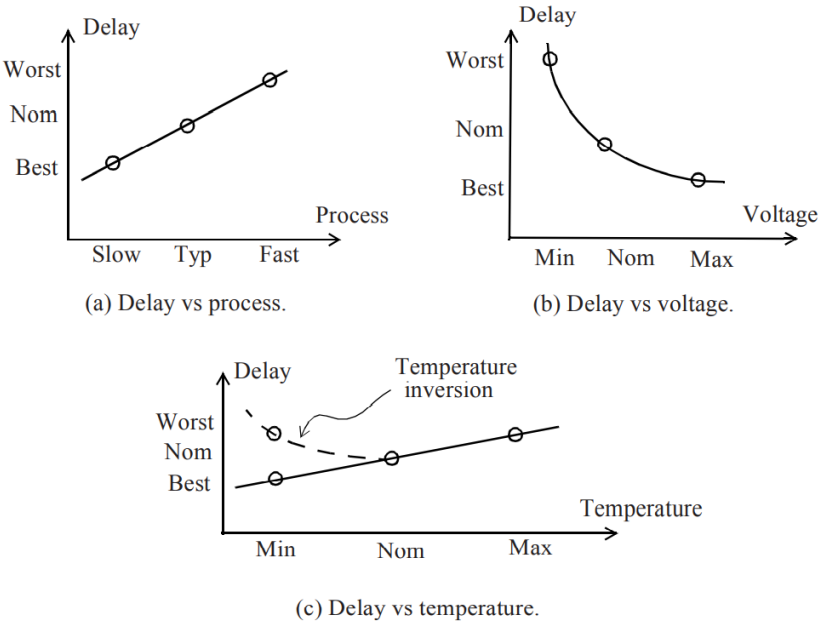


Figure 3.16 Delay variation with PVT.

A cell library specifies the characterization and operating conditions under which the library created. In Figure 3.17 shows the PVT corners of the inverter cell. For example, the header of the library may contain the following:

nom_process : 1; # Nominal environment conditions

nom_temperature **:** 125; # Temperature profile in which design will operate.

nom_voltage : 1.1; #The operating voltage of the design.

voltage_map(COREVDD1, 1.1);

voltage_map(COREGND1, 0.0);

operating_conditions("BCCOM"){

process **:** 1;

temperature **:** 0;

voltage **:** 1.1;

tree_type **:** "balanced_tree"; #The interconnect model.

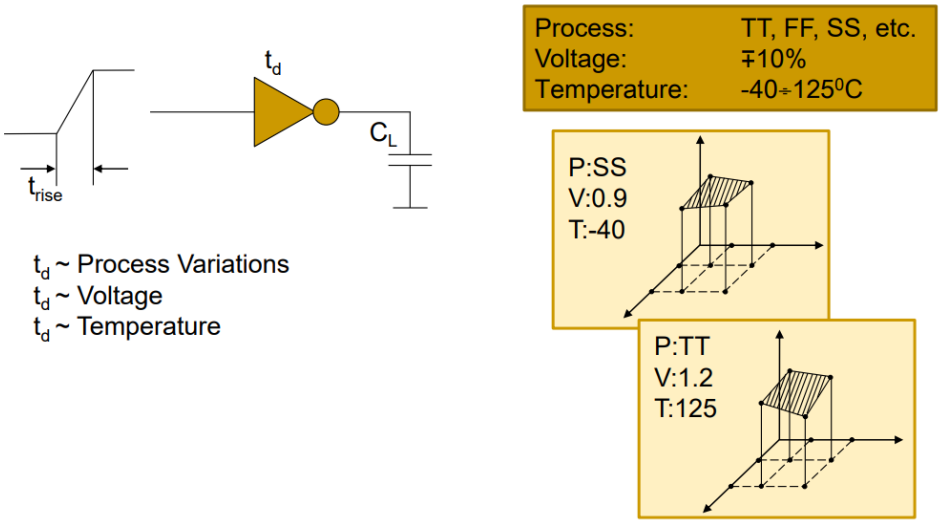


Figure 3.17: The PVT corners defined for Inverter cell.

1. **Worst Case :**

   Process – slow, Temperature – high (125 C) and Voltage – low.

   Due to temperature inversion concepts, the delay increases at lower temperature. This anomaly is present in nanometer technology nodes less than 32nm. The Vth (Threshold

voltage) margin w.r.t power supply is reduced for nanometer technology. At low power supply, the delay of lightly loaded cell is higher at low temperatures than at high temperatures.

2. **TYP (Typical) :**

   Process – typical, Temperature – nominal (25 C) and Voltage – nominal.

3. **Best-Case Fast (BCF) :**

   Process – Fast, Temperature – lowest(-40 C) and Voltage – highest (1.2V or 1.1% of Vdd).

   Environment Condition for Power analysis :

   1. **ML (Maximal Leakage)** :

   Process – Fast, Temperature – highest and Voltage – highest. This corner corresponds to Maximum leakage power. This also corresponds to the largest active power.

   2. **TL (Typical Leakage)** :

   Process – Typical, Temperature – highest and Voltage – nominal

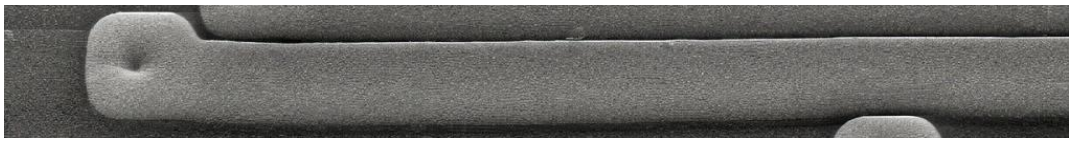
Chapter 4

Delay

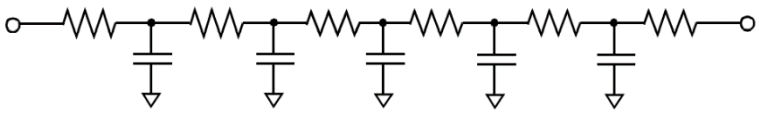
4.1 Interconnect Parasitics

In chips the net is routed based upon the logic connectivity of the design. The net has one driver and multiple fanout cells or blocks. The physical implementation of the net can travel on multiple metal layers on the chip. The metal layers are of different types with varying resistance and capacitance value depending on the length and width of the trace the value varies.

The net in the chip is represented equivalently in electrical representation as shown in Figure 4.1. The net is broken up into segments with each segment representing by the parasitics.



Signal trace viewed under Scanning Electron Microscope (SEM).



Distributed RC tree

Figure 4.1: Interconnect trace.

The interconnect is physically modelled using all the three parasitics (R, L and C). The resistance (R) in interconnect trace (Via or Metal layers) is due to resistance between the output pin of a cell and the input pins of the fanout cells.

The capacitance (C) is due to metal traces, ground capacitance and capacitance between neighbouring signal routes (Coupling capacitance).

The inductance (L) is due to current loops, the effect of inductance can be ignored within chip because presence of narrow and short current loop. The return path for current is through a power or ground signal routed in close proximity.

4.2 RC Interconnect Model

The RC interconnect can be represented by T-model and Pi-model.

T-model : The total capacitance C_t is modelled as connected halfway in the resistive tree. The total resistance R_t is broken in two sections with C_t represented at the mid-point of the resistive tree as shown in Figure 4.2.

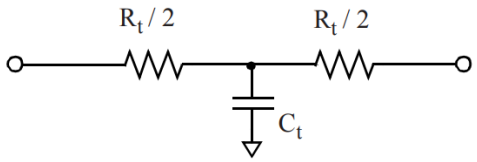


Figure 4.2: T-model representation.

Pi-model : The total capacitance R is modelled as connected halfway in the resistive tree. The total resistance C_t is broken in two sections with R_t represented at the mid-point of the resistive tree as shown in Figure 4.3.

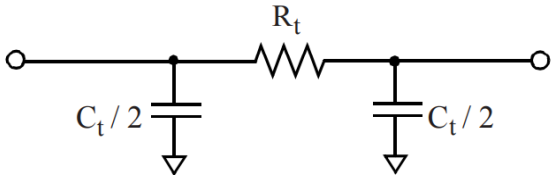


Figure 4.3: Pi-model representation.

4.3 Wireload Models

The wireload model are estimate capacitance, resistance and area overhead due to interconnect. The model used to estimate the length of a net based upon the number of fanout. The type of wireload depends on the area of the block or design. In Figure 4.4 shows for different areas (chips or block size), different wireload models would be used in determining the parasitics.

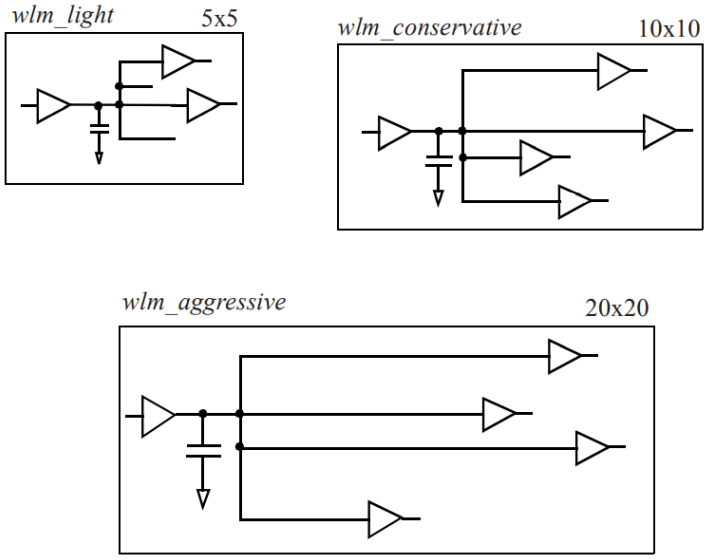


Figure 4.4: Different wireload models for different areas.

4.4.1 Interconnect Trees

For pre-layout estimation, the interconnect RC tree can be represented as

i. Best-case tree.

ii. Balanced tree.

iii. Worst-case tree.

The Figure 4.5 shows the RC tree representation of Best-case, Balanced and worst-case tree.

Best-case tree:

The load pin is physically adjacent to the driver. Thus, there is no wire resistance (R=0) in the path to the destination pin. There is wire capacitance and ground capacitance.

Balanced tree:

In design each path to the destination sees an equal portion of the total wire resistance and capacitance.

Worst-case tree:

In this case all the destination pins are together at the far end of the wire. Therefore, each destination pin sees the total wire resistance and the total wire capacitance.

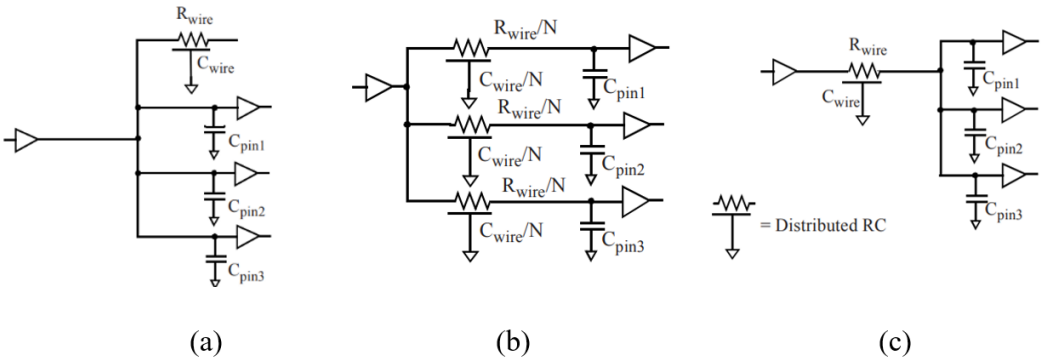


Figure 4.5 : RC tree representation used during pre-layout. (a) Best-case tree. (b) Balanced tree. (c) Worst-case tree.

4.4.2 Wireload Models

The wireload model can be specified using the following command :

set_wire_load_model "wlm_cons" -library "lib_stdcell"

Says to use the wireload model wlm_cons present in the

Cell library lib_stdcell.

In **top** wireload mode, all the nets within hierarchy inherit the wireload of the top-level.

In **enclosed** wireload mode, the wireload model of the block that fully encompasses the net is used for the entire net.

In **segmented** wireload mode, each segmented of the net gets its wireload model for the block that encompasses the net segment.

In Figure 4.6 shows various Wireload models.

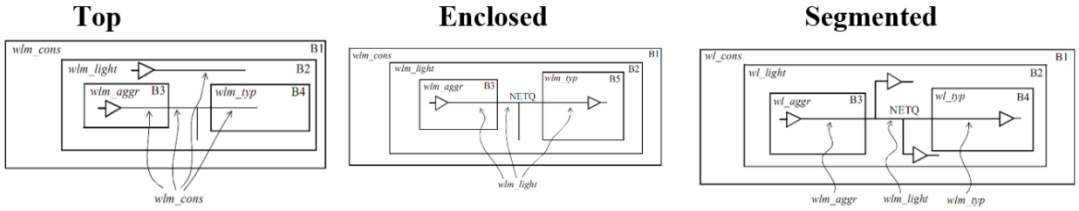


Figure 4.6: Wireload Models.

The wireload mode can be specified using the following command:

set_wire_load_mode top #Top wireload mode is set.

set_wire_load_mode enclosed #Enclosed wireload mode is set.

set_wire_load_mode segmented #Segmented wireload mode is set.

The wireload mode is selected based upon the chip area of the block.

A default wireload model may optionally be specified in the cell library as:

default_wire_load: "wlm_light**"**;

A wireload selection group which selects a wireload model based upon area is defined in the cell library.

wire_load_selection (WireAreaSelGrp){

wire_load_from_area (0, 50000, "wlm_light");

wire_load_from_area (50000, 100000, "wlm_cons");

wire_load_from_area (100000, 200000, "wlm_typ");

wire_load_from_area (200000, 500000, "wlm_aggr");

}

A cell library can contain many selection groups. We can select in STA by specifying in the command using

set_wire_load_selection_group WireAreaSelGrp.

4.4 Pre-layout Delay calculation.

The Delay of the cell (standard cell, macro, IO buffer or core) the values are extracted from table under various dependent conditions. The wireload model an approximate statistical delay value for net is obtained for pre-layout analysis. The pre-layout delay calculation is composed of as shown below :

i. Delay of the Net
ii. Delay calculation of Interconnect

4.4.1 Delay of the Net

The library description of each cell specifies the pin capacitance values for each of the input pins. Thus, the capacitive load at the net is equivalent addition of pin capacitance load of every fanout of the net and capacitance from

interconnect. In Section 2 we described about NLDM model for various timing arcs, in which delay entry function of input transition time and output capacitance. The output transition time of the logic cell is also described as a 2D table in terms of input transition and total output capacitance at the net.

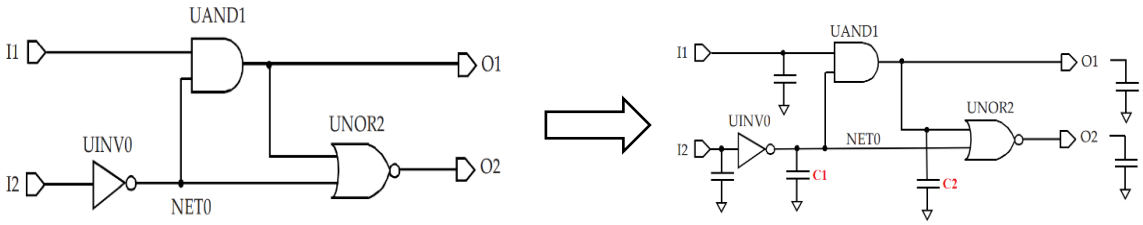


Figure 4.7: Logic block representation with capacitances.

In Figure 4.7 the total capacitance of NET0 is equivalent sum of C1, input capacitance of UAND1 and input capacitance of UNOR2.

4.4.2 Delay calculation of Interconnect

The interconnect parasitics are estimated using wireload models during STA analysis before optimization of logic and after optimization of the logic in design. The interconnect models used are best-case tree and worst-case tree. In best-case tree wireload model with resistance as zero and delay only because of purely capacitance. In worst-case tree wireload model with R and C value. This uses NLDM model for delay calculation.

4.5 Post layout Delay calculation.

In post-layout stage we extract parasitics from StarRC tool (Synopsys tool) of the net. We get the detailed description of parasitics of all nets present in design in .spef (SPEF file format). The .spef file is read by Primetime or ptshell of Synopsys tool and further we do simulation and analysis to check whether chip is working as per specified speed of operation or not.

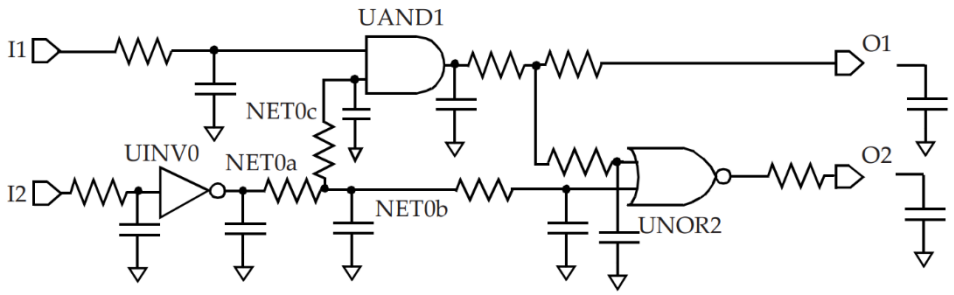


Figure 4.8: Logic block representation at post-layout stage.

The effective capacitance approach attempts to find a single capacitance that can be utilized as the equivalent load so that the original design as well the design with equivalent capacitance load behaves similarly in terms of timing at the output of the cell. This equivalent single capacitance is termed as effective capacitance.

The effective capacitance is a function of:

i. the driving cell, and

ii. the characteristics of the load or specifically the input impedance

of the load as seen from the driving cell.

The effective capacitance approximation is thus a good model for computing delay through the cell. The effective capacitance computation also provides the equivalent Thevenin source model which is then used to obtain the timing through the RC interconnect.

4.6 Path Delay Calculation.

The path delay calculated using timing graph. The timing through the combinational cell, Flip-flops and multiple paths are represented using timing arcs. The interconnect is represented with timing arcs from source to each destination point represented as a separate timing arc.

The entire design is annotated by timing arcs, computing the path delay involves adding up all the net and cell timing arcs along the path.

(a) Combinational Path Delay

In the Figure 4.9 we show the calculation of combinational path delay of a circuit. We need to consider both rising and fall edge paths for delay calculation.

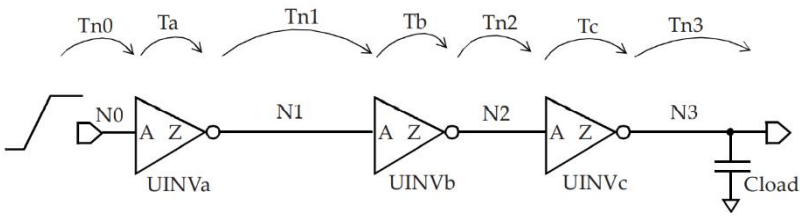


Figure 4.9: Timing a combinational path.

$$T_{fall} = Tn0_{rise} + Ta_{fall} + Tn1_{fall} + Tb_{rise} + Tn2_{rise} + Tc_{fall} + Tn3_{fall}.$$

$$T_{rise} = Tn0_{fall} + Ta_{rise} + Tn1_{rise} + Tb_{fall} + Tn2_{fall} + Tc_{rise} + Tn3_{rise}$$

(b) Flip-Flop paths

i. Input to Flip-flop path

In this we calculate the data path delay and capture clock path delay.

Data path is composed of nets, UNOR2 and UBUF cells.

Data path delay rise edge = $Tn1_{rise} + Ta_{fall} + Tn2_{fall} + Tbuf1_{fall} + Tn3_{fall} + Tb_{rise} + Tn4_{rise}$.

Data path delay rise edge = $Tn1_{rise} + Ta_{fall} + Tn2_{fall} + Tbuf1_{fall} + Tn3_{fall} + Tb_{rise} + Tn4_{rise}$.

Data path delay fall edge = $Tn1_{fall} + Ta_{rise} + Tn2_{rise} + Tbuf1_{rise} + Tn3_{rise} + Tb_{fall} + Tn4_{fall}$.

Capture clock path rise delay =$Tn5_{rise} + Tbuf2_{rise} + Tn6_{rise}$.

Capture clock path fall delay =$Tn5_{fall} + Tbuf2_{fall} + Tn6_{fall}$.

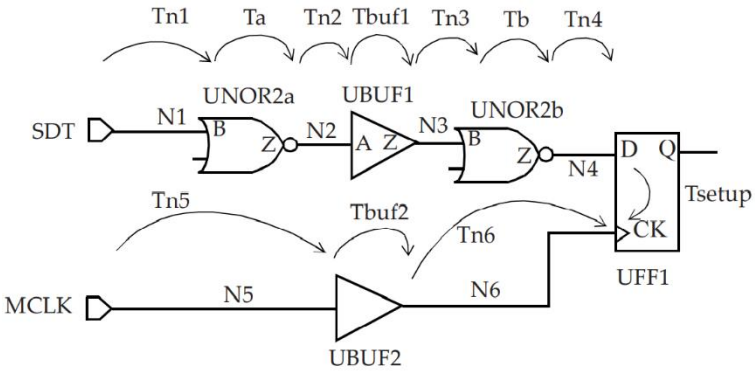


Figure 4.10: Timing Path Input to flip-flop.

ii. Flip-flop to Flip-flop path

The data path between two flip-flops and corresponding clock paths is shown in Figure 4.11. Data path delay consists of net N1, N2, N3 and cell UNANDa, UANDb. Launch clock path delay consists of net N4,N5 and cell UBUF5. Capture clock path delay consists of net N4,N5,N6 and cell UBUF6.

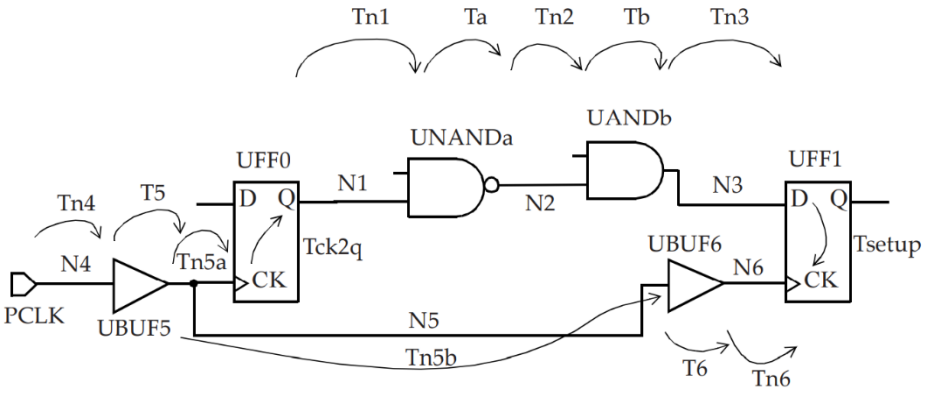


Figure 4.11: Flip-flop to flip-flop.

(c) Multiple paths

If multiple paths are there to arrive at a point. There two possible path longest path and shortest path. For worst path, we consider longest path which is termed as late path or max path. For slow path, we consider earliest path which is termed as early path or min path.

In Figure 4.12 we observe that at input of UFF4/D there are two path available.

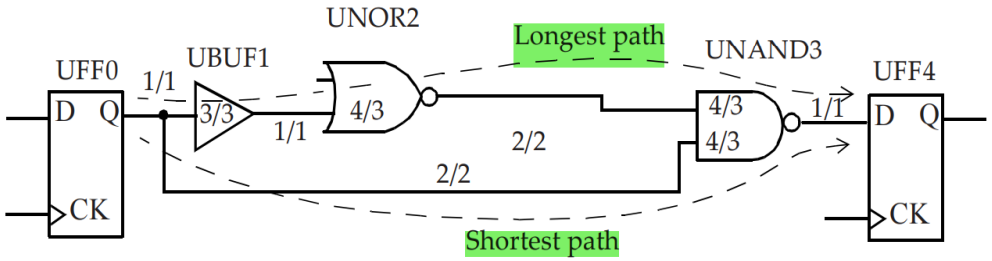


Figure 4.12: Longest and shortest path.

Slack Calculation

Setup Slack = Required time (RT) – Arrival time (AT).

Hold Slack = Arrival time (AT) – Required time (RT) .

Chapter 5

Timing Checks

In this section we describe the setup and hold check of the design. The timing checks are performed at multiple conditions including worst-case slow condition (WCS) for setup and best-case fast condition (BCF) for hold. The asynchronous timing check involve removal timing check and recovery timing check.

The Timing paths in a design can be considered as a collection of paths. Each path has a startpoint and an endpoint. The startpoint includes input ports and clock pins of FF's and memories. The endpoints include output ports and data input of FF's and memories.

The timing path can be:

i. from input port to an output part.

ii. from input port to an input of a flip-flop or memory.

iii. from the clock pin or a memory to an input of a flip-flop or memory.

iv. from the clock pin of a flip-flop or memory to an output port.

The timing paths are sorted into path groups by the clock associated with the endpoint of the path. The default path group that includes all non-clocked paths.

In Figure 5.1 we can observe the timing path groups for a design.

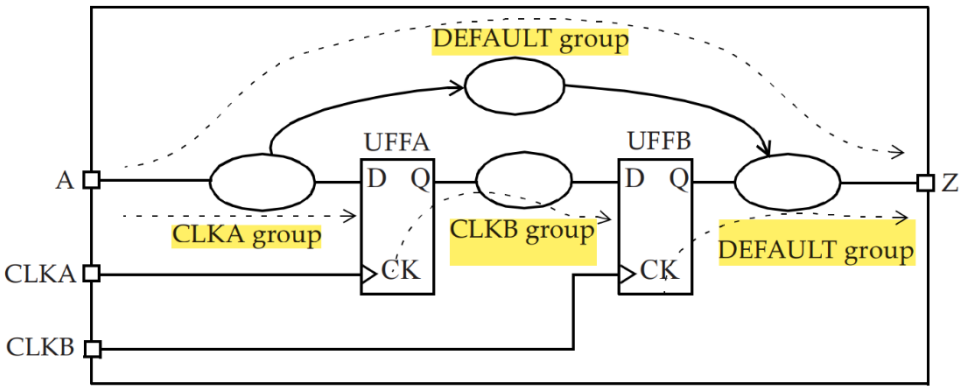


Figure 5.1: Timing path groups.

The Setup and Hold timing checks are performed on all four paths discussed in Timing path groups.

5.1 Understanding Timing Report

In the Figure 5.2 shows an example circuit for explaining the concept of setup timing check and hold timing check. The Figure 5.3 gives detailed explanation of reading a timing report.

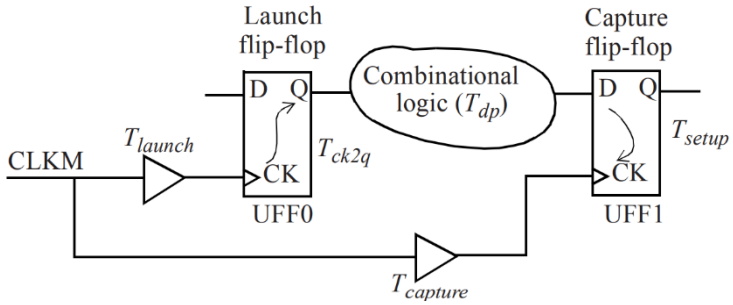


Figure 5.2: Data and Clock path of a circuit.

```
Startpoint: UFF0 (rising edge-triggered flip-flop clocked by CLKM)
Endpoint: UFF1 (rising edge-triggered flip-flop clocked by CLKM)
Path Group: CLKM
Path Type: max

Point                                    Incr       Path
-----------------------------------------------------------
clock CLKM (rise edge)                   0.00       0.00
clock network delay (ideal)              0.00       0.00
UFF0/CK (DFF )                           0.00       0.00 r
UFF0/Q (DFF ) <-                         0.16       0.16 f
UNOR0/ZN (NR2  )                         0.04       0.20 r
UBUF4/Z (BUFF  )                         0.05       0.26 r
UFF1/D (DFF )                            0.00       0.26 r
data arrival time                                   0.26

clock CLKM (rise edge)                  10.00      10.00
clock network delay (ideal)              0.00      10.00
clock uncertainty                       -0.30       9.70
UFF1/CK (DFF )                                      9.70 r
library setup time                      -0.04       9.66
data required time                                  9.66
-----------------------------------------------------------
data required time
data arrival time
-----------------------------------------------------------
slack (MET)
```

The Endpoint flip-flop has a clock CLKM. Thus path group is CLKM

The max path is for setup.

Clock Launch edge

Network Delay specifed using clock_latency command

The delay values are specified in the cell library. These are dependent on library (hvt,lvt,typical, temp, load, process node). The values varify from one cell to other cell.

AT = Tlaunch + T_SL + T_NL + Tnet (max_all_nets) + Tclk_q (max) + Tcomb(max)

Clock Capture edge

The uncertainty specified using set_clock_uncertainity command

RT = Tcapture + T_SL + T_NL + Tnet (min_in_capture_path) - Tsu(setup) - Tun (uncertainty)

RT - Required Time

AT - Arrival Time

Setup Slack = RT - AT

Figure 5.3: The setup analysis report given by the tool.

5.2 Setup timing check

A setup timing check verifies the timing relationship between the clock and the data pin of a flip-flop so that the setup requirement is met. The data should be stable for a certain amount of time, namely the setup time of the flip-flop, before the active edge of the clock arrives at the flip-flop. This requirement ensures that the data is captured reliably into the flip-flop.

In general, there is a launch flip-flop - the flip-flop that launches the data, and a capture flip-flop - the flip-flop that captures the data whose setup time must be satisfied. The setup check validates the long (or max) path from the launch flip-flop to the capture flip-flop. The first rising edge of clock CLKM appears at time T_{launch} at launch flip-flop. The data launched by this clock edge appears at time $T_{launch} + T_{ck2q} + T_{dp}$ at the D pin of the flip-flop UFF1. The second rising edge of the clock (setup is normally checked after one cycle) appears at time $T_{cycle} + T_{capture}$ at the clock pin of the capture flip-flop UFF1.

(a) Flip-flop to Flip-flop:

In the Figure 5.4 shows the flip-flop-to-flip-flop timing path waveforms.
The calculation is shown below.
Arrival Time (AT) = $T_{launch} + T_{clk\_q} + T_{comb} + T_{net}$.
Consider all max value for calculation for AT.

T_{launch} = The delay of the clock tree of the launch flip-flop. This includes net delay of path consists of Source delay, network delay, net delay of the path.
T_{clk_q} = The Launch flip-flop delay.
T_{comb} = The combinational delay of data path.
T_{net} = The Net delay of the data path.

Required Time (RT) = $T_{clk} + T_{capture} - T_{su} - T_{UN}$

T_{clk} = Clock Period.
$T_{capture}$ = The delay of the clock tree for capture flip-flop. This includes net delay of path consists of Source delay, network delay, net delay of the path.
T_{su} = The setup time of the capture flip-flop.
T_{UN} = The clock uncertainty time. It is combined effect of jitter and other extra delay.

Setup Slack = RT-AT.
If Slack = +ve value --------- Setup constraint satisfied.

If Slack = -ve value --------- Setup checks failed.

Skew = $T_{capture}$ - T_{launch}.

The Setup check always performed on longest or max timing path. It is verified at slow corner where the delays are the largest.

NOTE:

- The source latency does not affect paths that are internal to the design and have the same launch clock and capture clock. This is because the same latency gets added to both the launch clock path and the capture clock path.
- The latency does impact timing paths that go through the inputs and outputs of the design under analysis.

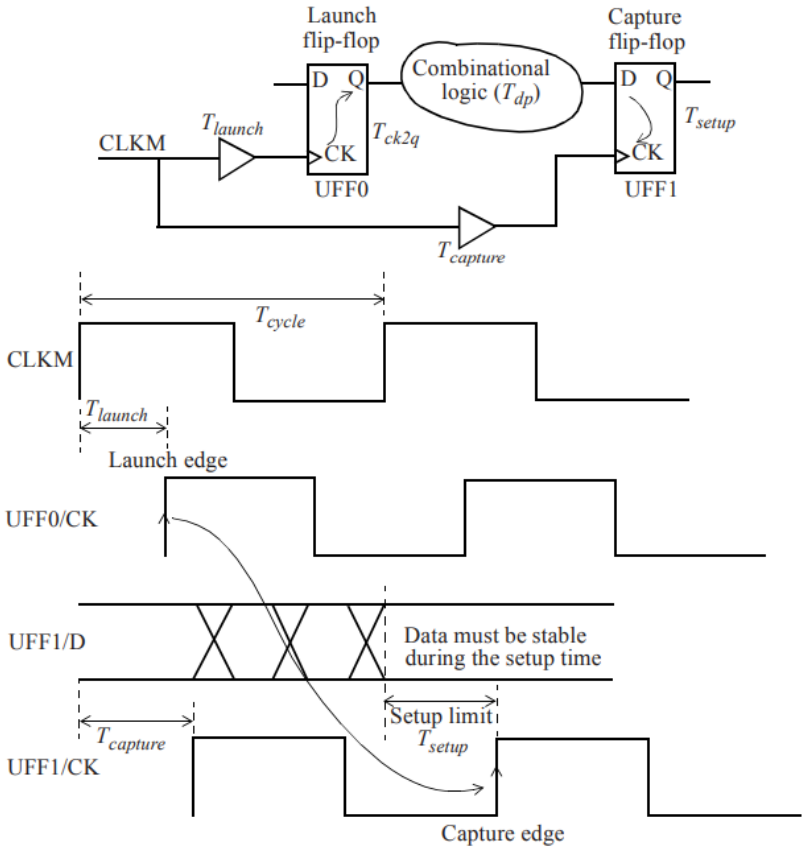


Figure 5.4: Data and clock signals for setup timing check.

(b) Input to Flip-flop

In this case the input port clocked by Virtual clock or by actual clock. The example is shown in Figure 5.5. On both the cases the calculation remains same just the Startpoint is changed.

The set_input_delay command is used to set input external delay.

AT = $T_{launch} + T_{input\_external\_delay} + T_{comb} + T_{net}$.

Consider all max value for calculation for AT.

$T_{input\_external\_delay}$ = This value is specified from set_input_delay command.

RT = $T_{clk} + T_{capture} - T_{su} - T_{UN}$.

Consider max value for setup check.

Consider all min value for calculation for RT.

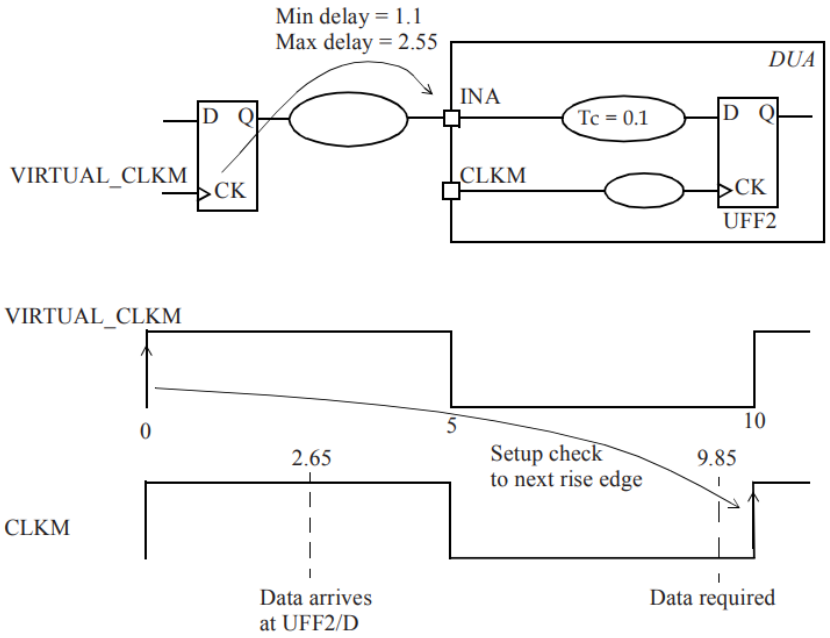


Figure 5.5: Setup check for the path through input port.

(c) Flip-flop to Output path.

The output port can be constrained w.r.t a virtual clock, Internal clock of a design, Input clock port or Output clock port. The Figure 5.6 gives an example for Flip-flop to output path.

The set_output_delay command is used to set output external delay.

AT = $T_{launch} + T_{clk\_q} + T_{comb} + T_{net}$.

Consider all max value for calculation for AT.

RT = $T_{clk} + T_{capture} - T_{output\_external\_delay} - T_{UN}$.

$T_{output\_external\_delay}$ ---- Consider max value for hold check.

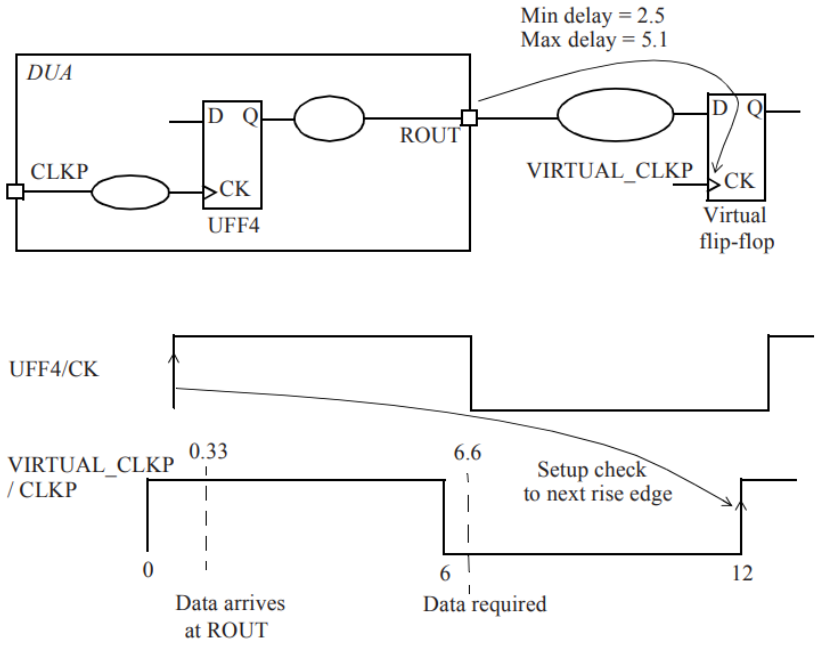


Figure 5.6: Setup check for the path through output port.

(d) Input to Output path

The design can have a combinational path going from an input port to an output port. This path is constrained using set_input_delay (input path) and set_output_delay (output path). In Figure 5.7 shows an example on Setup check from Input to Output path.

Virtual clocks are used to specify constraints on both input and output ports.

AT = $T_{launch} + T_{input\_external\_delay} + T_{comb} + T_{net}$.

Consider all max value for calculation for AT.

$T_{input\_external\_delay}$ ---- Consider max value for setup check.

RT = $T_{virtual\_clk} + T_{capture} - T_{output\_external\_delay} - T_{UN}$.

$T_{output\_external\_delay}$ ---- Consider max value for setup check.

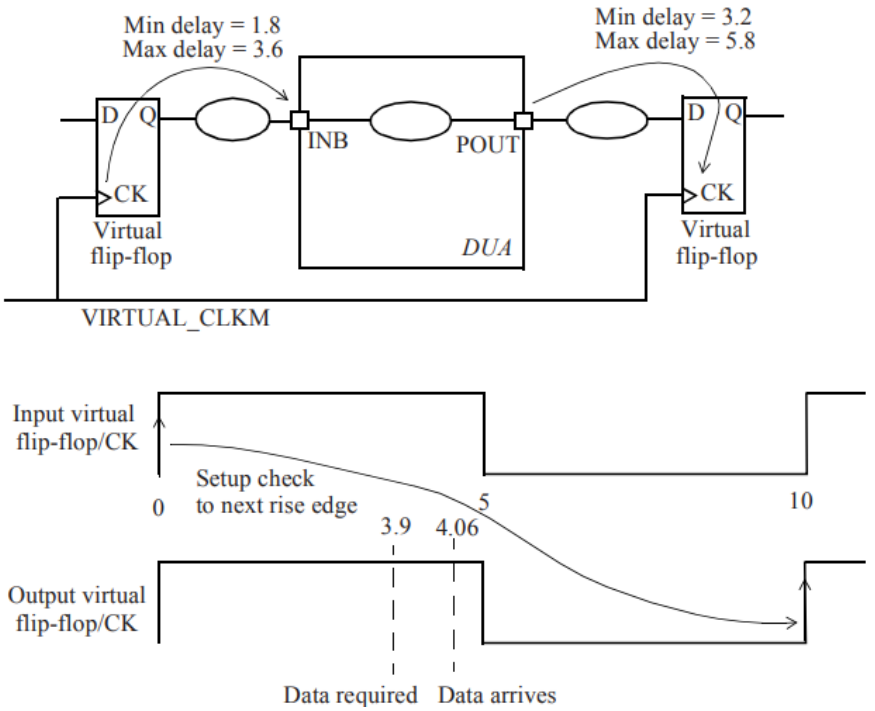


Figure 5.7: Combinational path from input to output path.

5.3 Hold timing check.

A hold timing check ensures that a flip-flop output value that is changing does not pass through to a capture flip-flop and overwrite its output before the flip-flop has had a chance to capture its original value. This check is based on the hold requirement of a flip-flop. The hold specification of a flip-flop requires that the data being latched should be held stable for a specified amount of time after the active edge of the clock. In Figure 5.8 shows hold requirement of a flip-flop.

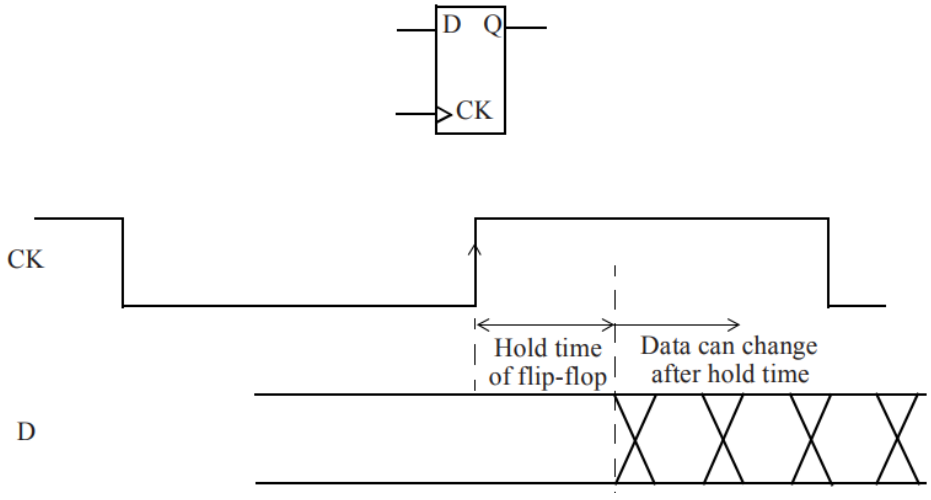


Figure 5.8: Hold requirement of a flip-flop.

Just like the setup check, a hold timing check is between the launch flip-flop - the flip-flop that launches the data, and the capture flip-flop – the flip-flop that captures the data and whose hold time must be satisfied. The clocks to these two flip-flops can be the same or can be different. The hold check is from one active edge of the clock in the launch flip-flop to the same clock edge at the capture flip-flop. Thus, a hold check is independent of the clock period. The hold check is carried out on each active edge of the clock of the capture flip-flop.

(a) Flip-flop to Flip-flop

AT = $T_{launch} + T_{clk\_q} + T_{comb} + T_{net}$.
Consider all min value for calculation for AT.

RT = ~~T_{clk}~~ + $T_{capture} + T_{hold} + T_{UN}$.
Consider all max value for calculation for RT.

NOTE :
No Clock period for hold check. The launch edge and capture edge both are at the same edge.

Hold Slack = AT – RT -------→ My Method

In tool it is Slack = RT – AT.
If Slack = -ve value --------- Hold constraint satisfied.
If Slack = +ve value -------- Hold checks failed.

The Hold check always performed on shortest or min timing path. It is verified at fast corner where the delays are the smallest.

The commands used in this are

i. create_clock.
ii. set_clock_uncertainty.
iii. set_clock_latency.
iv. set_input_transition.

In Figure 5.9 shows data and clock signals for hold timing check.

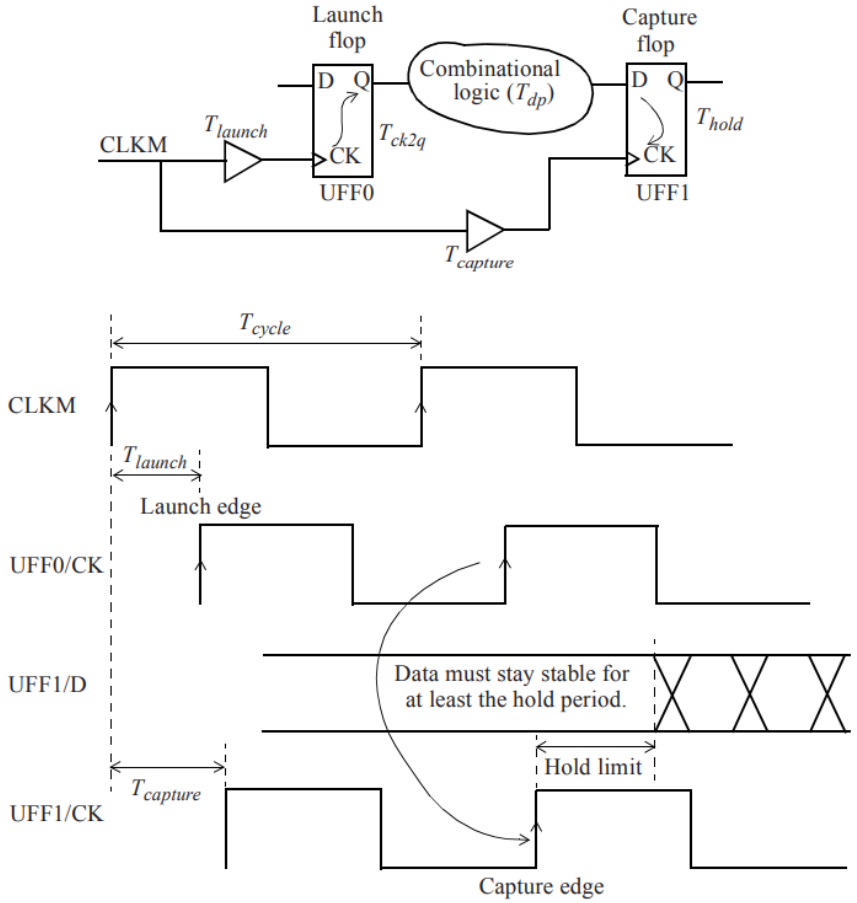


Figure 5.9: Data and clock signals for hold timing check.

(b) Input to flip-flop

AT = $T_{launch} + T_{input\_external\_delay} + T_{comb} + T_{net}$.

Consider all min value for calculation for AT.

$T_{input\_external\_delay}$ ---- Consider min value for hold check.

RT = ~~T_{clk}~~ + $T_{capture} + T_{hold} + T_{UN}$.

Consider all max value for calculation for RT.

In Figure 5.10 shows a hold report for input to flip-flop path.

```
Path Group: CLKM
Path Type: min

Point                                          Incr       Path
-----------------------------------------------------------------
clock VIRTUAL_CLKM (rise edge)                 0.00       0.00
clock network delay (ideal)                    0.00       0.00
input external delay                           1.10       1.10 f
INA (in) <-                                    0.00       1.10 f
UINV1/ZN (INV  )                               0.02       1.13 r
UAND0/Z (AN2  )                                0.06       1.18 r
UINV2/ZN (INV  )                               0.02       1.20 f
UFF2/D (DFF )                                  0.00       1.20 f
data arrival time                                         1.20

clock CLKM (rise edge)                         0.00       0.00
clock source latency                           0.00       0.00
CLKM (in)                                      0.00       0.00 r
UCKBUF0/C (CKB  )                              0.06       0.06 r
UCKBUF2/C (CKB  )                              0.07       0.12 r
UCKBUF3/C (CKB  )                              0.06       0.18 r
UFF2/CK (DFF )                                 0.00       0.18 r
clock uncertainty                              0.05       0.23
library hold time                              0.01       0.25
data required time                                        0.25
-----------------------------------------------------------------
data required time                                        0.25
data arrival time                                        -1.20
-----------------------------------------------------------------
slack (MET)                                               0.95
```

Figure 5.10: Hold timing report for Input to Flip-flop path.

(c) Flip-flop to Output path

AT = $T_{launch} + T_{clk\_q} + T_{comb} + T_{net}$.

Consider all max value for calculation for AT.

RT = $T_{clk} + T_{capture} - T_{output\_external\_delay} + T_{UN}$.

$T_{output\_external\_delay}$ ---- Consider min value for hold check.

In Figure 5.11 shows Hold timing waveform requirement for path through output port.

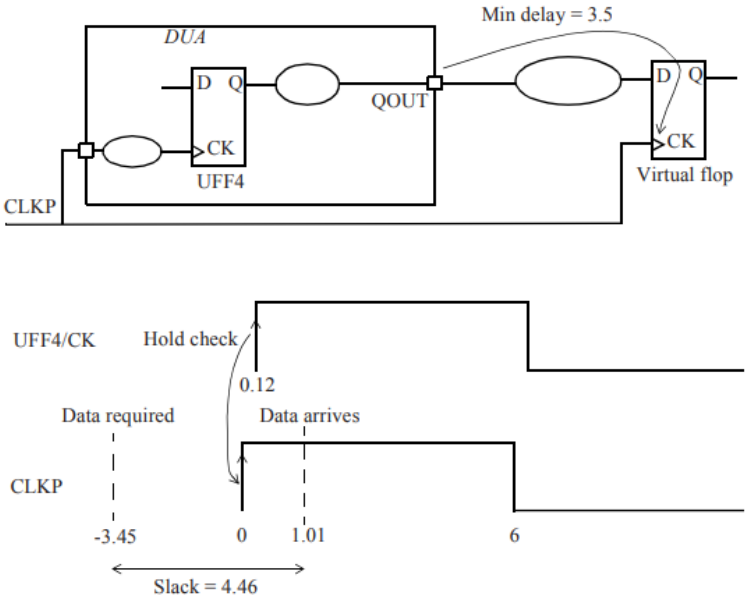


Figure 5.11: Path through output port.

(d) Input to Output path

AT = $T_{launch} + T_{input\_external\_delay} + T_{comb} + T_{net}$.

Consider all max value for calculation for AT.

$T_{input\_external\_delay}$ ---- Consider min value for hold check.

RT = $T_{virtual\_clk} + T_{capture} - T_{output\_external\_delay} + T_{UN}$.

$T_{output\_external\_delay}$ ---- Consider min value for hold check.

The hold timing check on an input to output path show in Figure 5.12.

```
Startpoint: INB (input port clocked by VIRTUAL_CLKM)
Endpoint: POUT (output port clocked by VIRTUAL_CLKM)
Path Group: VIRTUAL_CLKM
Path Type: min

Point                                    Incr       Path
-------------------------------------------------------------
clock VIRTUAL_CLKM (rise edge)           0.00       0.00
clock network delay (ideal)              0.00       0.00
input external delay                     1.80       1.80 r
INB (in) <-                              0.00       1.80 r
UBUF0/Z (BUFF  )                         0.04       1.84 r
UBUF1/Z (BUFF  )                         0.06       1.90 r
UINV3/ZN (INV  )                         0.22       2.12 f
POUT (out)                               0.00       2.12 f
data arrival time                                   2.12

clock VIRTUAL_CLKM (rise edge)           0.00       0.00
clock network delay (ideal)              0.00       0.00
clock uncertainty                        0.05       0.05
output external delay                   -3.20      -3.15
data required time                                  -3.15
-------------------------------------------------------------
data required time                                  -3.15
data arrival time                                   -2.12
-------------------------------------------------------------
slack (MET)                                          5.27
```

Figure 5.12: Hold timing report on an input to output path.

5.4 Removal Timing Check

A removal timing check ensures that there is adequate time between an active clock edge and the release of an asynchronous control signal. The check ensures that the active clock edge has no effect because the asynchronous control signal remains active until removal time after the active clock edge. This check is based on the removal time specified for the asynchronous pin of the flip-flop. Like a hold check, it is a min path check except that it is on an asynchronous pin of a flip-flop. The removal timing check is shown in Figure 5.13.

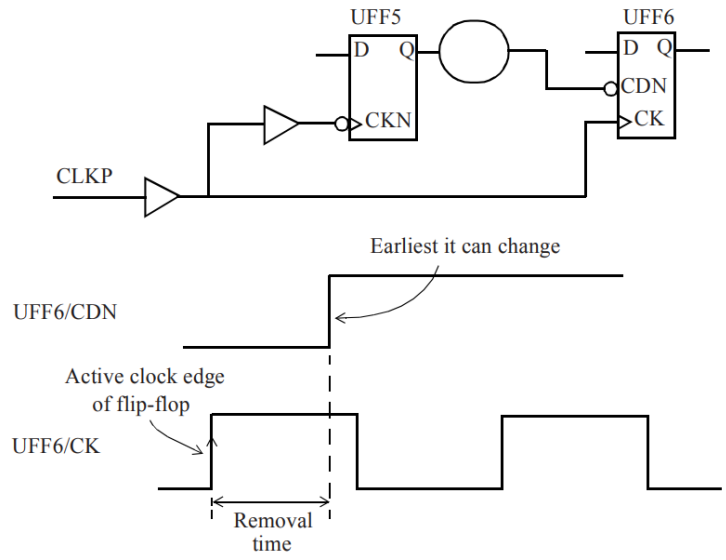


Figure 5.13: Removal timing check.

5.5 Recovery Timing Check

A recovery timing check ensures that there is a minimum amount of time between the asynchronous signal becoming inactive and the next active clock edge. This check ensures that after the asynchronous signal becomes inactive, there is adequate time to recover so that the next active clock edge can be effective. The recovery check is illustrated in Figure 5.14.

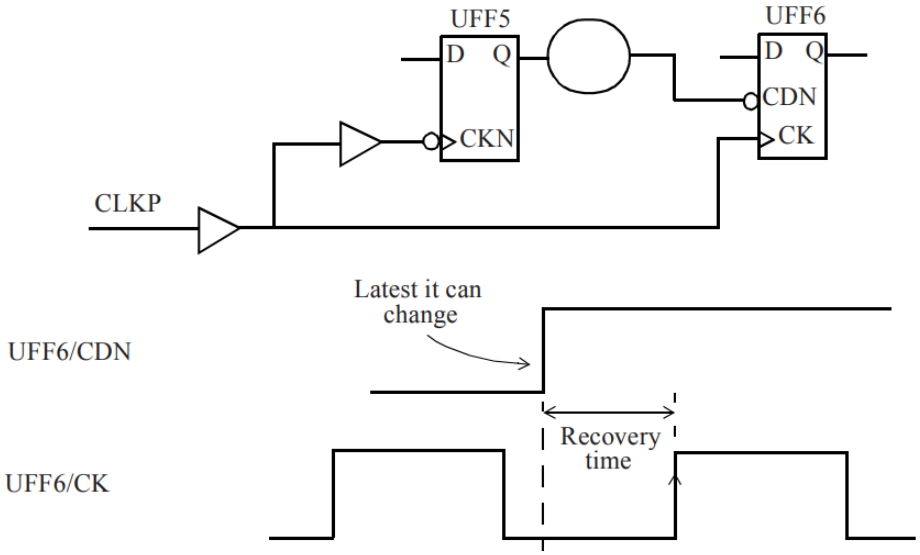


Figure 5.14: Recovery Timing check.

5.6 Multicycle Paths

The combinational datapath between Flip-flop can take more than one cycle to propagate through the logic. The following command are used for setup and hold timing check when it is multicycle paths.

Setup check :

set_multicycle_path #Multiplier_Value **-setup** {-start or -end} **-from** [<Startpoint>] **-through** [<Through point>] **-to** [<endpoint>] .

Hold check :

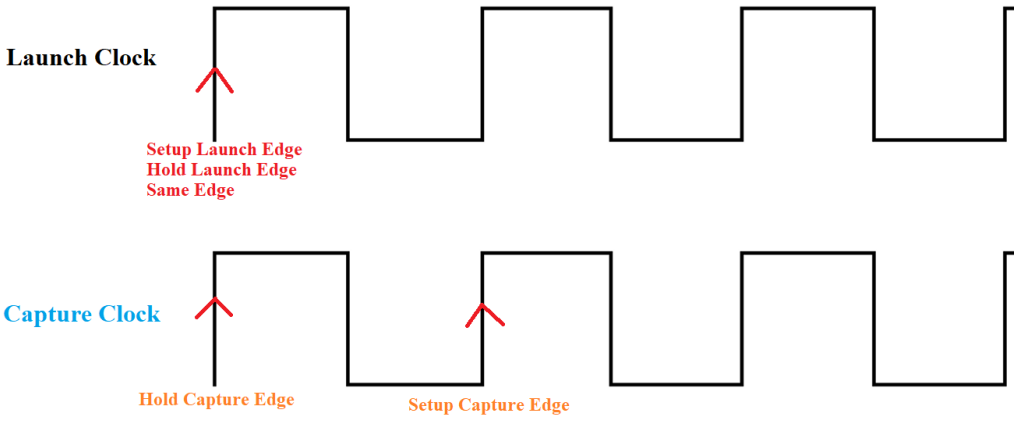
set_multicycle_path #Multiplier_Value **-hold** {-start or -end} **-from** [<Startpoint>] **-through** [<Through point>] **-to** [<endpoint>].

The command description is as follows :

- The **-end** option implies that we want to move the endpoint (or capture edge) back by the specified number of cycles, which is that of the capture clock.
- The **-start** option, specifies the number of launch clock cycles to move.
- The -end is the default for a multicycle setup and the -start is the default for multicycle hold.
- In Multicycle, the max path (setup) requires **N** clock cycles, the min path (hold) constraint to be **N-1** clock cycles.

The Basic understanding of Setup and Hold check for Launch and capture clock edge are described in the Figure 5.15 and 5.16.

In Figure 5.14 we see the launch edge for both setup and hold are at same edge. The capture edge for hold and launch edge for setup is at same edge. The capture edge



is next clock edge w.r.t launch edge.

Figure 5.15: Launch edge and capture edge for case 1.

In Figure 5.15 we see the Capture edge for both setup and hold are at same edge. The capture edge for hold and launch edge for setup is at same edge. The launch edge is one clock edge before w.r.t capture edge.

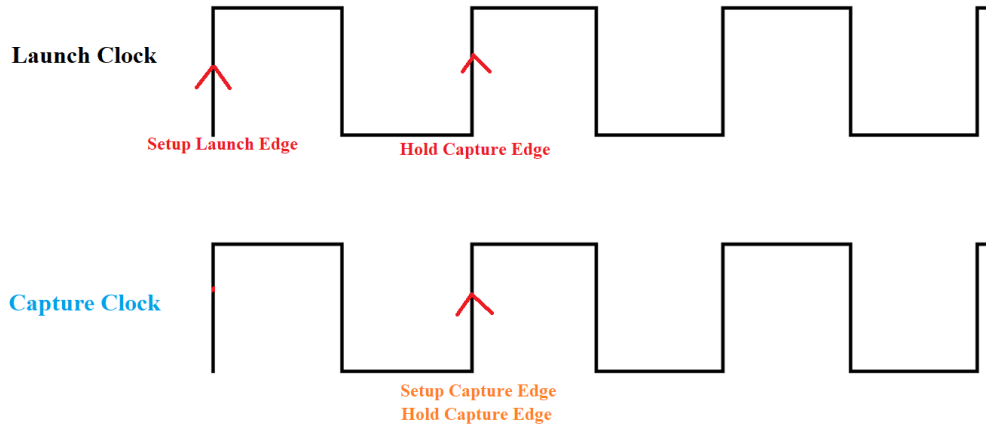


Figure 5.16: Launch edge and capture edge for case 2.

In Figure 5.17 we observe the setup multiplier is set to 3. Now the setup capture edge is 3rd clock cycle or 3rd edge. Due, to this now hold capture edge is one clock cycle before setup capture edge but, the hold launch edge is at 0th edge thus we set multiplier of 2 for hold edge and make hold capture edge and hold launch edge at same edge.

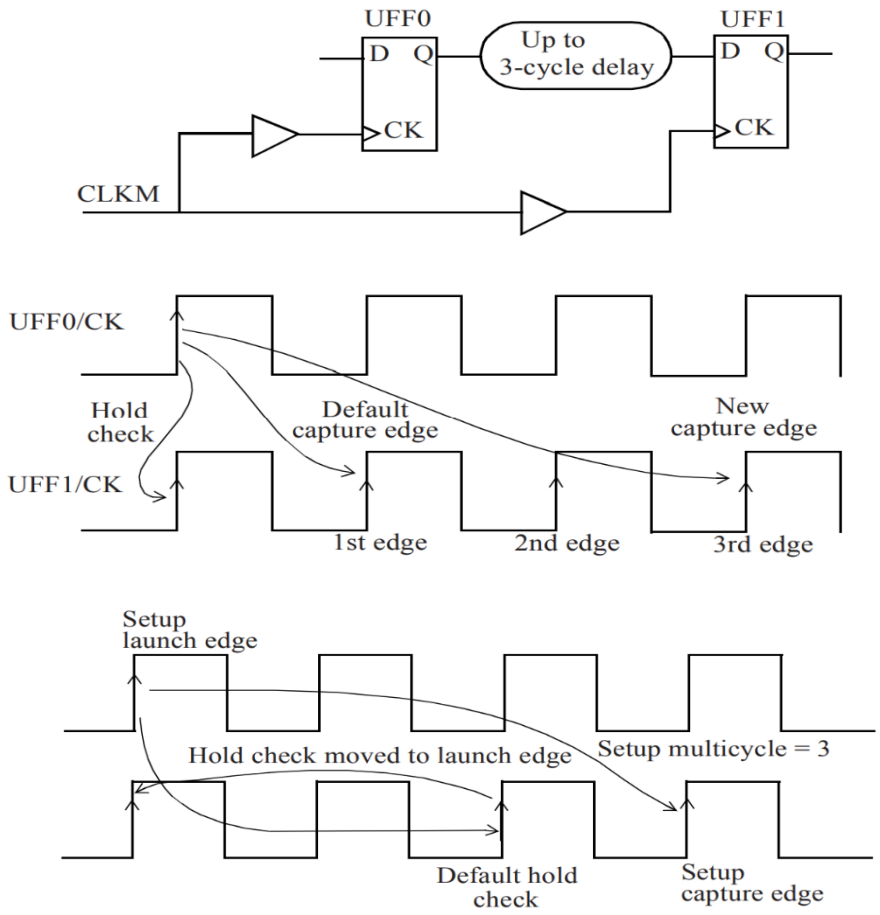


Figure 5.17: Understanding of Multicycle Path

The commands used for above example are:

create_clock -name CLKM -period 10 [get_ports CLKM].

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

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

In Figure 5.18 we get Capture clock edges for various hold and setup multicycle multiplier settings. The hold and setup multiplier are the values used based on constraints and same value is specified in the command set_multicycle_path

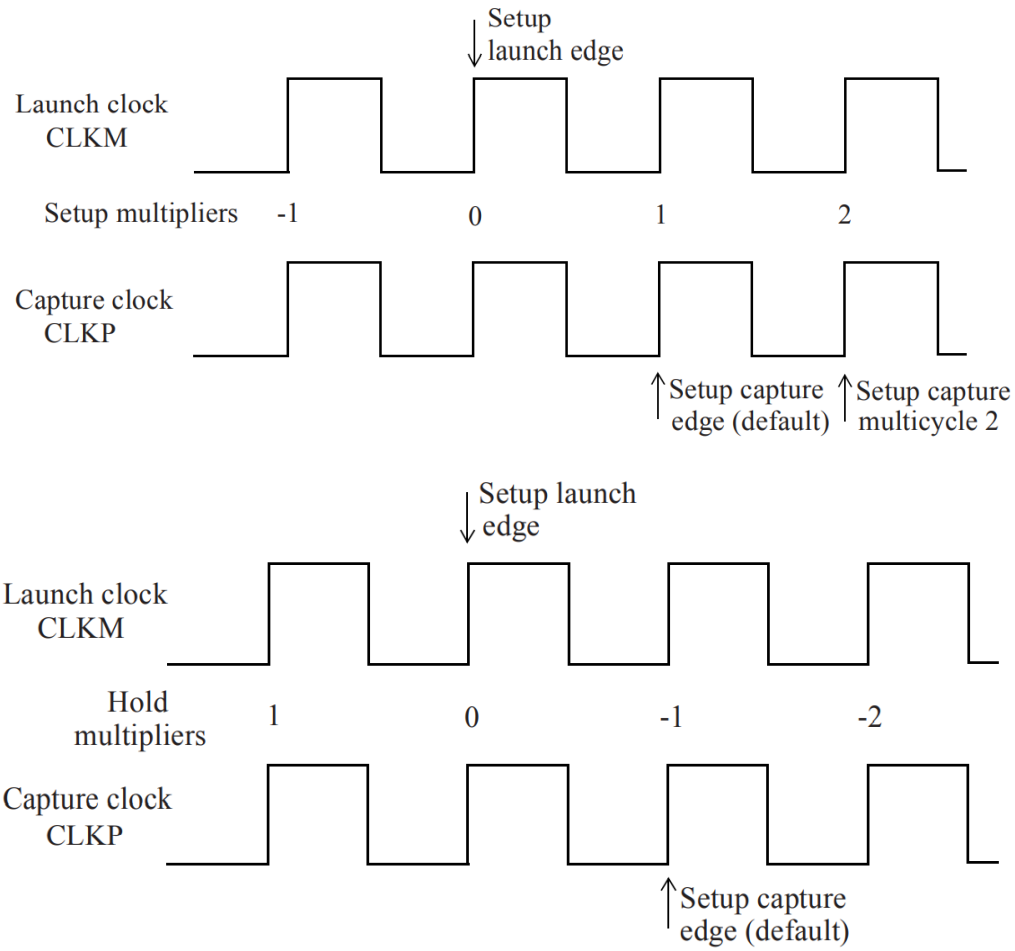


Figure 5.18: Capture clock edges for various hold and setup multicycle multiplier settings.

5.7 False Path

The timing paths are not real. The false path must be ignored by STA analysis.

The example for false path is :

- **i.** The path from one clock domain to another clock domain.
- **ii.** Clock pin of FF to input of another FF, through a pin of cell through the pins of multiple cells.

It is mandatory to specify the False path to reduce the analysis space is reduced.

A false path is set using the set_false_path specification. Some examples of specifying false path using the command are given below:

set_false_path -from [get_clocks SCAN_CLK] -to [get_clocks CORE_CLK]

Any path starting from the SCAN_CLK domain to the

CORE_CLK domain is a false path.

set_false_path -through [get_pins UMUX0/S]

Any path going through this pin is false.

NOTE :

- If a signal is sampled at a known or predictable time, no matter how far out, a multicycle path specification should be used so that the path has some constraint and gets optimized to meet the multicycle constraint.
- The false path is used on a path that is sampled many clock cycles later.

Chapter 6

Signal Integrity

In this section we will discuss on Signal Integrity concepts in nanometer technologies. The crosstalk noise refers to unintentional coupling of signals between two or more signal traces.

Noise refers to unintentional or undesired effects affecting the functionality of design or affect timing of devices. The reasons why the noise plays an important role in deep submicron technologies:

(a) Increasing number of metal layers.
(b) Vertically dominant metal aspect ratio means the trace is tall and thin which leads to increase in wire capacitance. The majority of capacitance comprised of sidewall coupling capacitance which maps into coupling capacitance between neighbouring wires.
(c) Higher routing density due to finer geometry.
(d) Larger number of interacting devices and interconnects.
(e) Faster waveform due to higher frequencies.
(f) Lower supply voltage.

The crosstalk noise is caused by the capacitive coupling between neighbouring signals on the die. The coupled nets can affect each other, and net can be categorized as **victim** and **aggressor**. The affected signal is victim, and the affecting signals is aggressor as shown in Figure 6.1. The C_{c1} and C_{c4} are coupling capacitance which couples signal of aggressor to victim.

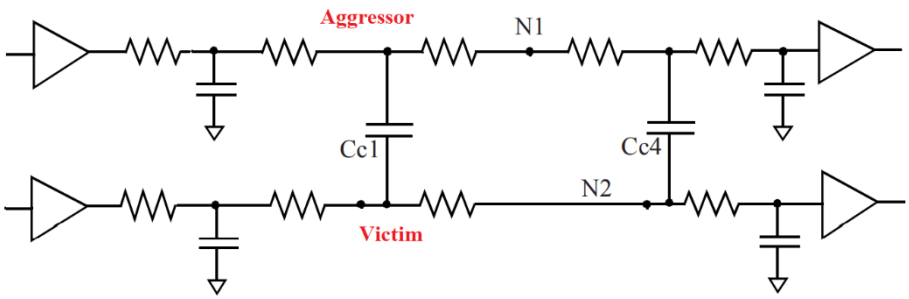


Figure 6.1: Coupled Interconnect.

The effects of crosstalk are:

i. Glitch.
ii. Delta delay.

6.1 Crosstalk Glitch Analysis

The glitch occurs on the stable signal (victim) due to charge transfer of aggressor switching through the coupling capacitance. The glitch can be positive or negative.

The magnitude of glitch depends on following factors:

i. Coupling capacitance between the aggressor net and victim net: If the coupling capacitance is large then magnitude of glitch is more.

$$\text{Glitch } \alpha\ C_{coupling}$$

ii. Slew of the aggressor net: The faster the slew at the aggressor net, the larger the magnitude of glitch.

Glitch α Slew rate at aggressor output.

iii. The smaller the grounded capacitance on the victim net, the larger the magnitude of the glitch.
iv. The smaller output drive strength of the cell driving the victim net, the larger the magnitude of the glitch.

Effects of Glitch on the functionality of the design:

i. The glitch magnitude if it is large enough to alter the logic value at the fanout cells. If the glitch occurs on the clock or asynchronous set or reset may lead to wrong functionality of the design. Even if glitch occur on the datapath at the input of sequential logic then cause incorrect data to be latched.
ii. Sometimes, a wide glitch may be propagated through the cells of the victim net and reach a sequential cell with catastrophic consequences for the design.

There two types of glitches:

i. Rise and Fall Glitch.
ii. Overshoot and Undershoot Glitch.

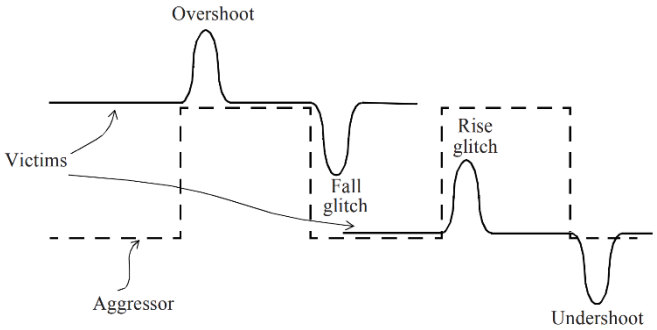


Figure 6.2: Types of Glitches.

6.2 Glitch threshold and propagation

Based upon glitch height and glitch width, a glitch can propagate through the fanout cell. The analysis is based upon the DC or AC noise thresholds.

The DC noise analysis examines the glitch magnitude.

The AC noise analysis examines glitch width and fanout cell output load.

(a) DC Threshold

- The DC noise margin is a check used for glitch magnitude and refers to the DC noise limits on the input of a cell while ensuring proper logic functionality.
- The DC noise margin limits are applications for every input pin of a cell. In general, the DC margin limits are separate for rise_glitch (input low) and fall_glitch (input high).
- Thus, a conservative glitch analysis checks that the peak voltage level (for all glitches) meets the VIL and the VIH levels of the fanout cells.
- Not all glitches with magnitude larger than the DC noise margin can change the output of a cell. The width of the glitch is also an important consideration in determining whether the glitch will propagate to the output or not.
- A narrow glitch at a cell input will normally not cause any impact at the output of a cell

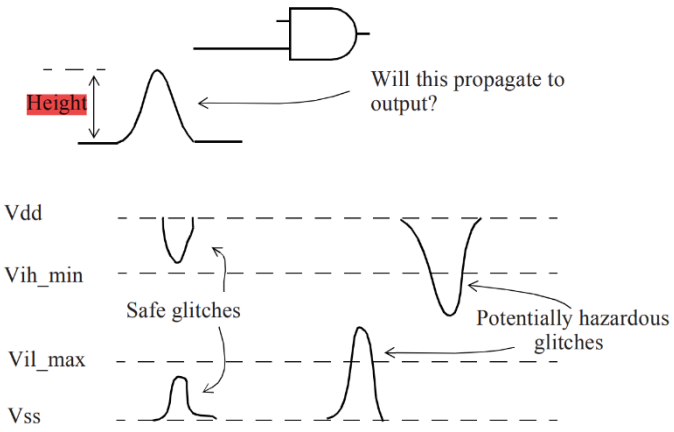


Figure 6.3: Glitch check based upon DC noise margin.

(b) AC Threshold

The AC analysis includes to verify the impact of glitches w.r.t the glitch width and the output load of the cell.

If the glitch is narrow or if fanout cell has a large output capacitance, the glitch would have no effect on the proper functional operation. If we increase the load at the output, makes the cell more immune to noise propagating from the input to the output.

The glitches below the AC threshold can be ignored or the fanout cell can be considered to be immune from such a glitch. The AC threshold region depends upon the output load and glitch width.

If the glitches are larger than AC threshold, the glitch at the cell input produces another glitch at the output of the cell. The output glitch height and width are a function of input glitch width and height as well as the output load.

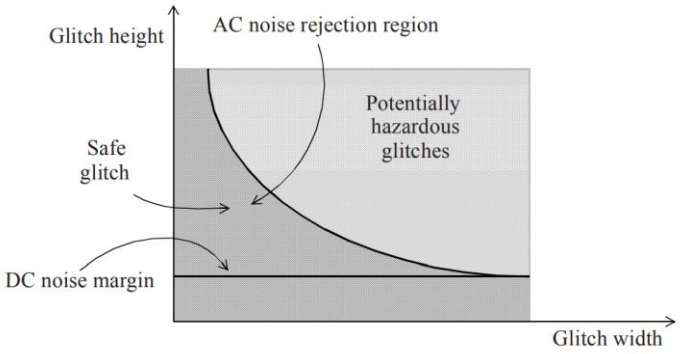


Figure 6.4: AC Nosie rejection region.

(c) Noise Accumulation with Multiple Aggressors

When multiple nets switch concurrently, the crosstalk coupling noise effect on the victim is compounded due to multiple aggressors. Then, add the glitch effect due to each aggressor and compute the cumulative effect on the victim. This also indicate the worst-case glitch on the victim. Another approach is use of RMS (Root-mean-squared) approach.

(d) Aggressor Timing Correlation

The crosstalk glitch analysis due to multiple aggressors, we include the timing correlation of the aggressor nets and determine whether the multiple aggressors can switch concurrently. The STA obtains this information from the timing windows of the aggressor nets. The timing windows during which a net may switch within a clock cycle. The switching windows (rising and falling) provide the necessary information on whether the aggressor nets can switch together.

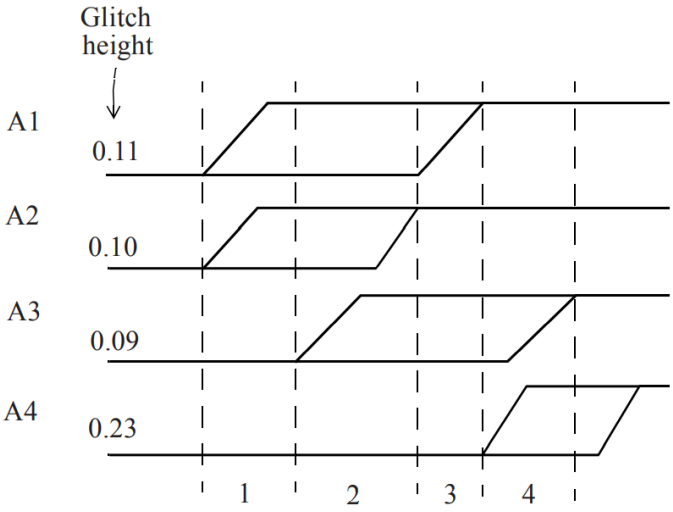


Figure 6.5: Switching windows and glitch magnitudes from multiple aggressors.

(e) Aggressor Functional Correlation

The functional correlation between various signals. For example, the scan control signals only switch during scan mode and are steady during functional mode of the design. Thus, the scan control signals cannot cause a glitch on other signals during functional mode. The scan control signals can only be aggressors during the scan mode.

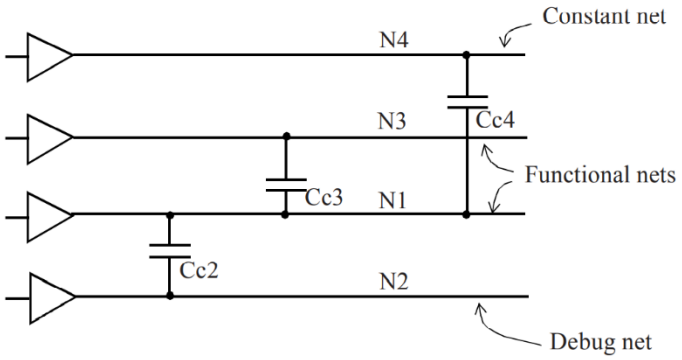


Figure 6.6: Three coupling capacitors and one aggressor net.

In Figure 6.6 the net N1 having coupled with three other nets N2, N3 and N4. The net N4 is a constant and cannot be aggressor on net N1 even though there is coupling capacitance between net N1 and N4. The net N3 carries functional data, only net N3 can be considered as a potential aggressor for net N1.

6.3 Crosstalk Delay Analysis

The capacitance extracted includes ground capacitance of the net and inter-signal capacitance. The total net capacitance is considered using the basic delay calculation. When the neighbouring nets are steady, the inter-signal capacitances can be treated as grounded. When a neighbouring net is switching, the charging current through the coupling capacitance impacts the timing of the net.

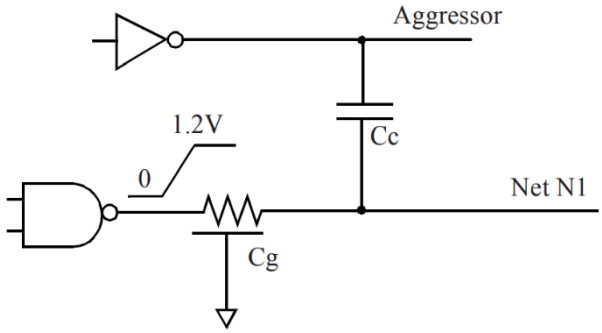


Figure 6.7: The ground capacitance and coupling capacitance impacting the crosstalk.

The Capacitive charging under various scenarios:

(a) Aggressor net steady

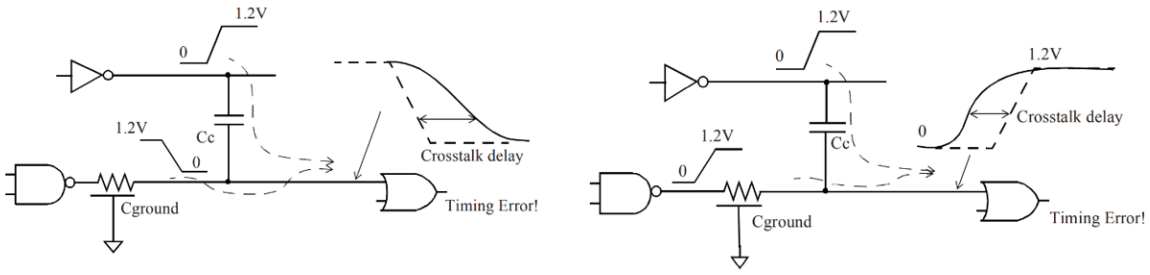
The net N1 has charge for Cg and Cc charged by the driving cell to Vdd. The delay calculation is normal way and does include crosstalk effect from aggressor net.

(b) Aggressor switching in same direction.

If the aggressor switching in the same direction results in a smaller delay for the switching net, this reduction in delay is labelled as negative crosstalk delay. This is considered for min path analysis.

(c) Aggressor switching in opposite direction.

If the aggressor switching in the opposite direction results in a larger delay for the switching net, this reduction in delay is labelled as positive crosstalk delay. This is considered for max path analysis.



(a) (b)

Figure 6.8: (a) Positive crosstalk. (b) Negative crosstalk.

6.4 Aggressor Victim relation

The relation of aggressor affecting victim is explained below :

(a) Accumulation with Multiple Aggressors

When multiple nets switch concurrently, the crosstalk delay effect on the victim gets compounded due to multiple aggressors. Most analyses for coupling due to multiple aggressors add the incremental contribution from each aggressor and compute the cumulative effect on the victim. The analysis of multiple aggressors for crosstalk glitch analysis, contributions can also be added using root-mean-squared (RMS) which is less pessimistic than the straight sum of individual contributions.

(b) Aggressor Victim Timing Correlation

The timing windows represent the earliest and the latest switching times during which a net may switch within a clock cycle. If the timing windows of the aggressor and the victim overlap, the crosstalk effect on delay is computed.
In Figure 6.9 the Bin 1 has A1 and A2 switching which can result in crosstalk delay impact of 0.26 (= 0.12 + 0.14). Bin 2 has A1 switching which can result in crosstalk delay impact of 0.14. Bin 3 has A3 switching which can result in crosstalk delay impact of 0.23. Thus, bin 1 has the worst possible crosstalk delay impact of 0.26.

The four types of crosstalk delays are positive rise delay (rise edge moves forward in time), negative rise delay (rise edge moves backward in time), positive fall delay and negative fall delay.

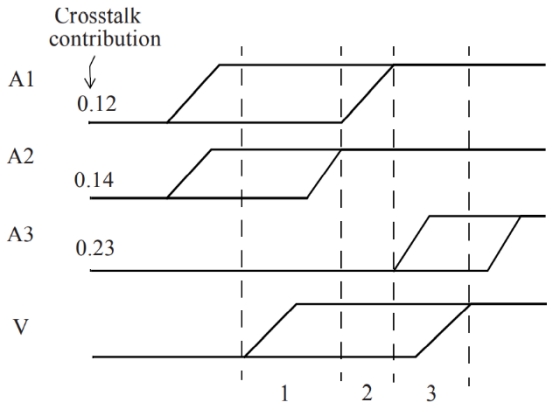


Figure 6.9: Timing Windows and crosstalk contributions from various aggressors.

(c) Aggressor Victim Functional Correlation

The crosstalk delay calculation can consider the functional correlation between various signals. For example, the scan control signals only switch during scan mode and are steady during functional mode of the design. Thus, the scan control signals cannot cause a glitch on other signals during functional mode. The scan control signals can only be aggressors during the scan mode.

6.5 Timing Verification using the Crosstalk Delay

The following four types of crosstalk delay contributions are computed for every cell and interconnect in the design:

i. Positive rise delay (rise edge moves forward in time).
ii. Negative rise delay (rise edge moves backward in time).
iii. Positive fall delay (fall edge moves forward in time).
iv. Negative fall delay (fall edge moves backward in time).

The crosstalk delay contributions are then utilized during timing analysis for the verification of the max and min paths (setup and hold checks).

(a) Setup Analysis

The STA with crosstalk analysis verifies the design with the worst-case crosstalk delays for the data path and the clock paths. The worst condition for

setup check is when both the launch clock path and the data path have positive crosstalk and the capture clock path has negative crosstalk.

Based upon above description, the setup (or max path) analysis assumes that:

- Launch clock path sees positive crosstalk delay so that the data is launched late.
- Data path sees positive crosstalk delay so that it takes longer for the data to reach the destination.
- Capture clock path sees negative crosstalk delay so that the data is captured by the capture flip-flop early.

(b) Hold Analysis

The worst condition for hold check is when both the launch clock path and the data path have negative crosstalk and the capture clock path has positive crosstalk.

Based upon above description, the hold (or min path) analysis assumes that:

- Launch clock path sees negative crosstalk delay so that the data is launched late.
- Data path sees negative crosstalk delay so that it takes longer for the data to reach the destination.
- Capture clock path sees positive crosstalk delay so that the data is captured by the capture flip-flop early.

6.6 Noise Avoidance Techniques

The impact and analysis of crosstalk effects:

i. Shielding:

The shield wires are placed on either side of the critical signals. The shields are connected to power or ground rails. The shielding of critical signals ensures no active aggressors for the critical signals since the nearest neighbour in the same metal layer are shield trace at a fixed potential.

ii. Wire spacing:

The wire spacing reduces the coupling capacitance to the neighbouring nets.

iii. Fast slew rate:

The fast slew rate on the net is less susceptible to crosstalk and is inherently immune to crosstalk effects.

iv. Maintain good stable supply:

The Minimizing jitter due to power supply variations. Therefore, adequate decoupling capacitance should be added to minimize noise on the power supply.

v. Guard ring: A guard ring (or double guard ring) in the substrate helps in shielding the critical analog circuitry from digital noise

vi. Deep n-well: This is similar to the above as having deep n-well for the analog portions helps prevent noise from coupling to the digital portions.

vii. Isolating a block: In a hierarchical design flow, routing halos can be added to the boundary of the blocks; furthermore, isolation buffers could be added to each of the IO of the block.

Chapter 7

Duty Cycle Distortion

IN this section we will discuss on DCD, aging of transistors and its effects on the chip. Synchronous circuits dominate the electronic world because clocking eases the design of circuits compared to asynchronous circuits. At the same time, clocking also introduces its share of challenges to overcome. In terms of accuracy, SPICE simulations have always been held as the gold standard. But SPICE simulations are compute-time intensive and typically run on just small portions of a design. The gate level simulation worked well for designs which use technology node 90nm or larger. As process nodes advanced and design size and complexity started growing, gate level simulation as a signoff tool started getting strained. Today's advanced-process-based designs are facing chip-signoff challenges due to limitations of STA and duty cycle distortions (DCD).

7.1 Duty Cycle Distortion (DCD)

The quantitative specification of the DCD is the ratio of the duration of an occurrence of a single logic-1 to the sum of the duration of an occurrence of a single logic-1and duration of an occurrence of a single logic-0. The Duty Cycle (DC) is also defined as ratio of the pulse duration to the pulse period.

$$DC = \frac{T_{on}}{T}$$

The Figure 7.1 shows the Duty cycle definition of a clock and Duty cycle degradation. For example, a clock signal of 2ns with DC as 50% means, 1ns is time duration for high pulse and other 1ns is time duration for low pulse.

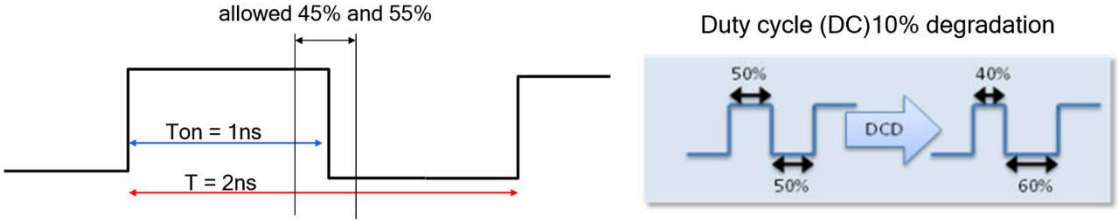


Figure 7.1: Duty Cycle Distortion.

Duty cycle distortion (DCD) is a propagation delay difference between low-to-high and high-to-low transitions of the clock and is typically expressed in percent. With 7nm

and below, deep clock distributions are prone to DCD accumulation as the signal propagates through different levels. With millions of gates on a single clock domain, even a picosecond DCD per gate will add up to significant distortions at the end points. While DCD results from manufacturing process variations, marginal designs and electrical noise, it gets worse with transistor aging. Traditionally, duty cycle correcting circuits have been added to designs to remedy the problem.

The duty-cycle distortion is the digital-signal impairment that appears because all rising-edges are delayed (or advanced) by the same value Δt_{rise} from the expected moment in time and all falling-edges appear delayed (or advanced) by the same value Δt_{fall} from the expected moment in time.

T_1 = duration of an occurrence of a single logic-1 affected by DCD.
T_0 = duration of an occurrence of a single logic-0 affected by DCD.
T_{bit} = duration of a bit if it was not affected by DCD.

Considering $\Delta t > 0$ for a delay and $\Delta t < 0$ for an advance in time, then:
$T_1 = T_{bit} + [\ \Delta t_{fall} - \Delta t_{rise}]$
$T_0 = T_{bit} - [\ \Delta t_{fall} - \Delta t_{rise}]$

and the DCD is:

$$DCD = \frac{T_1}{T_1+T_0} = \frac{T_1}{2*T_{bit}} = 0.5 + \frac{\Delta t_{fall} - \Delta t_{rise}}{2*T_{bit}}$$

7.2 Aging of Transistors

As transistors are switched on and off the drain currents can over time slowly decrease, this in turn changes path delays that make the chip speed slow down, and even fail to meet specifications. The aging effects change the Vt of devices, which in turn will slow down the rise and fall times of the clock signals. These aging effects over time will distort the duty cycle of the clock and can actually cause the clock circuitry to fail.

Device aging is dependent on workload, Voltage, Temperature and Frequency. The effects that cause transistor performance to shift over age are:

i. Hot Carrier Injection (HCI)
ii. Bias Temperature Instability (NBTI/PBTI)
iii. Self-Heating Effect (SHE)
iv. Time Dependent Dielectric Breakdown (TDDB)

a) Hot Carrier Injection (HCI)

HCI is commonly associated with a device operating in the saturation region – also, commonly referred to as "pinchoff" at the drain node. Carriers accelerated through the pinchoff depletion region are subjected to the gate-to-drain electric field. These carriers may originate from the channel current and/or from secondary carriers due to impact ionization. These energetic carriers may undergo a collision resulting in a vertical velocity vector and may then trap in the dielectric stack near the drain. Hot carriers may also break chemical bonds in the dielectric stack, resulting in the generation of additional traps.

The result is a localized reduction in the gate-to-drain electric field, as part of the electric field now terminates on the trapped charge. This is typically modelled as a reduction in the effective channel carrier mobility.

HCI is a current related event and happens when the transistor is switching. The effect is dependent on the gate load, input slew rate, voltage, and input frequency. In Figure 7.2 shows how the electron moves from channel to gate oxide region.

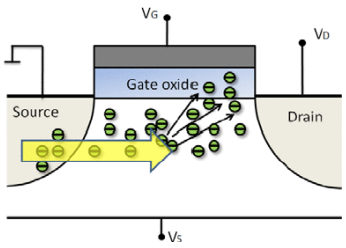


Figure 7.2: The HCI effect due to electron movement.

b) Bias Temperature Instability (NBTI/PBTI)

The BTI effect occurs even when there is no current flow (I_{on}) from source to drain, the gate voltage can cause charges to migrate into the insulating gate oxide. The effect is partly reversible if gate voltage is removed, the charges present in gate oxide will quickly leave and this effect is difficult to measure. The BTI occurs due to trapped carriers cause a shift in V_{th}.

Channel carriers enter the dielectric stack and fill trap states. BTI manifests as an adverse shift in the device threshold voltage – i.e., an increase in the absolute value of Vt for both nMOS and pMOS devices. Negative BTI (NBTI) refers to the pMOS device Vt shift, due to the negative gate-to-channel electric field direction; PBTI refers to the nMOS device Vt shift.

The delta in the threshold voltage eventually saturates over time as trap states are filled. Note that BTI models also include a (partial) recovery in the Vt shift for the time period when the device gate-to-channel electric field is reversed, as depicted below.

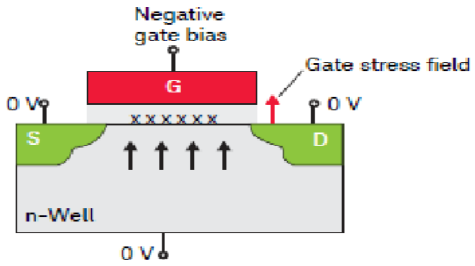


Figure 7.3: Bias Temperature Instability effect.

The BTI aging and its effects:

- Effect on path delay and transistor drive.
- Effect on clock skew.
- Effect on performance closure w.r.t setup and hold margins.
- Effect on Duty cycle of Clock path.
- Effect on Clock transition time.

c) Self-Heating Effect (SHE)

The vertical fins are wrapped in oxide layer, they operate at high voltages, high current densities, and high operating temperatures. These combination with poor heat dissipation led to localized thermal effects known as self-heating. The self-heating is localized effect means it doesn't affect certain circuit blocks, it affects individual fins within a single FinFETs.

Heat is generated by current flowing through channel and dissipates through the contacts, thermal coupling to the bulk. In multi-fin structure, central fins tend to be much warmer because they are away from contacts. This effect can accelerate aging /reliability issue of the device.

d) Time Dependent Dielectric Breakdown

The application of voltage across Gate can cause electrically active defects within oxide layers, trap charges. The amount of these charges accumulated in oxide layer can create a short circuit in devices. The Electric field and The oxide breakdown is catastrophic effect or device failure.

HCI and BTI involve carrier trapping and de-trapping of charges. Applying a bias voltage populates trap locations with carriers; removing it allows the carriers to return to ground state, restoring the transistors original behaviour. The reliability depends on circuits Duty Cycle (DC). The rapid switching of CMOS devices is an advantage, it allows more recovery time than traditional DC stress.

7.3 Aging Modelling

The Aged devices from aging library provided from the foundry. In this case of devices, we park clock value, then only one edge of clock will be affected during aging analysis, while devices with clock toggling will have both edges affected during aging analysis. So, the Duty Cycle Delay (DCD) shape will depend on your circuit topology. In Figure 7.4 shows DCD shape depend on circuit topology.

The circuit is simulated at different stress level these are :

i. DC Stress

ii. AC Stress

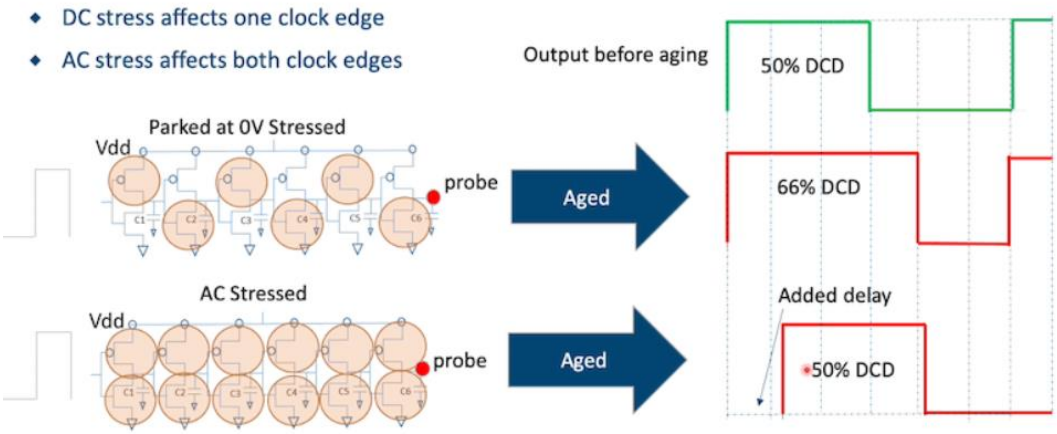


Figure 7.4: DCD shape under different stress.

a) DC Stress: Clock root parked at high or low. The DC stress is considered as worst condition check.
   - DC = 0v. Clock source path at 0v, fall-path is aged for max-pulse DCD check.
   - DC = 1v. Clock source path at 1v, rise-path is aged for min-pulse DCD check.

b) AC Stress: Toggle based stress; the toggling allows recovery of transistors.

7.4 Rail-to-Rail Failure (R2R)

A power supply line provided by a power supply unit is referred to as a power rail. The entire range from the maximum voltage of a power line (V_{cc}) to its minimum voltage (GND) is referred as rail-to-rail. So, the rail-to-rail failure means the signal is unable to get required voltage i.e., either it is not getting required high voltage or required minimum voltage.

In Clock paths signal unable to reach or touch voltage rails then it is R2R failure. The R2R failure is catastrophic effect, it means the clock is getting dead or unable to get proper power supply. This might be occurring because of weak driver or net routing is very long. In Figure 7.5 shows the rail-to-rail failure as the signal unable to reach voltage rails.

The threshold to categorize as Logic 1 can be 95% of V_{cc} and for Logic 0 can be 5% of GND. This threshold depends on technology node. In Figure 7.5 we can see Logic 1 V_{cc} is 0.785V and GND is 15mV.

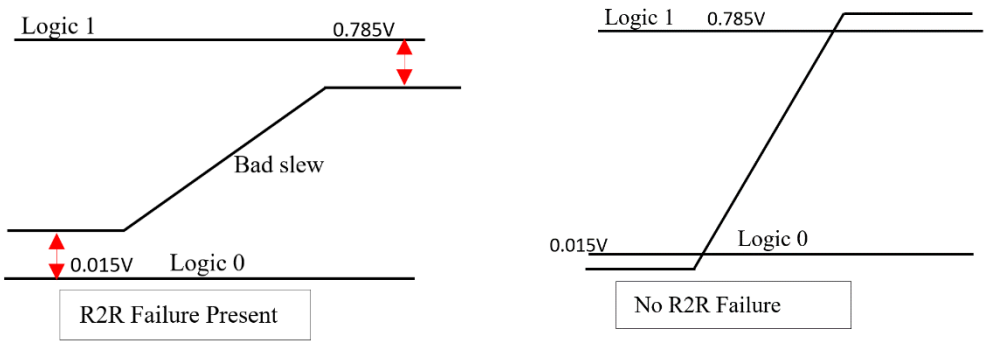


Figure 7.5: Rail-to-Rail failure mechanism.

7.5 DCD Violation

The causes for Duty Cycle Distortion and Rail-to-Rail failure are due to following:

- Process skew.
- Bad Slew or High transition: The transition time of a net is the longest time required for its driving pin to change logic values. When a signal takes too long transiting from one logic level to another, a transition violation is reported. This led to R2R failure.
- Cell V_{th} (threshold voltage) type: The cell V_{th} can also make the delay faster or slower based on threshold voltage needed to turn on the device. The variation in the trigger threshold value over the time (aging effect). This leads to the rising and falling edges to appear at moments in time slightly different than expected. Thus, generating a duty cycle distortion which manifests as jitter on the original clock signal.

- Driver size of the cell: The large driving cell can drive a larger load and decrease the delay of the cell or reduce the driving strength of the cell then will increase the delay of the cell.
- Long chain of buffer and inverter: The clock path cell level must be small or there should not be many levels in clock path.
- Lengthy buffer chain: The buffers generally have unequal rise and fall time when compared to inverters. The buffers chain generally leads to duty cycle getting distorted more.

7.6 Jitter

The jitter is deviation in timing between Ideal and actual clock signal. In Figure 7.6 shows that jitter waveform. The sources of PLL jitter:

i. PLL jitter.
ii. Noisy input signal.
iii. PDN induced jitter.

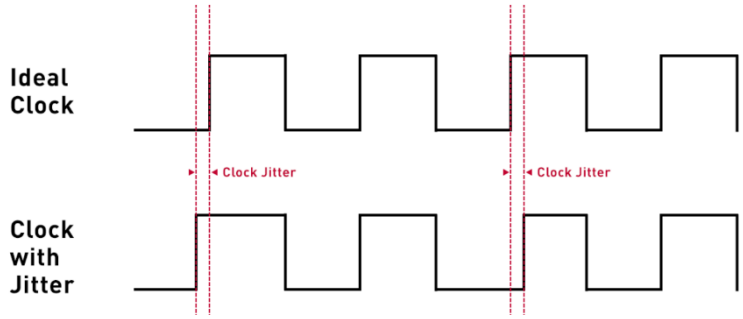


Figure 7.6: The Jitter Waveform

7.7 Min Pulse Width (MPW)

The Min pulse width check is a part of STA check, that checks pulse width shrinking on clock path. This is check is available in Primetime or Tempus tool as inbuilt command. The MPW calculation doesn't account for:

i. Rail-to-Rail (R2R) checks: The tool cannot detect where the clock goes off or clock becomes dead.
ii. Aged libraries: The tool doesn't consider the OMI models pf the standard cell libraries.

The Primetime or Tempus tool doesn't give accurate simulation because SPICE simulation is not done. Therefore, for Duty Cycle Distortion (DCD) simulation is done using Clockedge tool provided by Infinisim. This tool can detect R2R failure, aged libraries and can provide accurate results.

7.8 STA and Infinisim

The STA tool does not compute the full clock signal waveforms. Instead, they estimate timing values by inferring them from pre-characterized timing libraries for different PVT corners. While this makes STA fast, it is not accurate enough at finer geometries, failing to detect DCD and rail-to-rail failures directly.

Infinisim' s ClockEdge is a high-capacity, high-performance, SPICE accurate, end-to-end integrated clock analysis solution. ClockEdge can handle chips that incorporate multiple topology high-speed clocks and used for full-chip sign-off. It plugs into current design flows, allowing designers to simulate large clock domains with millions of gates.

The following sections gives brief explanation of STA limitations and Infinisim tool how it overcomes STA limitations.

a) Static Timing Analysis Limitations

The following list STA limitations

i. Duty Cycle Distortion (DCD): STA tools use approximate interconnect delay model and often miss duty cycle distortion errors in long signal nets due to resistive shielding.
ii. Rail-to-Rail Failure Detection: STA tool do not catch R2R failure directly. The STA measures insertion delays and slew rates. This doesn't compute full clock signal waveforms.
iii. Aging: The STA tool doesn't consider how the clocks are parked during idle states. There is very limited data on aging impacted R2R, duty cycle and insertion delay.

iv. Jitter: The STA tools doesn't simulate jitter as part of the tool. The tool assumes a margin for jitter that might be overly pessimistic or optimistic.

b) Infinisim approach

The following list explains how Infinisim overcomes STA limitations.

i. Duty Cycle Distortion (DCD): The Clockedge tool computes duty cycle using analog simulation of the entire circuit with full interconnect. The tool also identifies nets failing duty cycle requirements using user defined criteria. The Figure 7.7 shows Infinisim output for a clock duty cycle.

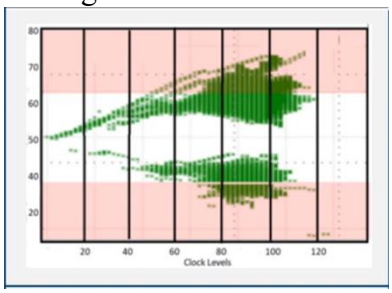


Figure 7.7: Duty Cycle results from Infinisim tool. (Source: Infinisim Webinar)

ii. Rail-to-Rail Failure Detection: The Infinisim tool generates analog signal waveforms for the clock. The tool also estimates the frequency at which R2R failure will occur at each gate and identify weak driver gates. Measure the signals that donot cross the supply voltages.
In Figure 7.8 shows the R2R failure for a clock path.

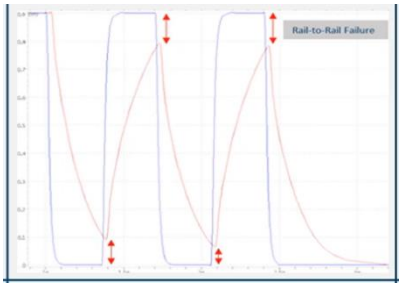


Figure 7.8: Rail-to-Rail Failure detection on clock path. (Source: Infinisim Webinar)

iii. Aging: The tool performs comprehensive full clock domain signal degradation analysis. The multiple parking strategies simulated for

analysis. In Figure 7.9 shows the duty cycle difference between fresh and aged effect.

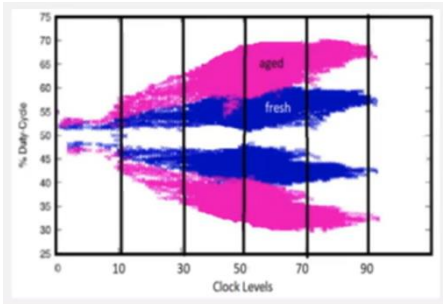


Figure 7.9: Duty cycle values of Fresh and Aged effect. (Source: Infinisim Webinar)

iv. Jitter: The tool computes impact of timing. The clock domains with multiple power supply noise profiles are used for block-level jitter analysis.

7.9 Infinisim' s ClockEdge

Infinisim' s ClockEdge is a high-capacity, high-performance, SPICE accurate, end-to-end integrated clock analysis solution. ClockEdge can handle chips that incorporate multiple topology high-speed clocks and used for full-chip sign-off. It plugs into current design flows, allowing designers to simulate large clock domains with millions of gates. Infinisim tools designers can look for aging issues and then apply changes to clock gating, such as holding a clock high versus low, etc. It is then possible to iterate and look to see if issues such as duty cycle distortion, clock skew, slew or insertion delay are going to be a problem.

Features of ClockEdge:

- SPICE accurate results overnight, for clock domains containing 4+ million gates.
- Leverages distributed computing to simulate and analyze large complex clocks.
- Handles complex clock topologies includes trees, grids/mesh, and spines.
- Reports include timing, process variation, power, and current analysis.

- OCV analysis: during design for guard-band reduction, in post-design phase to estimate yield.
- Results from ClockEdge are integrated into CTS flow for optimizing design.
- Timing optimization during design iterations.
- Base-layer tapeout/Metal-layer tapeout signoff verification.
- Post-fab investigation into performance degradation and potential improvements for next revision.

7.10 ClockEdge Flow

The ClockEdge inputs are like STA these are:

- Gate level Netlist (Verilog): The gate level circuit description in Verilog or VHDL language.
- Constraints: Sets of constraints to enable timing checks of the paths that dictate the desired performance of the design.
- Technology library: Technology file defines basic characteristic of cell library pertaining to a particular technology node. The technology library includes timing information and contains several other attributes.
- Parasitics: The nets have parasitic resistance and capacitance that are captured by SPEF.

The ClockEdge flow is Verilog gate-level clock domain tracing and SPICE netlist generation. The Path sensitization to ensure clock signal propagation through complex gates. The tool is high speed, high capacity, and SPICE accurate circuit simulation for PVT.

The ClockEdge Outputs include:

- Slew rates at gate and interconnect nets.
- Gate delays in the clock path.
- Skew between launch and capture flops.
- Duty cycle at gate and interconnect nets.
- Peak-to-peak voltage swing.
- HTML based GUI to view reports and plots.
- Additional custom reports generated.

The ClockEdge Flow diagram with required inputs and output reports generated.

Chapter 8
Conclusion

Designing an SoC at 7nm and smaller process node is a big task, requiring specialized knowledge of clock analysis to ensure first-pass silicon success. The Static timing concepts and STA flow were discussed in detail. The Duty cycle distortion digital-signal impairment was discussed in detail. The definition of DCD based on edge-delay was presented.

Infinisim has the experience in analysing the effects of clock aging. The ClockEdge tool from Infinisim is focused on giving designers accurate aging analysis of their clock networks, providing results quickly overnight. You get to see both DC and AC stress conditions for your aged clock domains. ClockEdge delivers SPICE accurate results on clock domains containing 4+ millions of gates, and higher verification coverage compared to competitive products in the market. It easily plugs into current design flows used by customers. And it can accurately analyze top-level, block-level, and hard-macro level clocks to cover all blind spots. The tool finds DCD, jitter, aging and rail to rail issues that are routinely missed by traditional STA-based methodologies. Adding a new tool like ClockEdge into your EDA tool flow is a smart step to mitigate the effects of device aging and other effects.