DESIGN, CHARACTERIZATION AND AUTOMATION OF AN ULTRA-HIGH TEMPERATURE STANDARD CELL LIBRARY FOR HARSH ENVIORNMENTS

BY

VENKATARAMAN JEYARAMAN

Bachelor of Engineering

University of Madras

Chennai, India

2001

Submitted to the faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2004

DESIGN, CHARACTERIZATION AND AUTOMATION OF AN ULTRA-HIGH TEMPERATURE STANDARD CELL LIBRARY FOR HARSH ENVIORNMENTS

Thesis Approved:

Dr.Chris Hutchens

Thesis Advisor

Dr.Louis Johnson

Dr.Yumin Zhang

Dr.A.Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGEMENTS

First, I would like to thank my committee chairman and advisor Dr. Chris Hutchens, through whose patience, understanding, and valuable advice, this work has been accomplished. I would also like to express my gratitude to Dr. Louis Johnson and Dr. Yumin Zhang for serving as my committee members.

I would also like to convey my special thanks to the members of the MSVLSI (Mixed Signal VLSI) Group at the Department of Electrical and Computer Engineering, Oklahoma State University, namely – in no particular order – Dr. Liu, Xunyu Zhu, Narendra Kayathi, Sameer Kadam, Jerry Brewer, Vinay Chitturi, Jianning Wang and Saurabh Kasat for all their help and suggestions along the course of this work.

I would like to thank my Parents and Sister for their encouragement and motivation. I would like to thank my friends for their support. To all of them I dedicate this work.

This work was funded by Halliburton Energy Services.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Glossary

g_x Horizontal grid spacing.

g_y Vertical grid spacing.

s_s Safety zone required to avoid butting DRC errors.

w_p Power rail width

h Height of the cell

w_{use} Usable cell width

D_I Intrinsic delay

D_S Slope delay

D_C Interconnect delay

D_T Transition delay

T_{period} Time Period

T_{SU} Setup time

T_{HD} Hold time

$T_{Clk\text{-}Q}$ Clock to Q propagation delay

R_{out} Output Resistance

C_L Load Capacitance

f Number of fan-ins

R'_P Rise resistance of inverter

R'_N Fall resistance of inverter

C'_p PMOS gate capacitance of inverter

C'_n NMOS gate capacitance of inverter

N_{ML} Low noise margin

N_{MH} High noise margin

V_{th} Threshold voltage

V_T Thermal voltage

C_{ox} Gate oxide capacitance

μ_0 Zero bias mobility

m Subthreshold swing coefficient

W Width of transistor

L Length of transistor

F Number of fingers

Chapter 1

Introduction

In the recent years there has been a significant increase in the complexity and operating temperature of integrated circuits. As the complexity of circuit designs grows, it is becoming highly impossible to design and layout those circuits by hand. As the operating temperature increases, conventional bulk-Silicon devices fail to operate due to the drift of bulk device parameters. This project is focused to overcome these problems by building a cell library with SOS technology that can work at ultra-high temperatures (200^{o}C).

1.1 Comparison between SOS and Bulk Process

In conventional bulk-Silicon circuits the active elements are placed in the thin surface layer and a depletion layer isolates it from the silicon body. The leakage current of the PN junctions so formed exponentially increases with the temperature and is responsible for several serious reliability problems. Excessive leakage currents and high power dissipation limits the operation of bulk-Silicon circuits at high temperature. Another important limitation of bulk-Silicon circuits is the formation of parasitic n-p-n and p-n-p transistors in the neighboring insulating tubs resulting in latch-up and significantly degrades circuit performance.

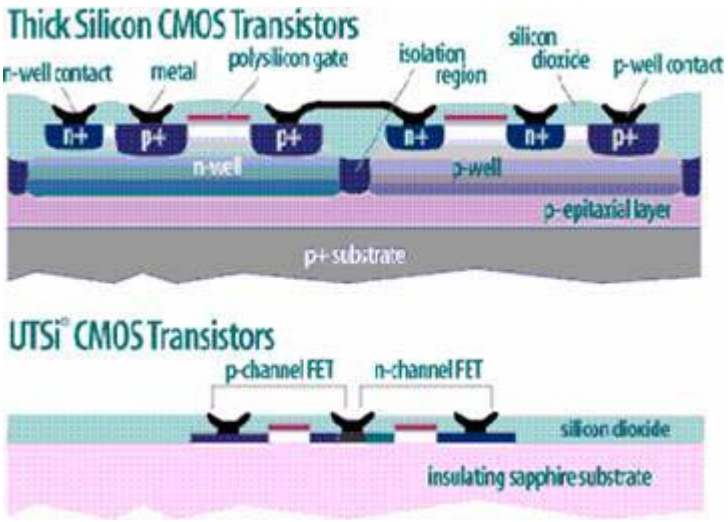

Figure 1.1. Comparison of standard bulk CMOS to Peregrine USTi® [1]

The drift in the device parameters of bulk CMOS device at elevated temperatures significantly affects the correct operation of both digital and analog circuits in terms of stand-by power dissipation, bandwidth, and precision and eventually results in complete loss of functionality or even to an extent of destruction of the device due to thermally-induced latch-up[19].

The focus of this thesis is to build a robust standard cell library for Ultra-High Temperature. A number of factors were considered to accomplish the successful operation of the cell library. From previous work done, the transistors were characterized to know the variation of mobility and threshold with variation in (W/L) ratios and temperature. At 225 ^{0}C, the SOS 3 input NAND gate shows only slight variations of noise margin, switching point and maximum gain.

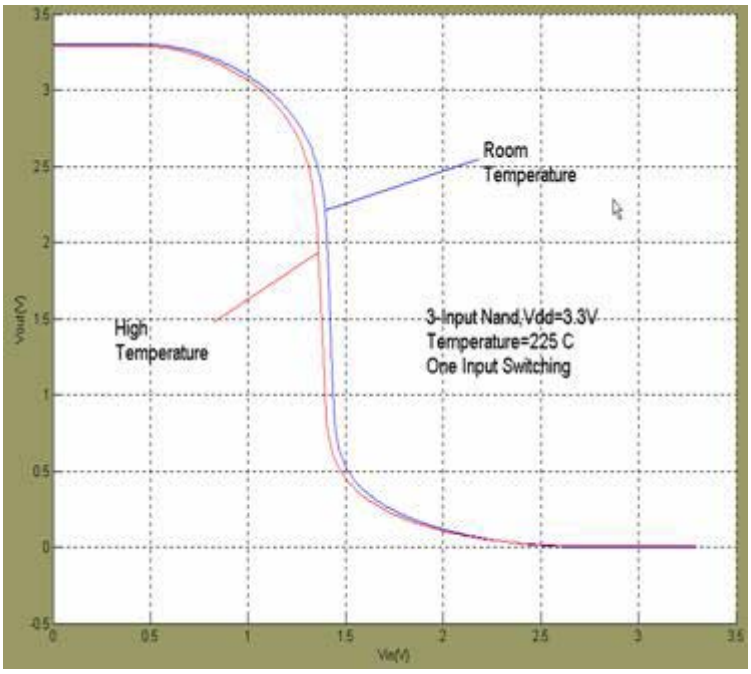

Figure 1.2. VTC curve of 3-Input NAND gate

The main challenge to make the cell library robust is to minimize the leakage current by selecting the correct value of L without affecting the bandwidth too much. Traditionally the leakage of the chip is estimated using the leakage per micron obtained from sub-threshold behavior of the transistors.

The sub threshold current can be expressed based on the following [20],

$$I_{ds} = \mu_0 C_{ox} \frac{W}{L}(m-1)(v_T)^2 \times e^{(V_g - V_{th})/mv_T} \times (1 - e^{-v_{DS}/v_T})$$

$$m = 1 + \frac{C_{dm}}{C_{ox}} = 1 + \frac{\frac{\varepsilon_{si}}{W_{dm}}}{\frac{\varepsilon_{ox}}{t_{ox}}} = 1 + \frac{3t_{ox}}{W_{dm}}$$

Where V_{th} is the threshold voltage, and $v_T = KT/q$ is the thermal voltage, C_{ox} is the gate oxide capacitance; μ_0 is the zero bias mobility and m is the subthreshold swing coefficient. W_{dm} is the maximum depletion layer width, and t_{ox} is the gate oxide thickness (approximately 95nm). C_{dm} is the capacitance of depletion layer. From the above equation the leakage current decreases with increase in the length of the transistor. The length of 1.6um was selected after testing transistors with different lengths to satisfy bandwidth, threshold voltage degradation and noise margin requirements.

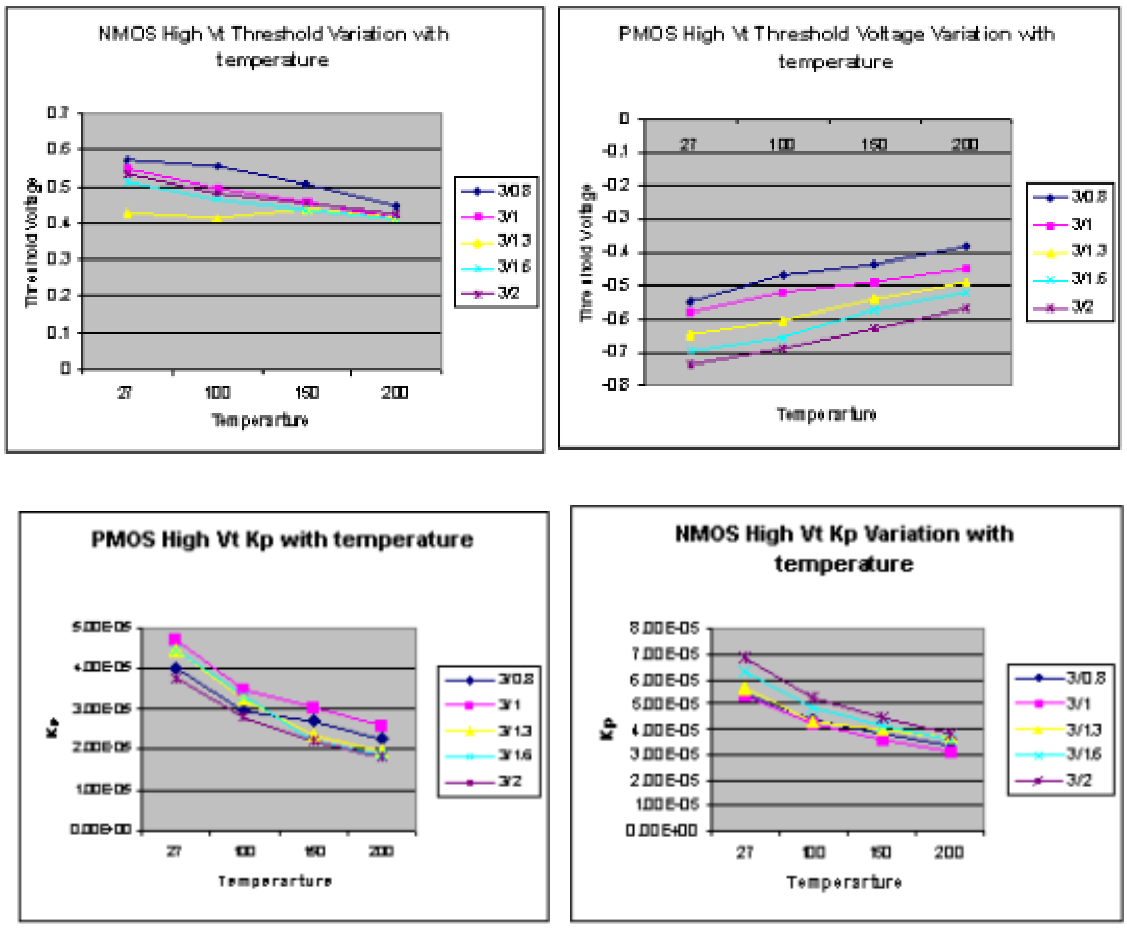

Figure 1.3. Variation of Kp and Threshold with Temperature

The reduction of leakage current also results from reduction of the drain junction area and from a change of physical mechanism in SOS [1], the quasi-neutral region surrounding the drain junction is totally suppressed and the generation/recombination mechanisms in the depletion region is dominant so that the leakage current only increases with temperature as intrinsic concentration n_i instead of square of n_i as in the case of bulk devices. Subsequently, without even taking into account the very large well-to-substrate leakage currents, SOS has the potential to offer a reduction of static power dissipation by at least 2-3 orders of magnitude at 200 C when compared to bulk. The variation of threshold voltage with temperature is as low as 1 mV/^{o}C because of the suppression of depletion width, which is temperature dependent [1]. From the above plots it is clear that the mobility of PMOS devices are lesser than the NMOS devices, hence the noise margins of the inverters with minimum geometry have $N_{ML} < N_{MH}$. In order to maximize the noise margins the transistors are beta matched with a 1.8 beta ratio.

One of the disadvantages of SOS is the floating body problem; hence transmission gates are avoided in the design of cell library. Also previous test results show that for this process the NMOS devices tend to leak more than PMOS devices hence NAND type structures are mostly used with greater frequency than NOR type structures because in NOR type structures there are multiple paths for the NMOS leakage current to flow but in NAND there is only one path for leakage current to flow. Taking into account of all these factors the cell library is built in Peregrine SOS Process for sustaining ultra high temperature and is compatible with synthesis tools such as Cadence and Synopsys.

1.2 Synthesis Tool-Based Design

In general, synthesis tool-based designs are performed using the following steps [2]:

1. Description of circuit behavior in some high-level language, such as Verilog, VHDL or System C.
2. Compilation of behavioral or RTL description into a logical netlist using logic synthesis tools such as Cadence or Synopsys.
3. Translation of the logical netlist into a geometric netlist, followed by placement and routing, with Placement-and-Routing (PNR) tools.

The second step presumes that the design environment already contains descriptions of some structural logic primitives (e.g. primitives for NAND gates, latches, flip- flops, etc), as those primitives will comprise the netlist produced by the synthesis tool. Similarly, the last step presumes that the translation of a netlist to geometric shapes is already defined for the design environment, i.e. The logic primitives referred to by the netlist is already present in some physical library. Hence, for the design environment, a library which contains both physical (i.e. layout) primitives and logic primitives which correspond to those structural primitives must already be present.

Therefore, with this design method, it is mandatory that a standard cell library be present.

Further, the standard cell library should consist of:

1. Layout, Schematic and Symbols with a fixed Naming Convention.
2. Logic description libraries, both for synthesis and simulation purposes, which features simplified timing and power dissipation modeling capabilities
3. Other geometric descriptions as needed by the PNR tools, if the full layout is deemed too complicated for this purpose.
4. List of logic primitives which correspond to those cells, including pinout.

1.3 Thesis Organization

This thesis consists of 6 chapters. Chapter 2 describes the details of cell library, layout format and its implementation. Chapter 3 describes the literary review of various delay models and estimation of linear delay model parameters. Chapter 4 describes the relationships between intrinsic delay with internal architecture and transition delay with output loading of the cell, hardware test plans for verifying the delay model is generated. Chapter 5 summarizes the measurements and test results and Chapter 6 is conclusion and future work on the cell library.

Chapter 2

Cell Library Format

The following section starts with an introduction to ASIC design and then describes the process flow for the cell library, design considerations of the cell library, their layout formats, cell grid selection, and an overview of a fully routed chip. A complete listing of all the cells resides in Appendix A. The final section 2.4 shows an example of a cell.

2.1 Introduction to ASIC design

Typically, ASIC design starts with the functional description of the physical layout with some kind of high level hardware language such as Verilog, VHDL or system C. The flow diagram of ASIC design is shown in Figure 2.1.

The flow diagram shows the various steps involved in ASIC design. The cell library contains the description of all the cells to facilitate synthesis for translation of the behavioral code to netlist and have the physical description of the cells for the place and route tool to route the design.

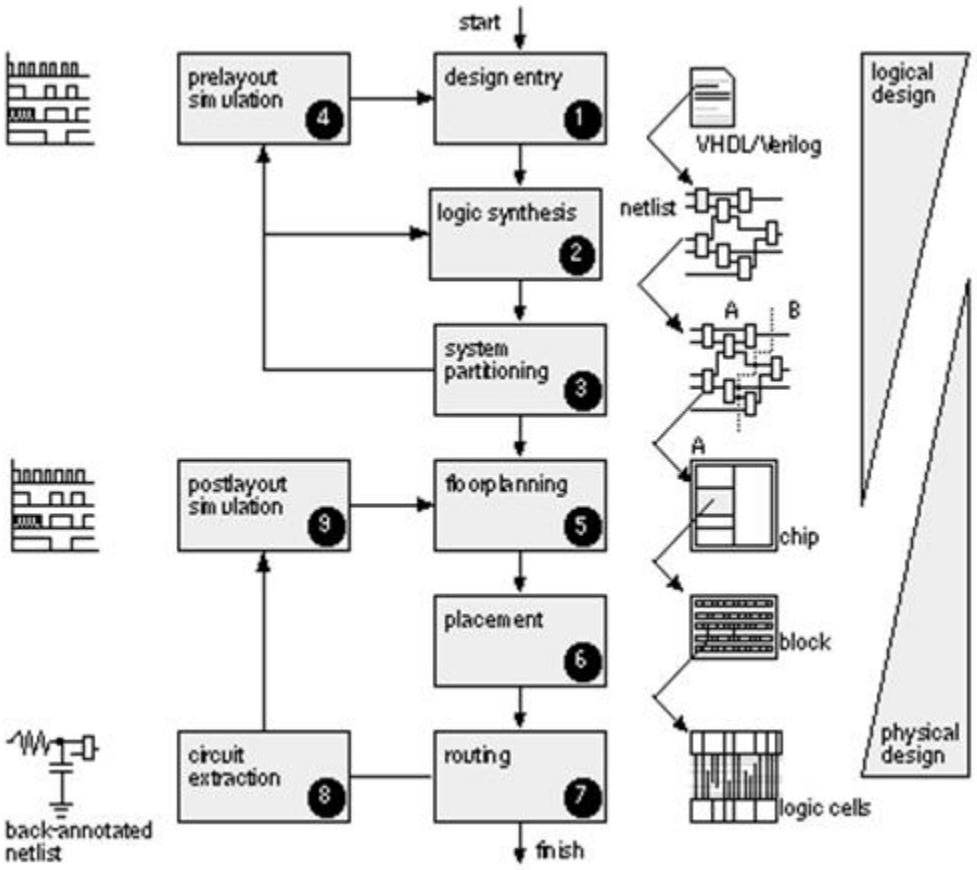

Figure 2.1. ASIC Design Flow [21]

The different blocks in ASIC design are as follows[21]:

1. Design entry- Enter the design into an ASIC design system, either using a hardware description language (HDL) or schematic entry.
2. Logic synthesis- Use an HDL (VHDL or Verilog) and a logic synthesis tool to produce a netlist —a description of the logic cells and their connections.
3. System partitioning- Divide a large system into ASIC-sized pieces.
4. Prelayout simulation- Check to see if the design functions correctly.
5. Floorplanning- Arrange the blocks of the netlist on the chip.
6. Placement- Decide the locations of cells in a block.

7. Routing- Make the connections between cells and blocks.
8. Extraction- Determine the resistance and capacitance of the interconnect.
9. Post layout simulation- Check to see the design still works with the added loads of the interconnect.

The process flow can be divided into two main stages:

1. Logical Design- In this phase of the design the designer has more control over the design. This is a very important phase, if the logic is wrong it must be fixed here. The delay format file (.lib) is used to translate the RTL to netlist and this is performed repeatedly to make sure the speed, logic and power satisfy the needs.

2. Physical Design – In this phase of the design the designer has more control over the area and static timing. The LEF(Library Exchange Format) file is provided to Silicon Ensemble P&R tool, which contains a partial description of all the cells in the library, some design rules pertinent to placement and routing process (such as metal and via spacing) and routing rules defined by the designer of the library (such as pitch and direction of metal tracks).

Steps 1–4 are part of logical design, and steps 5–9 are part of physical design. There is some overlap. For example, system partitioning might be considered as either logical or physical design. To put it another way, when we are performing system partitioning we have to consider both logical and physical factors[4]. It is important to note that both the steps are an iterative process until speed, power and area criteria are met.

2.2 Standard cell Library Process Flow

Basically, the process involves three stages: creating the standard cells, extracting timing from each cell to create the timing file for synthesis and creating physical description library for place and routing. Creating standard cells involves selecting the right cells for the cell library drawing layouts according to specifications, schematics and symbols performing LVS and extracting SPICE level netlist for each cell. The timing is extracted using OCEAN (Open Command Environment for Analysis) Scripting which automates spice simulations to extract the intrinsic rise time, intrinsic fall time, rise resistance, fall resistance, setup time and hold time from the cells, an example script for an inverter is in Appendix B. These timing information is put together to form a .lib file which can be used by synthesis tool to convert the Behavioral code to verilog netlist and meet timing constraints.

Physical design involves in creating the LEF file from Abstract generator, which basically has the definitions of different routing layers and preferences of these layers in routing, the area of each cell along with the dimensions of the power and ground rails and the pin positions of each cell so that the router can route the I/O pins.

However created, each cell in an ASIC cell library must contain the following:

- A physical layout
- A behavioral model
- A Verilog/VHDL model

- A detailed timing model
- A test strategy
- A circuit schematic
- A cell icon
- A wire-load model
- A routing model

The physical layout describes the actual dimension of the cell with all the pin connections. The behavioral model describes the function of the cell without any actual dimensions of the cell. A Verilog/VHDL model is used to describe the cell during synthesis. While a detailed timing model of the library is used during synthesis for the synthesizer to optimize the design and verify the timing requirements of the design. A detailed test strategy is required to evaluate the performance of the cell library before it is actually used to generate layouts and is explained later. A circuit schematic is required to perform LVS with layout and make sure it is error free. A wire load model is needed to estimate the parasitic RC delay of the interconnects. Synthesizers do not take wire delay into account while synthesizing hence a separate wire load model has to be provided to the synthesizer and the additional wire delay forced on to the total delay of the design. A routing model is necessary for proper routing of the design in order to avoid large skews in the design and care must be taken while routing global signals such as clock lines where large skews could affect the performance of the design. Hence good clock tree models have to be developed and included in the cell library.

2.3. Cell Format

Below, in Figure 2.2, is a summary of the layout rules for the standard cell library. The power rails are 5um wide, routed horizontally in metal1. The I/O of the cell is routed vertically in metal2 over the cell, connecting to terminal pins defined by labeled metal2-metal1 pins. All I/O pins are placed on a g_x by g_y grid (called routing pitch), which starts g_x in from both vertical edges and $g_y/2$ in from the horizontal cell edge. All cells will be n g_x by m g_x**.** Since there is over-the-cell routing, I/O terminals can be placed anywhere on the predefined grid points.

Also since routing tools use fixed-grid two-level routing the terminals must have a center-to-center spacing along both axes [9]. The overall width of the cell is an integer multiple of g_x. All metal1 must be wholly contained between the power rails, only polysilicon and locos are allowed to extend to within $\mathbf{s}_s$ of the cell boundary. Metal2 runs vertically with Metal1 running horizontally when used. The grid spacing g_x and g_y are set respectively by the minimum spacing of two m1 m2 vias.

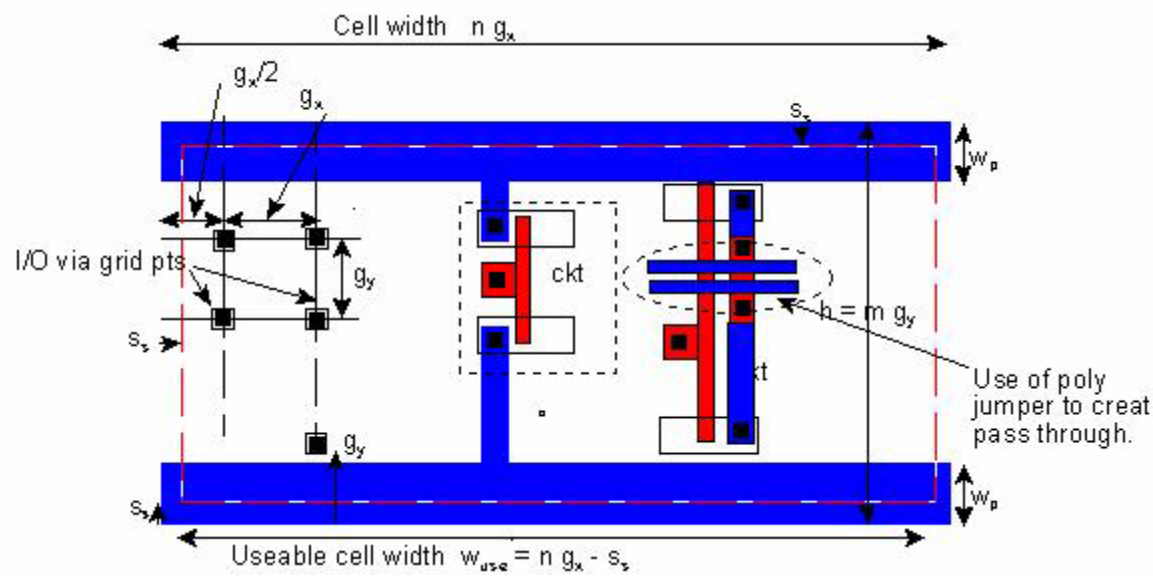

Figure 2.2. Layout for logic cell library.

Parameter	Value (um)	Comment
g_x	2.2um	Horizontal grid spacing.(isolated metal width)
g_y	2.2um	Vertical grid spacing.
s_s	0.5um	Safety zone required to avoid butting DRC errors.
w_p	5um	Power rail width
h	55um	m equal 25
w_{use}	n g_y - 2s_s	n must be an integer

Table 1.1 Cell geometry definitions and values.

The routing pitch should at minimum be a line-to-via pitch, as defined in Figure 2.3(b), where the closest separation (line to metal extension of via) still satisfies design rule for metal-to-metal separation. Ideally, it should be at least via-to-via pitch (see picture 2.3(c)) which is 2.2um. This will allow the routing tool to drop via where necessary. Using only line-to-line pitch as in Figure 2.3 (a) is avoided as the routing tool may fail since it is unable to drop a via when it is needed. All available metal layers follow the rules, even if there is no intention to actually use it for routing. Two metal layers are

available for routing during the LEF file generation, and this will allow the routing tool to decide which metal layer will actually be used. This in turn will more likely result in better routing.

In a multi-metal process, if for any reason the routing pitch is not identical for all metal layers, then the ratio of pitch between any two metal layers should be kept simple, such as 2:1 or 3:2 (ideally should be 1:1 if possible). Complex ratios, such as 11:9, should be avoided. In this library 1:1 routing pitch is used so that the router can drop a via where ever needed.

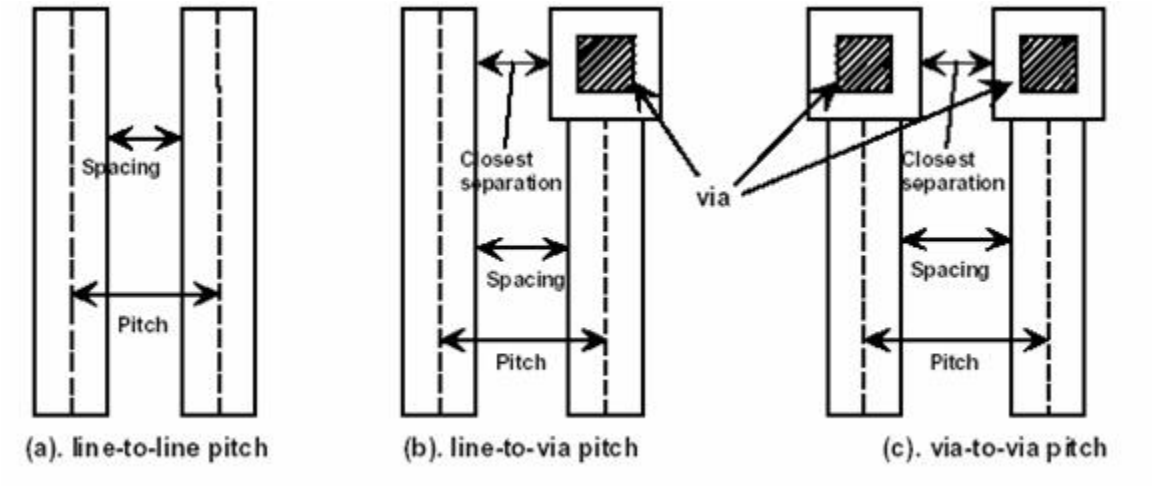

Figure 2.3. Definition of Routing Pitch [2]

The transistors are stacked horizontally, with their width parallel to the vertical axis as shown in the above figure. The maximum width of any single PMOS transistor is 18 um, with the largest NMOS 9 um. While larger widths could be drawn the widths are set as such to minimize gate delay. Larger effective device widths must be composed of multiple fingers**.** Multiple fingers are designated by the following nomenclature

F@Wum. Where F is the number of fingers and W is the width of the finger. In addition, poly and n locos may also be used within the cell proper as an interconnect material however; all runs must be less than 16 squares. For example if you tap the center of the input line poly of an inverter you can have full 32 squares in series or for 1um wide ploy this is better than a 16um ploy run to each gate from the input.

The cells are tiled horizontally and vertically, so there must be no design rule violations when cells are abutted. This keeps all cell poly, metal1 and locos a minimum of s_s from the north and south edges of the power rail and the east and west edges of the cell. No metal1 or metal2 should be placed outside the power rails, as this would conflict with the router.

Generally, no metal2 is used inside the cell, as it must be made available for external routing and I/O connections. Every unused metal2 slot is used for a routing channel over the cell. The library contains a total of 89 cells. Cells present include the range of 2, 3 and 4 input "simple" gates -- nand/and, nor/or, muxes, and-or / or-and, etc. There are a wide range of inverters and buffers, 1X to 4X, to cover a range of drivable loads. There are 8 types of flip-flops, and 10 latches. Lastly, there are various adder and subtractor bit slices and pad cells.

For the SOS process (V_{TN} = 0.720V, V_{TP} = -0.850V at room temperature [24]) high V_T devices are chosen as V_T degrades with temperature and still has the required noise margin, the average delay through an inverter is 0.3ns. For gates with longer stacks of

devices -- a four-input NOR, for example -- the delay can range up to 4.5ns. The effective switched input capacitance of the typical 1X inverter is a mere 30fF. Using a 3-input NAND gate as a more typical gate, the "average" gate has a delay of 0.9ns, and an energy consumption of 81.6 fJ at 3.3V. This corresponds to 81.6nW per MHz of effective throughput.

All gates are “optimal noise margin” ratioed and as a result the effective Wn to Wp ratio is 1.8 for all gates. By example for a 1X inverter $(W/L)_n$ equal 2um/1.6um and $(W/L)_p$ equal 3.6um /1.6 um. For a 3x 3 input NAND (W/L)n equal 3x3x2um/1.6um or laid out as 3@6um/1.6um while (W/L)p for the PMOS devices is 3x3.6 um/1.6um (10.8um/1.6um) No NMOS or PMOS device is less than 2um/1.6um or 3.6um/1.6um or greater than 9um or 18um respectively.

The general layout of Silicon Ensemble chip is as shown in Figure 2.4.The total area of the chip is determined by the padframe and depends on the number of power, ground and I/O pads used.

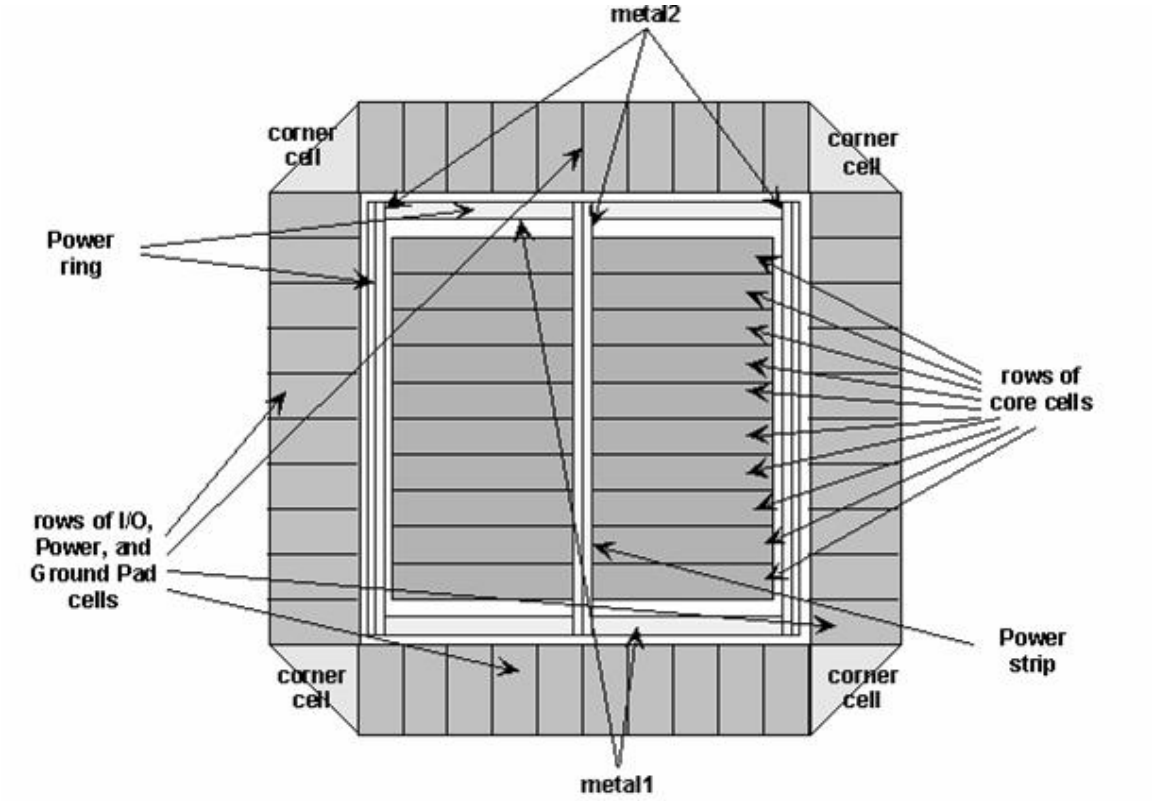

Figure 2.4. The general layout of a complete SE-routed chip [2]

Inside the padframe is the power ring for equal distribution of power through out the chip and the core cells are placed inside the power ring. For each metal layer, the direction could be horizontal or vertical, but one direction is always taken as the preferred direction, and the other one is automatically non-preferred in this case Metal1 is routed horizontally and Metal2 is routed vertically. The placement of the cells and routing is done according to the verilog netlist given to the SE by the designer. The rules previously discussed are necessitated by the way SE performs routing.

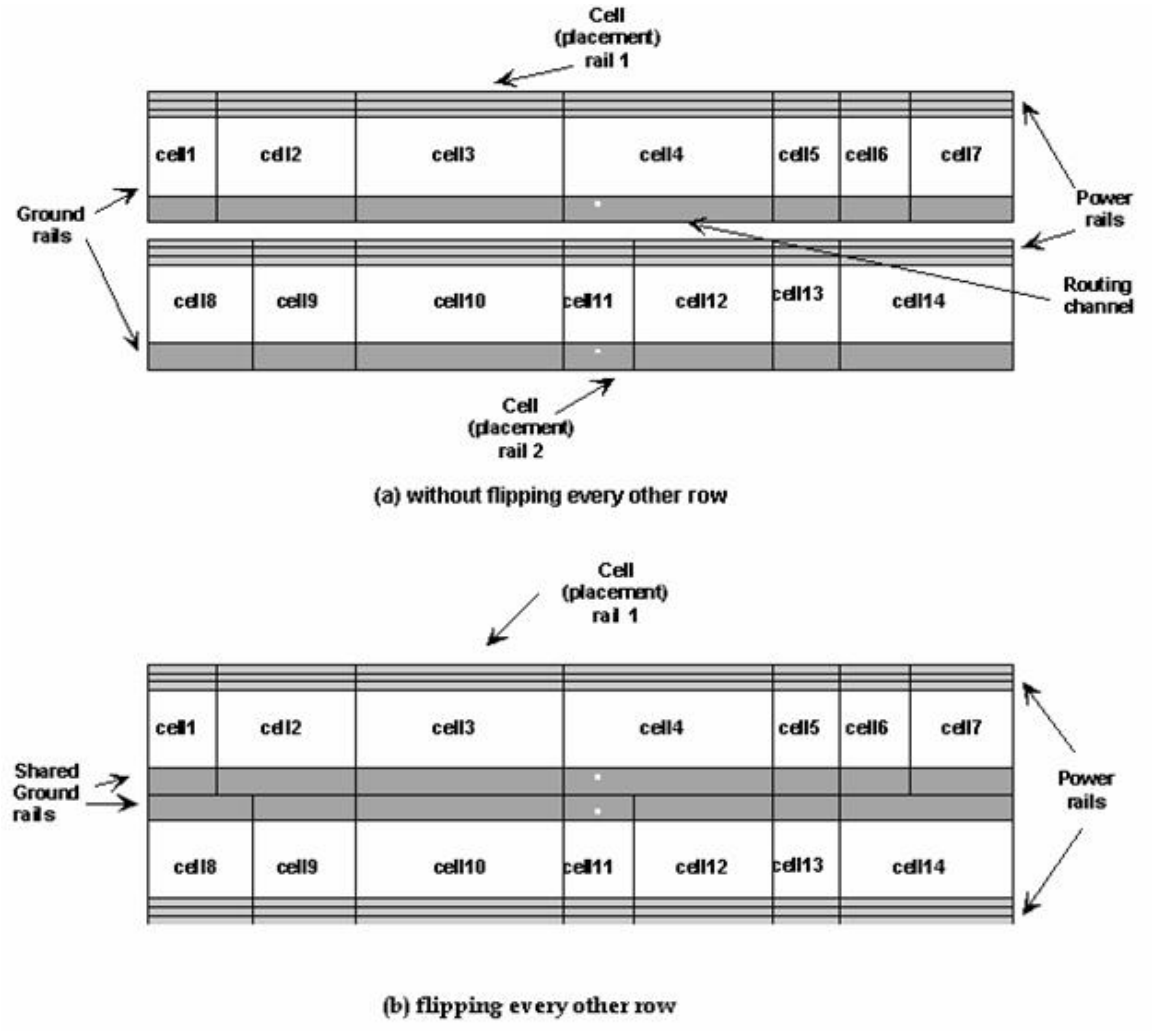

Figure 2.5. Stacking of Cells by SE [2]

The stacking of the cells is as shown in Figure 2.5. There are two ways of stacking the cells inside the power rings one way is to stack them without flipping every other row, this is not used as it consumes more area the other way is to flip every other row so that the power and ground rings are overlapped alternately saving area.

2.4. Example of a cell

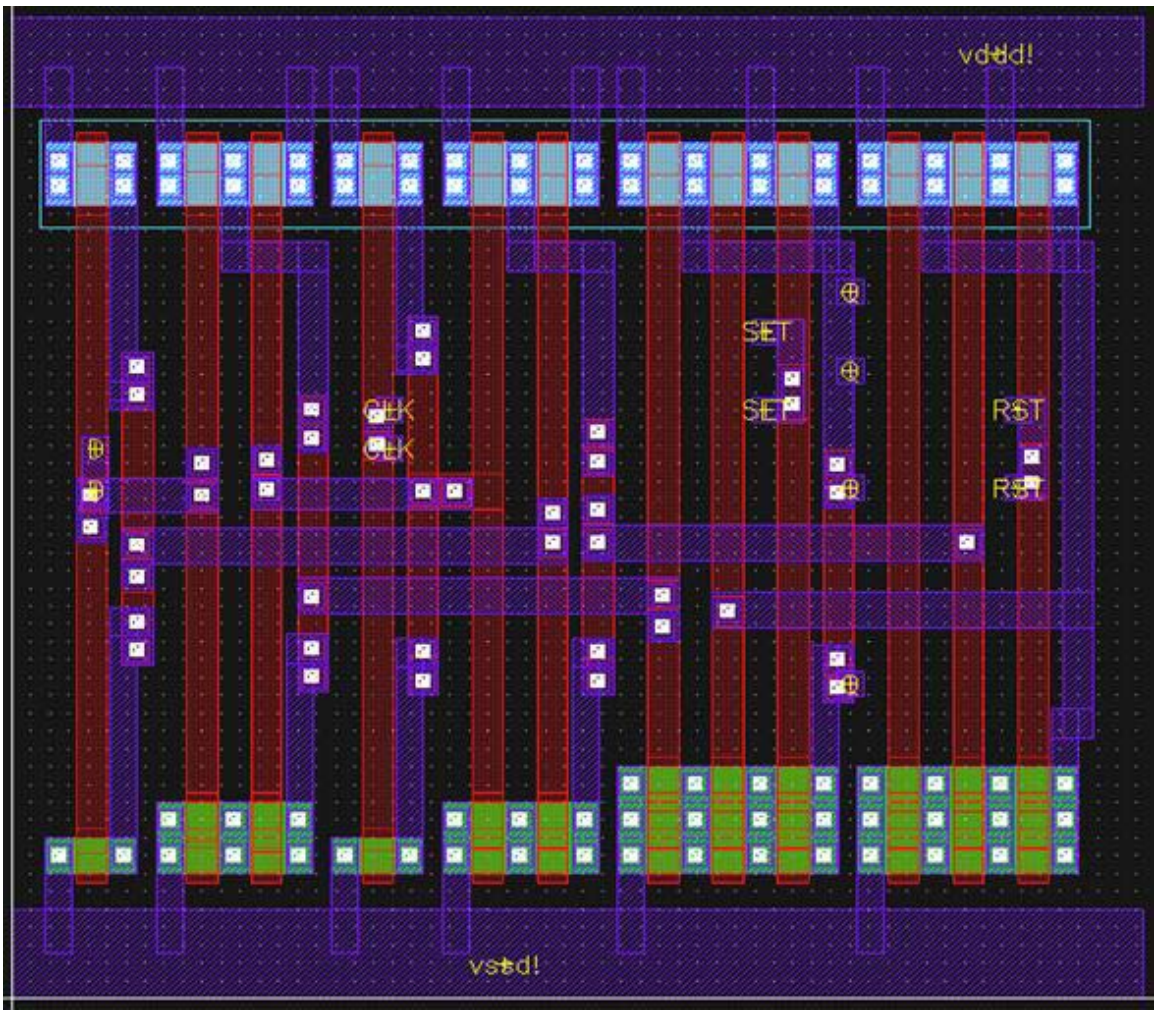

Figure 2.6 Active high D Latch with Async Set and Reset

Some parameters of the cell:

Cell Width = 59.4µm (27 times the pitch)

Cell Height = 55µm (25 times the pitch)

Power rails width = 5µm

Pitch = 2.2µm

Preferred Directions – Metal1 (Horizontal), Metal2 (Vertical).

Note that in figure 2.6 a safety margin of more than 0.5μm is provided on either side of the cell this would avoid any kind of DRC errors and lets the router to abut the cells without wasting space. The width is always made a integer multiple of the pitch so that the continuity of routing grids between any two adjacent cells is guaranteed. Poly jumpers are used where ever possible to avoid the use of metal2 since metal2 is dedicated to routing and any use of it would result in the loss of a routing track. Multiple I/O pins are placed in the cells, this enhances the routing process of the router since it can pick any I/O pin at its convenience and thus route efficiently.

Chapter 3

Timing Characterization

This section starts with the background study of previous works on cell characterization, a literary review of different delay models used by synopsys and a detailed explanation of the linear delay model used to characterize this cell library, with estimation of capacitance and power parameters.

3.1 Background

A number of techniques are available for characterizing the cell library; this work focuses on the simple and accurate characterization method. This section gives a brief explanation on the previous works on the characterization process. In particular, since the CMOS Generic timing model (Linear timing model) used by synopsys tools is largely based on Penfield-Rubinstein-slope model[5], most of the works on timing characterization discussed here will be the ones which focuses on the particular model.

In [2] Jos B. Sulistyo and Dong S. HA used the linear delay model to characterize the cell library. In their work the most commonly used 50%-50% delay is used to estimate the linear model delay parameters.

In [4], a method to characterize cell delay and capacitance parameters is proposed by Patel, and a system described for implementing the method. In this characterization technique he determines the actual switching voltages of the cell, as well as delay definition as switching-to-switching voltage instead of the more commonly used 50%-to-50% delay. While this proposed model is more accurate than the linear model for this type of cell, it is highly inconsistent for cases where a cell drives another cell of different type which results in different switching voltages.

Jou et al. in [7,8] proposed techniques to simplify characterization tables for complicated cells these tables can be 2-D or 3-D tables which are more accurate than the linear model. However, exhaustive calculations make the tool much slower. The proposed techniques are particularly useful for cases where the internal structures of the cells are known.

Further, in [3], a similar method was proposed by Cirit, which slightly differs in that it assumes the cell being characterized as a black box and the internal architecture of the cell is not taken into account. Initially this approach was followed to characterize this cell library due to its simplicity, but later a distinct relationship was observed between the delay and internal structure of the cell which makes the characterization process more accurate.

The cell delay model used in this work is the one described in Chapters I and II of [6]. Specifically, for timing characterization, the CMOS Generic delay model is used due to

its simplicity and relatively small numbers of simulations needed to characterize cells with acceptable accuracy. Also, in [5], several delay models are explained by Eshraghian and Weste, one of which, namely the Penfield-Rubenstein-slope model, appears to be identical with the Generic CMOS delay model used here.

3.2 Literature review on Different Delay Models

3.2.1 CMOS Non-Linear Delay Model

The CMOS nonlinear delay model [6] uses lookup tables and interpolation to compute delays; this needs intelligent software to do mathematical analysis which makes the process slow. Due to its complexity the model can provide close timing correlation with a wide variety of submicron delay modeling schemes.

This model requires a clear understanding of the following:

- The total delay equation
- Cell delay (D_{cell})
- Propagation delay ($D_{propagation}$)
- Transition delay ($D_{transition}$)
- Connect delay ($D_{Connect}$)

3.2.1.1 Total Delay Equation

The delay analysis of this model involves calculating the total delay of a logic gate i.e. the delay between the input pin of a first gate and the input pin of the next gate as shown in Figure 3.1 . This includes four major components: D_{cell}, $D_{propogation}$, $D_{transition}$ and $D_{connect}$.

If the cell delay tables are available for a timing arc, the total delay equation [6] is

$$D_{total} = D_{cell} + D_{connect} \tag{3.1}$$

If the propagation delay tables are available instead, the total delay equation is

$$D_{total} = D_{propagation} + D_{transition} + D_{connect} \tag{3.2}$$

3.2.1.2 Cell Delay

The delay contributed by the gate itself, is typically defined as the 50 percent input pin voltage to 50 percent output voltage [6]. Cell delay is usually a function of both output loading and input transition time. D_{cell} is computed in two ways, depending on the timing data provided,

- Performing table lookup and interpolation in a cell delay table provided in the library.
- Using the propagation and transition tables, following this equation:

$$D_{cell} = D_{propagation} + D_{transition} \tag{3.3}$$

3.2.1.2. Propagation Delay

$D_{propagation}$ is the time from the input transition to completion of a specified percentage (for example, 10 percent) of the output transition. $D_{propagation}$ is often a function of output loading and input transition time. If propagation delay tables are defined for a timing arc, cell delay tables must not be specified. The presence of propagation delay tables indicates that cell delays are computed by adding the propagation and transition delays.

3.2.1.3. Transition Delay

$D_{transition}$ is the time required for an output pin to change state. It is used as a term in the cell delay calculation if propagation tables are specified. After applicable transition degradation, Computing $D_{transition}$ involves performing table lookup and interpolation. $D_{transition}$ is a function of capacitance at the output pin and can also be a function of input transition time in submicron technology.

3.2.1.3. Connect Delay

$D_{connect}$ is the time it takes the voltage at an input pin to charge after the driving output pin has made a transition. This delay is also called the *time-of-flight delay*—the time it takes for a waveform to travel along a wire. The CMOS nonlinear delay model computes connect delay by using the same equations as the generic delay model.

The CMOS non linear delay table is dependent on two parameters as shown in Figure 3.2. The Cell Fall/Rise delay has a non linear variation with Input transition time and output capacitance.

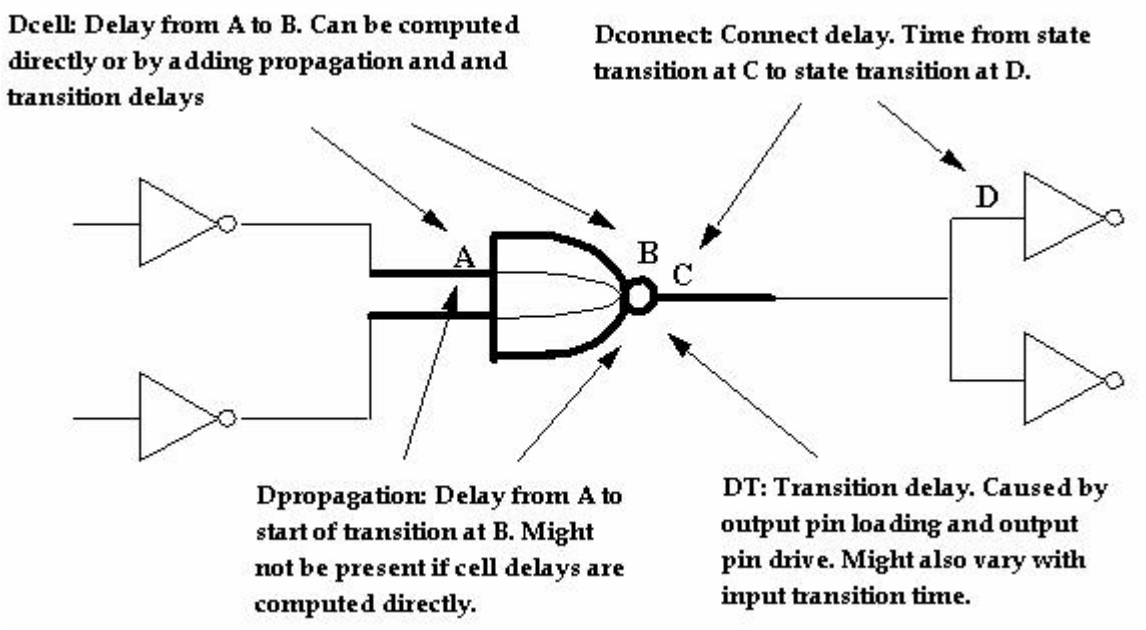

Figure 3.1 Delay Equation Components for CMOS Nonlinear Delay Model [6]

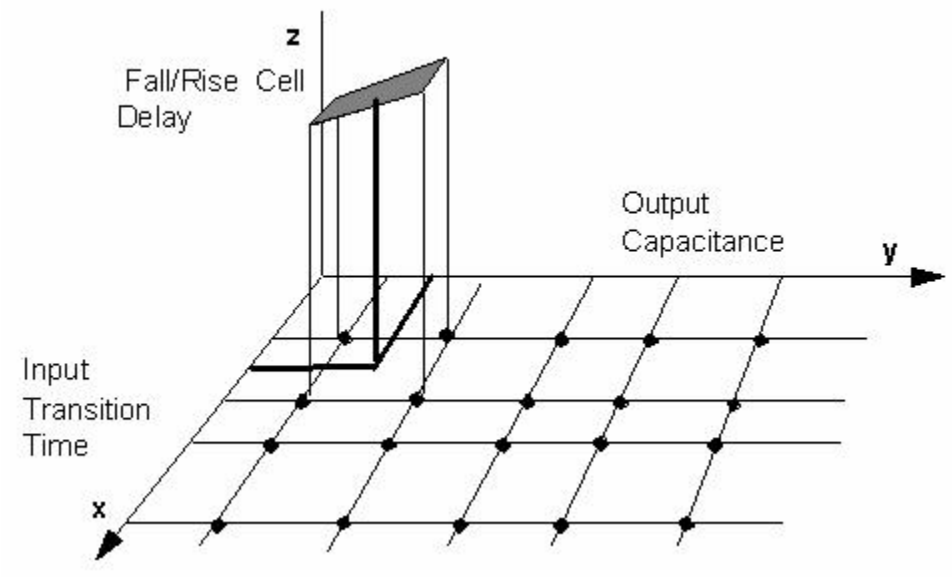

Figure 3.2 Variation of Cell Delay with Input Transition and Output Capacitance [6].

3.2.2 Scalable Polynomial Delay Model

Scalable polynomials provide a smaller and faster alternative to nonlinear lookup tables. However they still need curve-fitting methods to convert the characterization data into computationally efficient polynomial equations with sufficient user-defined accuracy [6]. This makes this model complex. Currently, synthesis tools generate polynomials depending on the variables supported by the nonlinear delay model tables based on input transition time and the output capacitance load. One major disadvantage is that while synthesis tools use this model they do not currently support temperature and voltage effects.

This model needs a clear understanding of the following:

- Polynomial representation
- Model description

Scalable means that the form and order of the polynomials are determined by, or scaled according to the given data. Given a predefined accuracy, it is possible to model almost any delay data with polynomials.

The advantage of using a predefined set of polynomials is that it attempts to fit all cases with accuracy and efficiency using an arbitrary equation at the cost of complexity. However, it allows having a single library instead of multiple libraries for different operating conditions.

The process of library development of scalable polynomial models is similar to that of non linear model developement; a circuit level simulator such as spice is required for characterizing the library. However, an additional step is required which uses curve fitting methods to translate the characterized data into computationally efficient polynomial equations to achieve sufficient user-defined accuracy.

3.2.2.1. Polynomial Representation

A number of numerical computations involve polynomials; the fundamental theory of polynomials typically involves a Taylor series [6], where an analytical function is expressed as a finite series of polynomials.

The complete decomposed polynomial form represents the scalable polynomial delay model syntax. The following example shows two variable functions, but it is easy to extend to the case of more variables. A two-variable polynomial function $D_{x,y}$ can be written as $D(x,y) = P_m(x)Q_n(y)$, where x and y are variables and P_m and Q_n are the mth- and nth-order polynomials, respectively. There are (m+1) (n+1) coefficients for any given m or n, as the follow equation illustrates.

$$P_1 Q_2 = (a_0 + a_1 x_1)(b_0 + b_1 x_2 + b_2 x_2^2) = A_{00} + A_{10} x_1 + A_{01} x_2 + A_{11} x_1 x_2 + A_{02} x_2^2 + A_{12} x_1 x_2^2 \quad (3.4)$$

3.2.2.1. Model Description

The scalable polynomial delay model syntax allows three variable polynomials to be specified. The three dimensions are slew, input load, and output load. The model considers analytical parameters (that is, physical parameters) that affect the delay calculation results to be variables.

For a large data set with abrupt changes, the entire operating region cannot be represented by a single polynomial equation. In this case a piece wise delay model is more helpful to specify breakpoints over the characterization data domains.

The error distribution patterns between the fitted equation and data sets, generally result in the use of two kinds of curve fitting algorithms; Least Square Error (LSR) and Least Square Relative Error (LSRE). The choice of the fitting algorithm depends on the values of characterization and the level of accuracy needed.

3.2.3 Piecewise Linear Delay Model

In the Linear delay model the effect of different wire lengths on a net is overlooked. The equations used by a piecewise linear model to calculate timing delays takes into consideration the delay effects of the actual length of wire has on various components of the total delay equation. The wire is divided into pieces based on the length with each length having a different delay associated with it.

This is achieved by assigning different values for resistance and capacitance values to different lengths of the wires. Hence this model has an overload of assigning resistance and capacitance values for all ranges of wire lengths.

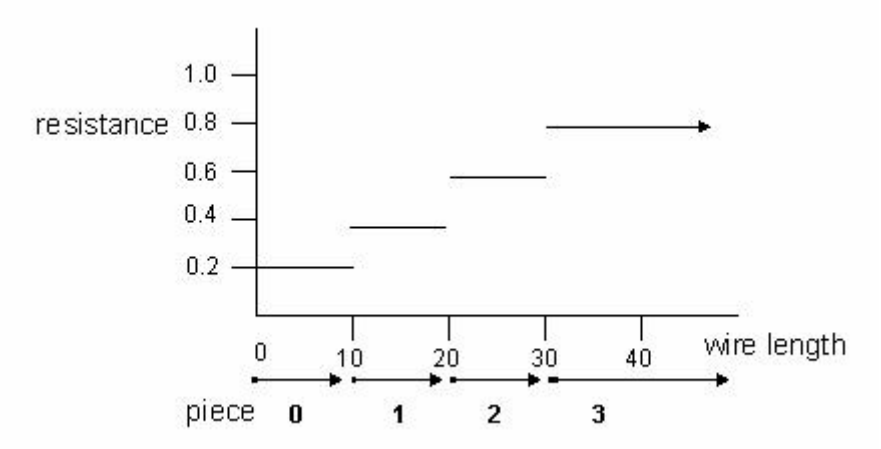

Figure 3.3. Variation of resistance with wire length [6]

The total delay is defined as the delay between the input pin of a gate and the input pin of the next gate, which includes the connect delay (piecewise delay) from the driving pins to the load pins. This delay modeling divides the total delay in a network into four physical components whose sum is the total delay through a circuit element:

$$D_{Total} = D_I + D_S + D_C + D_T \tag{3.5}$$

Where,

D_I - Intrinsic delay inherent in the gate, independent of any particular instantiation and zero load is applied.

D_S - Slope delay caused by the ramp time of the input signal.

D_C - Interconnect Delay i.e. delay due to wire load calculated using piecewise model.

D_T - Transition delay caused by loading of the output pin.

All the above parameters are similar to the CMOS Generic delay model, and will be discussed in detail in section

3.2.4 CMOS2 Delay Model

The CMOS2 delay model uses advanced methods for modeling the effect of input signal ramp times of cell delays. Although a high level of accuracy is achieved by using the CMOS2 delay model to calculate propagation delays, the computational complexity of the model makes it less preferred [6].

The delay model requires a clear understanding of the following:

- Total Delay Equation
- Intrinsic Delay
- Transition Delay
- Edge-rate Delay
- Connect Delay

3.2.4.1. Total Delay Equation

The following equation calculates the total delay for the CMOS2 delay model.

$$D_{cell} = D_I + D_T + D_E \quad (3.6)$$

D_I and D_T are Intrinsic and Transition delay respectively and are the same as in the CMOS Generic delay model. The total delay equation has another term called Edge-Rate Delay(D_E) which requires a detail explanation..

3.2.4.2. Edge-Rate Delay

The incremental delay caused by the input edge rate is modeled as the Edge-Rate Delay. The edge-rate delay calculations in the CMOS2 model allows for a definition of the nonlinear variation of the delay caused by the input edge rate. Edge-rate delay is modeled as a two-piece, piecewise linear incremental delay. The required calculations draw on a set of parameters that characterize the edge rate of an input signal to the cell.

3.2.5 CMOS Generic delay Model or Linear Delay Model

This delay model is mathematically the simplest form of delay model which allows rapid characterization with modest number of simulations, while still achieving an acceptable accuracy [2]. Due to its simplicity it is widely used for characterization of cell libraries.

In this model, transition delay is modeled to be linearly proportional to load capacitance, while slope delay is modeled as linearly proportional to the transition delay of the driver. The linear delay model divides the total delay of a network into four physical components whose sum is the total delay through a circuit element:

$$D_{total} = D_{Intrinsic} + D_{Transition} + D_{Slope} + D_{Connect} \quad (3.7)$$

3.2.5.1 Intrinsic Delay

The **Intrinsic delay** ($D_{Intrinsic}$) of a cell is defined as the propagation delay of the cell without any load applied to its output, when it is driven by another identical load less cell. It is also called the fixed (or zero load) delay from the input pin to the output pin of a circuit element [2]. This means that both the driving and driven cell should be load less. It is not practical to measure such a delay as this is a hypothetical condition. However, it is possible by means of simulation to measure it. This means that the driver cell must drive the driven cell indirectly as shown in Figure 3.4.

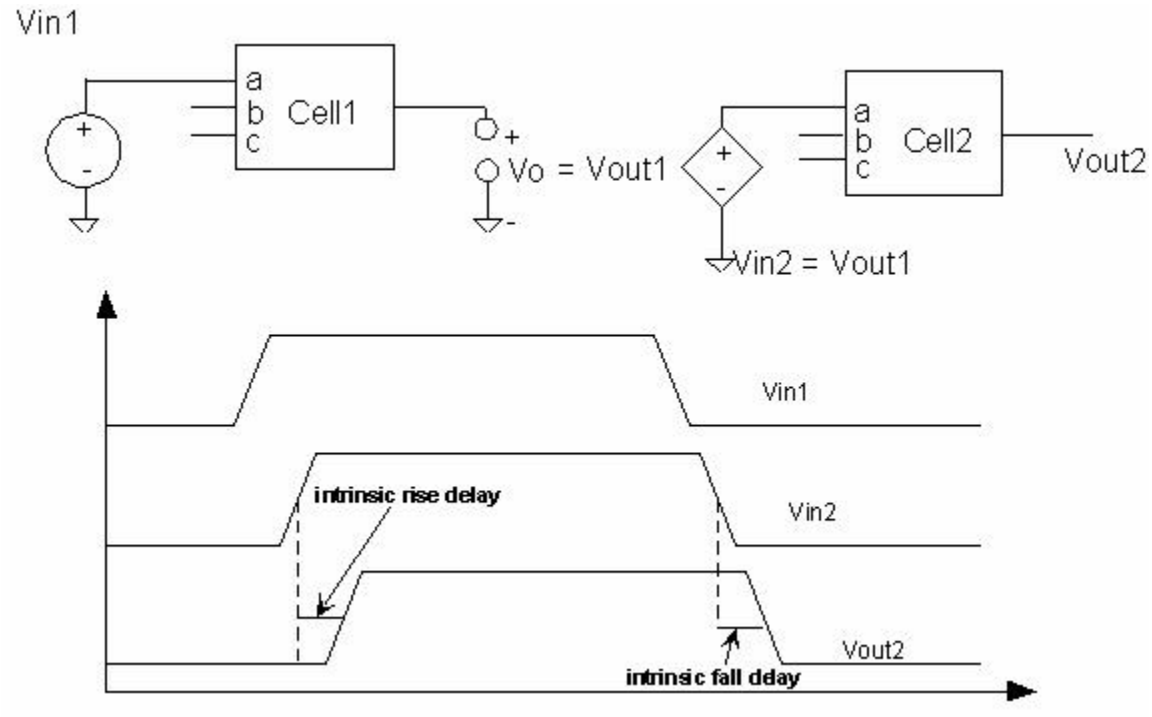

Figure 3.4 Intrinsic delay definition [2]

The output of Cell1 (Vout1) drives the input of Cell2 (Vin2) indirectly. The intrinsic rise delay may differ from the intrinsic fall delay and depends on the internal architecture of the cell and each input may have different intrinsic delays, do there can be six different delays from the above circuit with three inputs there are three different intrinsic rise delays and three different intrinsic fall delays.

3.2.5.2 Transition delay

The **Transition delay** ($D_{Transition}$) is defined as the additional delay of a cell in the loaded condition, but driven by another identical load less cell, it is the time it takes the output of the loaded cell to change state [2]. The transition time of the output pin on a net is a function of the capacitance of all pins on the net and the capacitance of the interconnect network that ties the pins together.

The equation for transition delay can be modeled as:

$$D_T = R_{Cell} (C_{wire} + C_{pins}) \tag{3.8}$$

This equation calculates the rise and fall delays of the cell. This delay is caused as the result of a cell having to drive a capacitive load causing the output to transition less steeply compared to the load less case as shown in Figure 3.5 and also increases delay with the load.

Transition delay is of two types, the rising transition delay is given by,

$$D_{Transition_{rising}} = R_{Cell_{rise}} \times C_{Load(pin+wire)} \tag{3.9}$$

The falling transition delay is give by,

$$D_{Transition_{falling}} = R_{Cell_{fall}} \times C_{Load(pin+wire)} \tag{3.10}$$

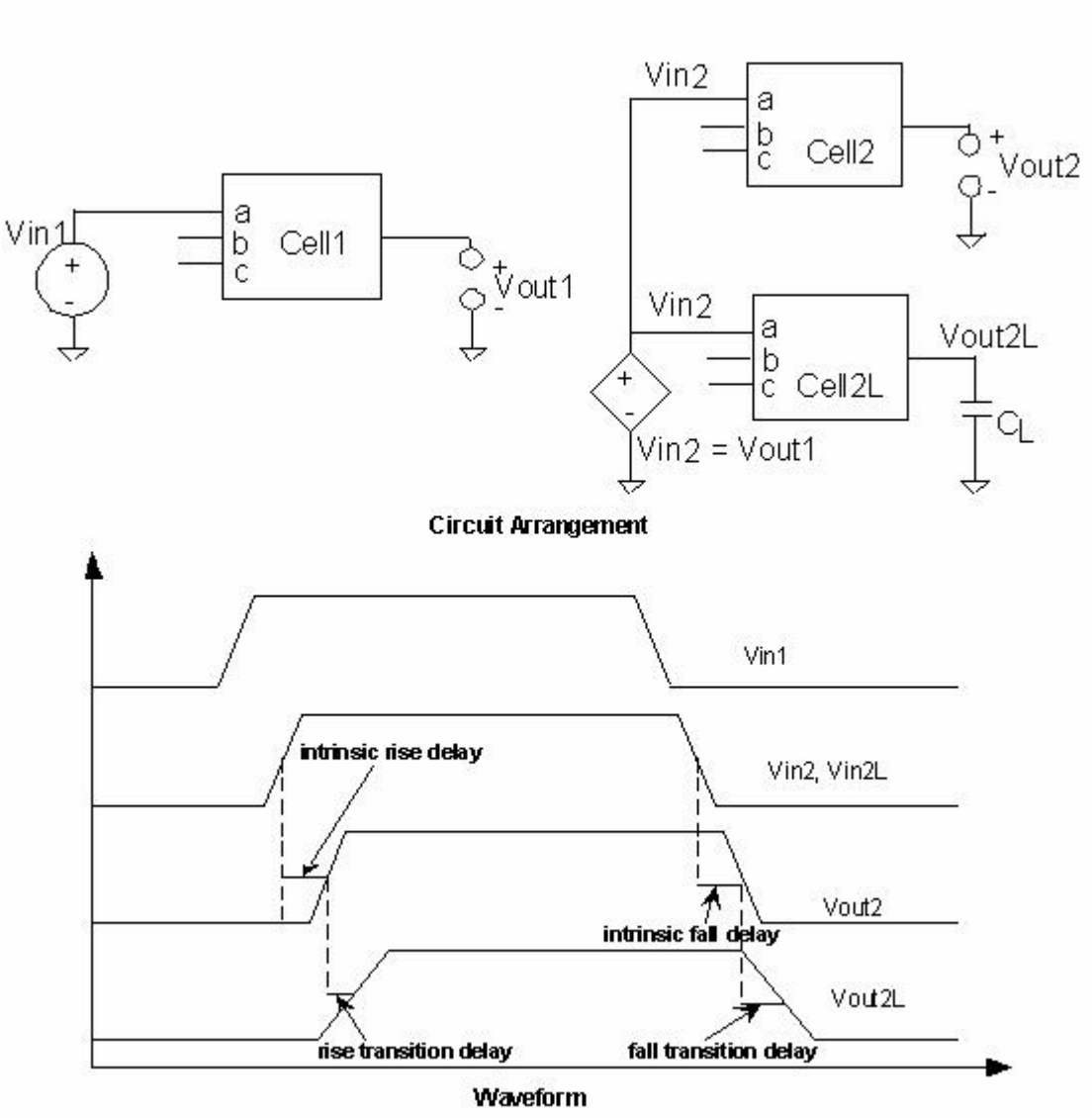

Figure 3.5 Transition delay definition [2]

3.2.5.3 Slope delay

The Slope delay (D_{Slope}) is an incremental time delay caused by slowly changing input signals; this is due to the transition delay of the driver, when it directly drives the driven cell, due to the input capacitance of the driven cell.

This transition delay causes the output slope of the driving cell to be less steep than the one without load capacitance. This slow ramp signal causes an additional delay at the output of the driven cell.

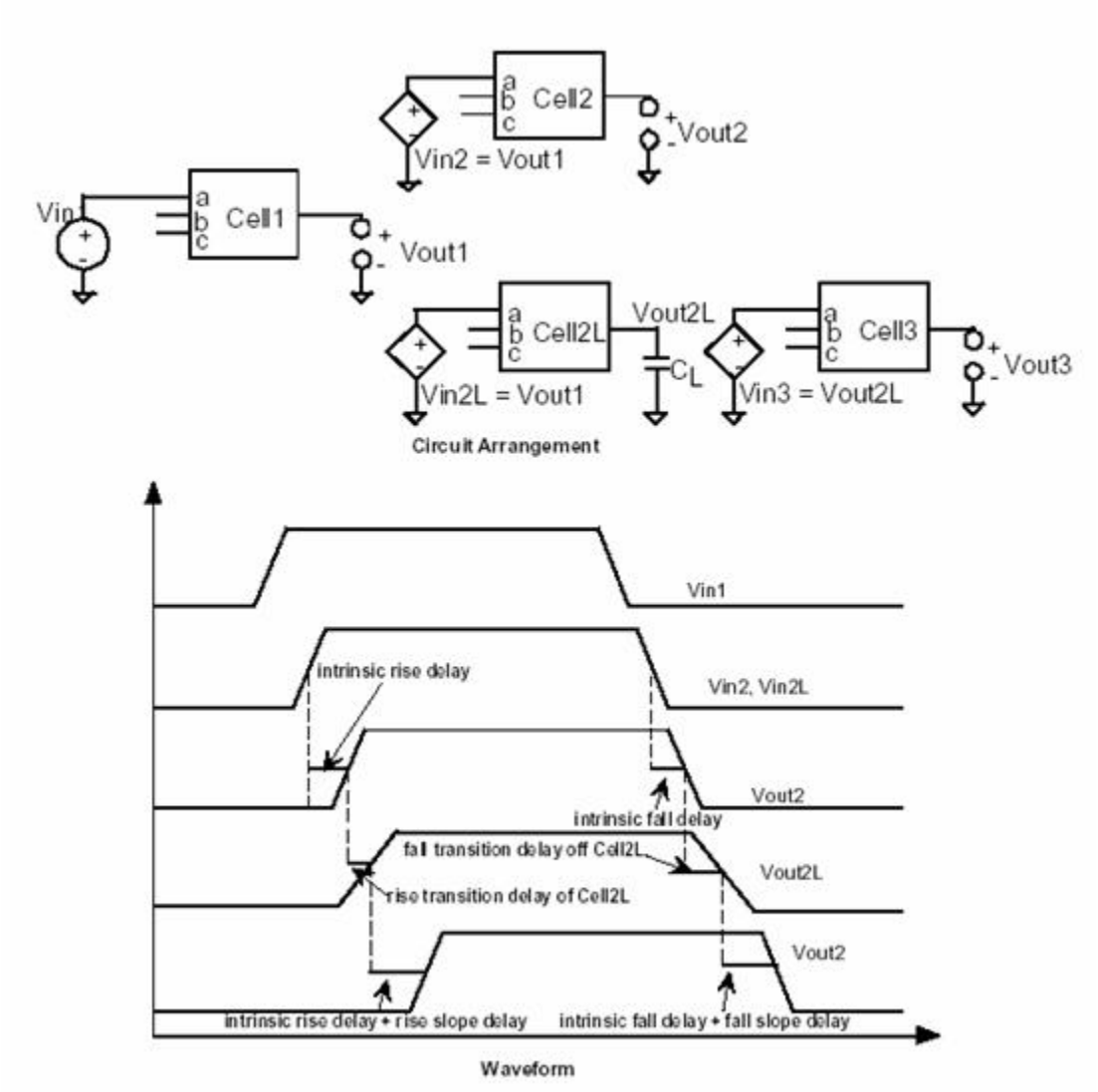

Figure 3.6 Transition delay definition [2]

This delay is a strong function of ramp time, in some technologies this delay does not vary over a wide range of ramp values. Slope delay is included in the delay equation to

allow additional accuracy in the modeling of technologies that are sensitive to input ramp time.

The slope delay is calculated as the product of the transition delay of the previous stage and the slope sensitivity of the cell.

$$D_{Slope} = S_S \times D_{Transition(\mathrm{Pr}ev)} \tag{3.11}$$

Where, S_S is Slope Sensitivity factor accounting for the time at which the input begins to rise but has not reached the threshold level at which the channel begins to conduct

$D_{Transition(\mathrm{Pr}ev)}$ is the transition delay calculated at the driver output pin.

For an inverter the Slope delay is of two types, the rising slope delay is given by

$$D_{Slope_{rising}} = S_{S_{fall}} \times D_{Transition_{fall(input)}} \tag{3.12}$$

The falling slope delay is given by,

$$D_{Slope_{falling}} = S_{S_{rise}} \times D_{Transition_{rise(input)}} \tag{3.13}$$

3.2.5.4 Connect delay

The Connect delay ($D_{Connect}$) of an element is the time it takes the voltage at an input pin to charge after the driving output pin has made a transition. This delay is also known as time-of-flight delay, which is the time it takes a waveform to travel along a wire. The connect delay can be modeled as three types. See Figure 3.7.

3.2.5.4.1 Best Case Tree Model

In this model the load pin is physically adjacent to the driver. All the wire capacitance isincurred, but none of the wire resistance must be overcome. Wire resistance is assumed negligible. The best-case connect delay is calculated from the following equation. Because R_{wire} is always zero in this case, the resulting DC is always zero.

$$D_{Connect_{best}} = R_{wire}(C_{wire} + C_{pin}) = 0 \tag{3.14}$$

3.2.5.4.2 Worst Case Tree Model

In this model the load pin is at the extreme end of the driver. Each load pin incurs both the full wire capacitance and the full wire resistance, as shown in the following equation.

$$D_{Connect_{worst}} = R_{wire}(C_{wire} + \sum_{pins} C_{pin}) \tag{3.15}$$

3.2.5.4.3 Balanced Case Tree Model

In this model all the load pins are on separate and equal branches of the interconnect wire. In the Balanced case, each load pin incurs an equal portion of the wire capacitance and wire resistance, as shown in the following equation.

$$D_{Connect_{balanced}} = \frac{R_{wire}}{N}\left(\frac{C_{wire}}{N} + C_{pin}\right) \tag{3.16}$$

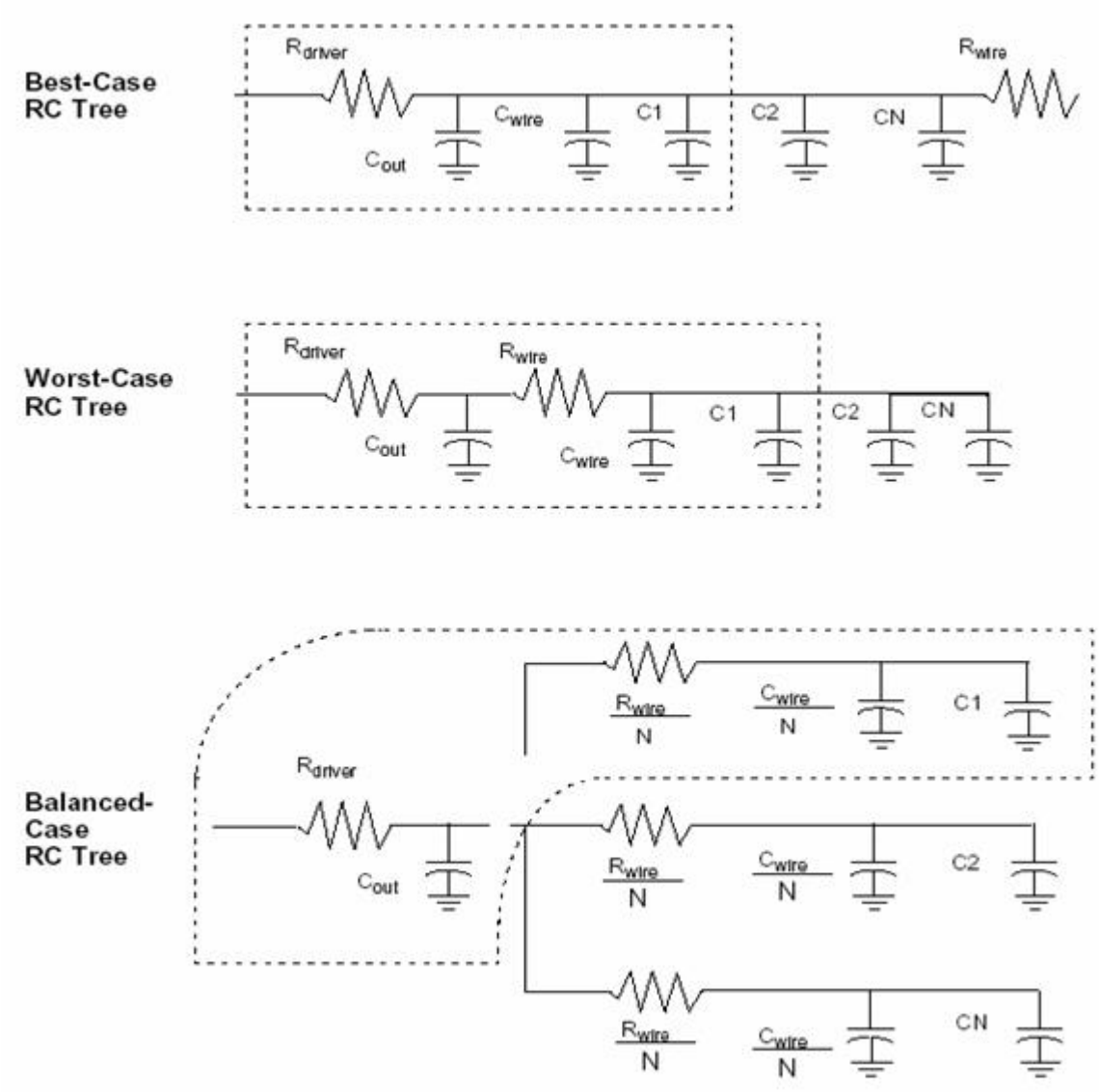

Figure 3.7 Modeling of Connect delay [6]

3.2.5.5 Interconnect delay

Interconnect delay is defined as the delay caused by the connect delay and transition delay as a result of the number of fan-outs. Interconnect delay can be described by the following equation

$$D_{Interconnect} = D_{Transition} + D_{Connect} \tag{3.17}$$

Then, the total delay equation can be reduced as

$$D_{total} = D_{Intrinsic} + D_{Interconnect} + D_{Slope} \tag{3.18}$$

The linear delay model parameters for a 2-input Nand gate is as shown in figure 3.8

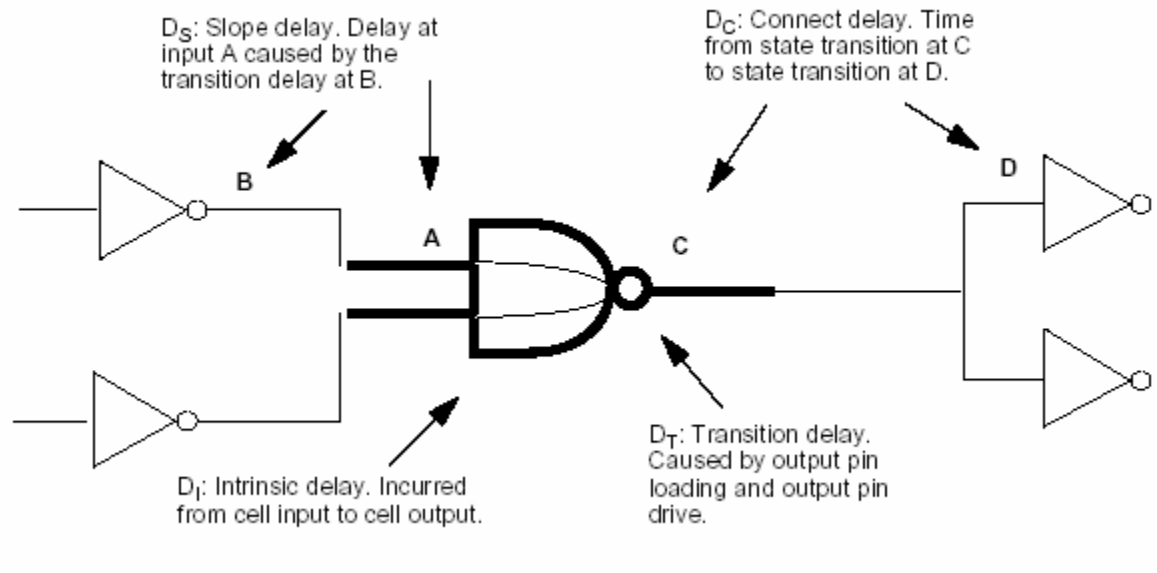

Figure 3.8 Linear delay model parameters for 2-input NAND gate [6]

3.3 Estimation of Linear Delay Model Parameters

Linear model parameters can be extracted using SPICE simulations, the parameters that need to be extracted are intrinsic delay, cell resistance (Rise/Fall), input capacitance and slope delay. The cell resistance refers to some stipulated linear factor. It does not refer to the resistance that is used in conventional circuits. Normally resistance is defined as the derivative change in current with respect to change in voltage applied at a node as shown in figure 3.9(a).

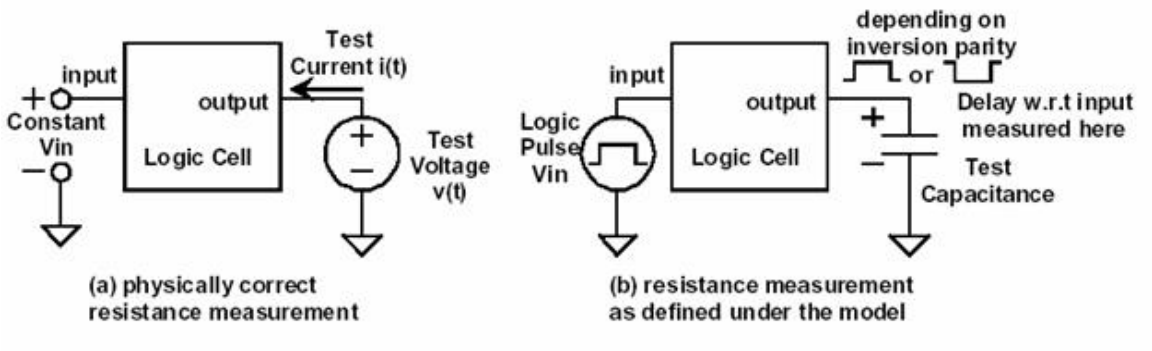

Figure 3.9 Estimation of Resistance [2]

The resistance in figure 3.9(a) the output resistance is calculated as

$$R_{out} = \frac{d(I_{test})}{d(V_{test})}\Big|_{V_{in}=\text{Constant}} \tag{3.19}$$

This is the actual small signal resistance of the circuit, and as such is very nonlinear in large signal circuits, In this model the resistance is calculated as a function of delay and load capacitance.

$$R_{out} = \frac{\Delta(delay)}{\Delta(\text{capacitance})} \tag{3.20}$$

This output resistance can vary for rising delays and falling delays, it can also vary depending which input pin is triggered and how many input pins are triggered, and here we have always accounted for worst case delays while estimating output resistance.

The output resistance is used to calculate the transition delay of the cell by synthesis tools. Though the value of physical definition resistance of a pin stipulates it to be an unique value the value of output resistance has many non unique values whose value depends on which pin triggers the output transition.

3.3.1 Linear delay parameters Estimation for Combinational Logic

3.3.1.1 Intrinsic delay Measurement

As mentioned previously, the intrinsic delay of a cell is defined as the propagation delay of the cell without any load applied to its output, when it is driven by another identical load less cell. This delay can be found from the measurement setup as below

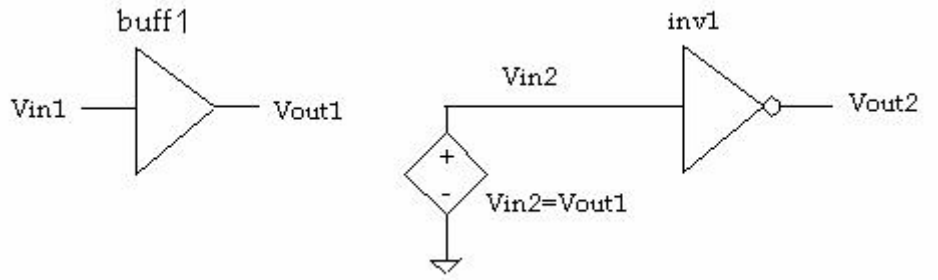

Figure 3.10 Intrinsic delay measurement

The intrinsic delay is the inherent delay of the circuit and has no dependence on the load of the circuit, this can be measured by simulations but practically it is not possible to measure this directly. However, intrinsic delay can be calculated solving numerically. The cell buff1 of Figure 3.10 is a buffer cell and is used as waveform shaper, it does not drive the inverter directly but is made to indirectly drive the inverter, adding this buffer makes the input wave form less steep and would make it a more typical waveform with zero transition delay.

The waveform in Figure 3.11 shows the measurement of intrinsic delay, the input is passed through an isolation buffer and then fed to then inverter cell.

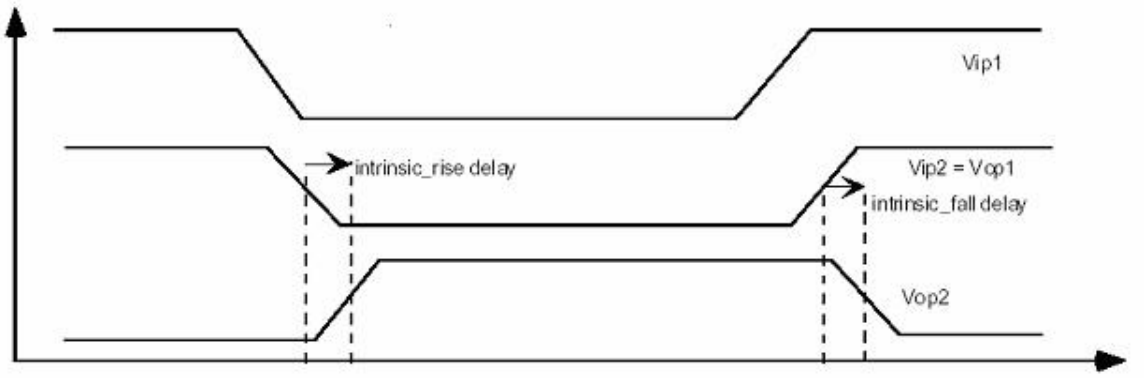

Figure 3.11 Intrinsic delay waveform

The intrinsic delay is measured for both rising and falling waveforms to determine the intrinsic_rise and intrinsic_fall values, all delays are measured as 50%-50% transition delays [2].

3.3.1.2 Transition delay and Output resistance Measurement

The transition delay of the circuit is defined as the delay of a cell under loaded condition, but driven by another identical load less cell, it is the time it takes the output of the loaded cell to change state. Transition delay is calculated as the product of output resistance and load capacitance.

Mathematically

$$D_{transition} = R_{output} \times C_{load} \tag{3.21}$$

Where

$$\mathrm{R_{Output}} = \frac{\text{Intrinsic Delay-Loaded Delay}}{\text{Load Capacitance}} \tag{3.22}$$

$$C_{load} = C_{pin} + C_{wire} \tag{3.23}$$

The setup for measuring output resistance is as shown in figure 3.12. The input is fed to the buffer for wave shaping and then coupled to both an inverter without load to find the Intrinsic delay and to an inverter with varying load to determine loaded delay.

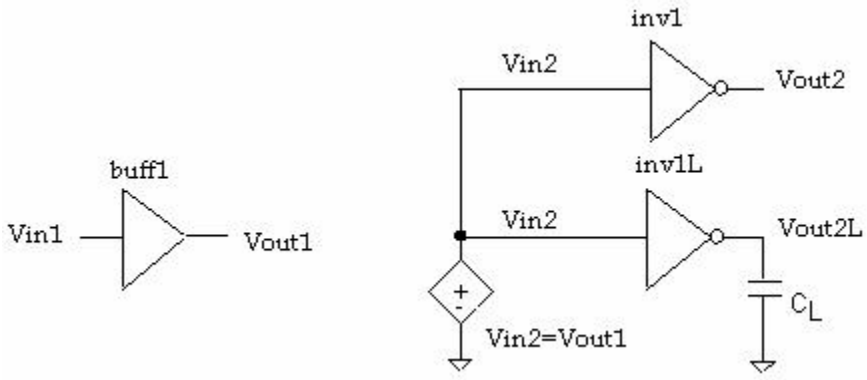

Figure 3.12 Transition delay measurement

The load C_L is varied from 1X to 4X load and the delay is calculated adding the difference in delays from previous loaded condition, this gives the Δ change in delay. The output resistance is calculated by dividing this change in delay by the change in load i.e. 4X load. The waveform for measuring t_{phl} and t_{plh} is shown in figure 3.13

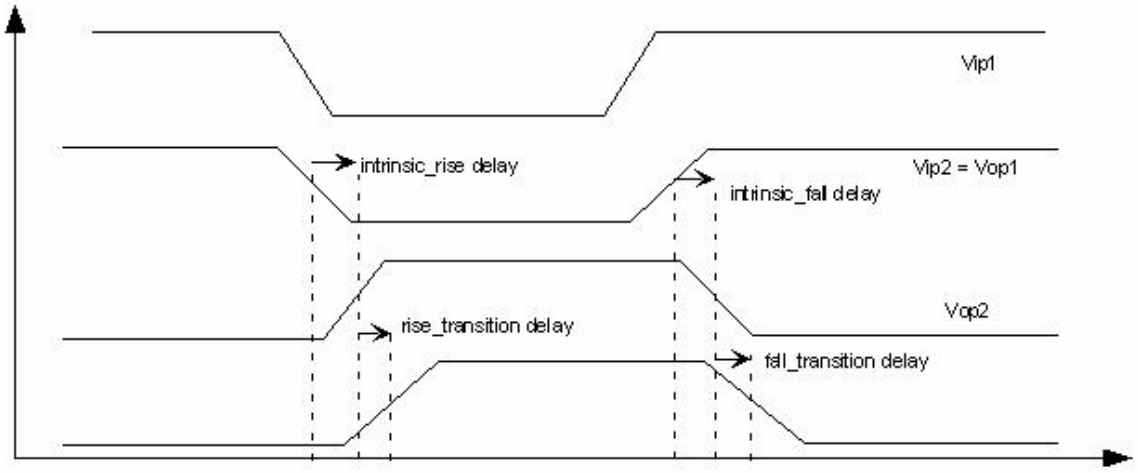

Figure 3.13 Transition delay waveform

3.3.1.3 Slope delay Measurement

As explained previously slope delay is an incremental time delay caused by slowly changing input signals; this is due to the transition delay of driver, when it directly drives the driven cell, due to the input capacitance of the driven cell. The delay is an additional delay due to the varying input ramp signals at the input pin of the cell. The setup for measurement of slope delay is as shown in figure 3.14.

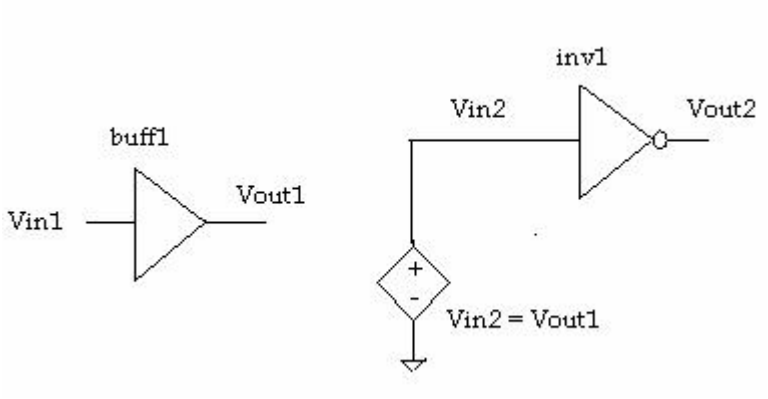

Figure 3.14 Slope delay measurement

The input ramp is changed and the buffer is used to provide the necessary waveform shaping for the inverter. Due to the change in input ramp an additional delay is added to the output Vout2 which accounts for the slope delay. First the intrinsic delay of the inverter inv1 is calculated using Vin1 as ideal slope, the output Vout1 adds a transition delay which is applied to the input of inverter inv1, then the input ramp is changed and the difference in intrinsic delay of inverter inv1 is calculated, this process is repeated for different ramp values and the slope delay in each case is calculated and their average gives the slope delay of the inverter. This is termed as slope sensitivity in the synopsys library.

There is rise slope sensitivity for both rising and falling edges termed as slope_rise and slope_fall respectively. The slop delay waveform is as shown in figure 3.15. Vout2-1 and Vout2-2 are obtained by varying the input slope of Vin1.

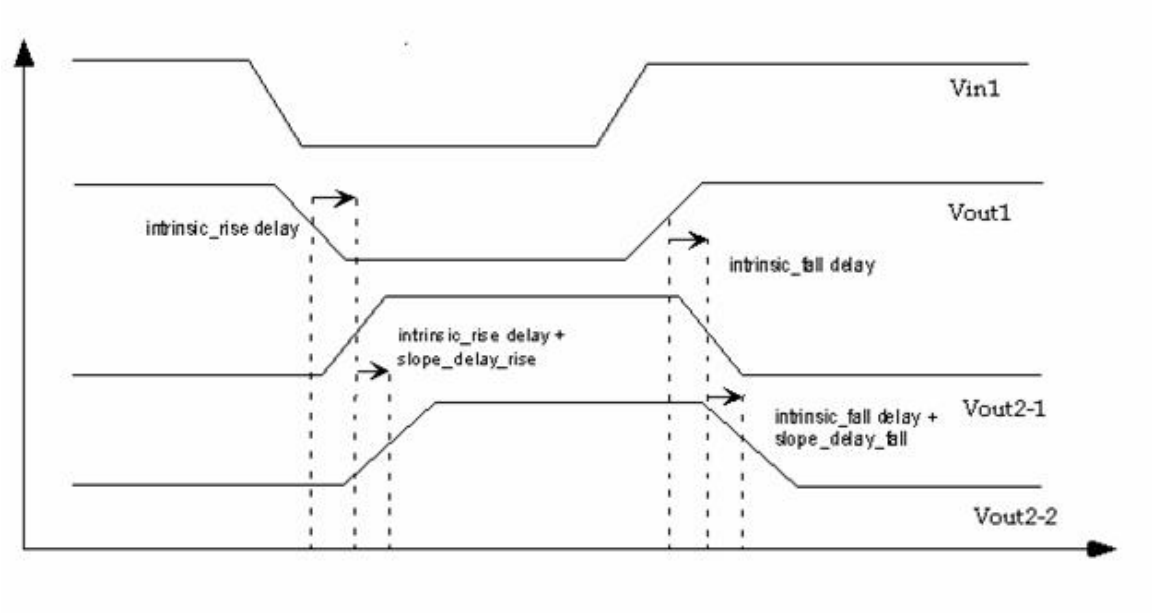

Figure 3.15 Slope delay waveform

3.3.2 Linear delay parameters Estimation for Sequential Logic.

Sequential cells have to be characterized with additional information in order to model them accurately. In addition to delay, the cells have to be characterized for

1. Setup and Hold time.
2. Recovery and Removal time.
3. Minimum and Maximum pulse widths.

3.3.2.1 Estimation of Setup and Hold Time

The setup time is generally defined as the minimum time allowed between the arrival of the data and the transition of the clock signal, so that the output signal will reach the expected logical value within a specific delay. If the data arrives later than the setup time an incorrect value is latched at the output [18]. This specific delay is introduced so that the setup does not degrade the clock to Q propagation time more than a pre determined tolerance value. This value must be modeled with reasonable accuracy in order for proper latching of data. The waveform in figure 3.16 shows data arriving times and setup requirements for active high flip-flop.

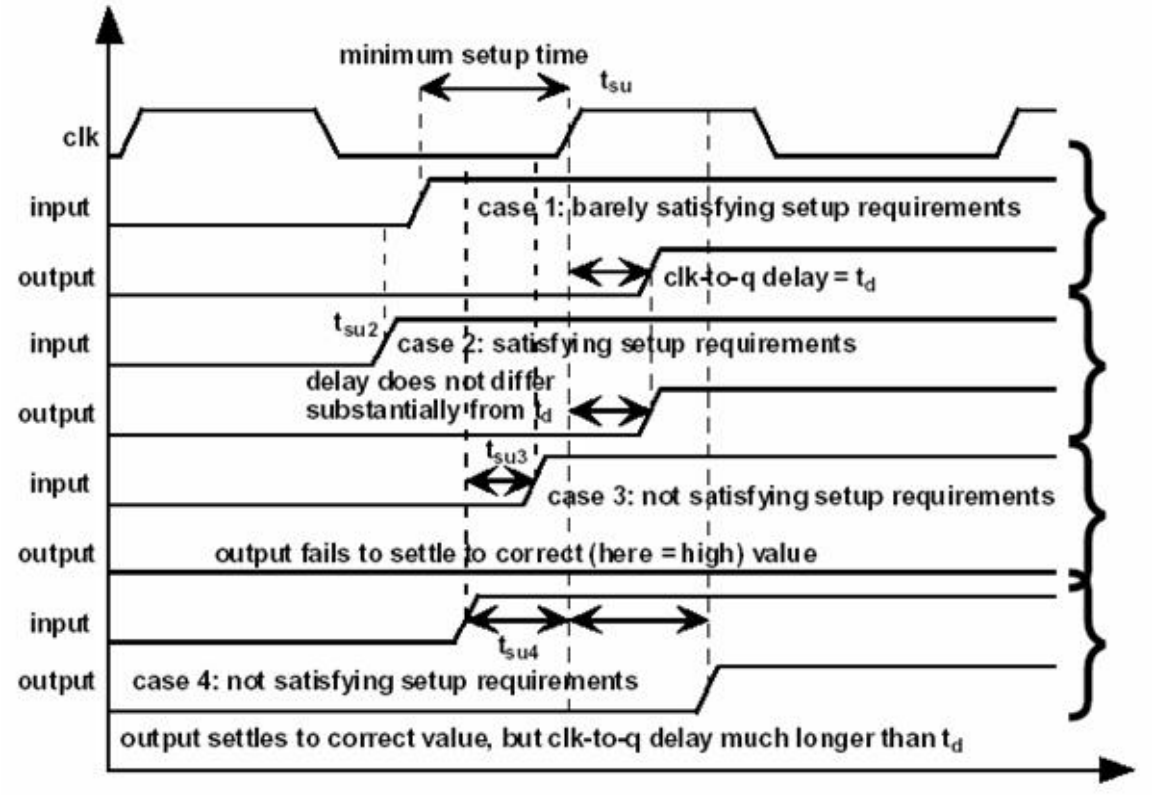

Figure 3.16 Setup time measurement for positive edge Flip flop [2]

In order to measure the setup time an iterative process is required in which an initial guesses of arrival time for the data is taken such that it satisfies the setup time requirement and the delay from clock-to-Q is calculated. The arrival time of the data with respect to the clock is then decreased until the output reaches 90% of its original value, the time difference between the arrival time of the data and the clock gives the setup time of the flip-flop. A safety margin i.e. of 20% is then added to the setup time. The typical waveforms are shown in figure 3.16 where the data arrival times and the setup requirements for active high flip-flop are demonstrated.

The hold time is generally defined as the minimum time allowed between the transition of the clock and the latching of data such that the output still maintains the expected logic level. If the data changes before the hold time an incorrect value is latched at the output [18].

Hold time measurement is similar to setup time, again it is an iterative process in which an initial latch up time for the data is guessed at such that it satisfies the hold time requirement and the delay from clock-to-Q calculated. The transition time of the data with respect to the clock is then decreased until the output reaches 90% of its original value, the time difference between the clock and the data gives the hold time of the flip-flop, again a safety margin e.g. 20% is added to the hold time. The waveform in figure 3.17 shows the data transition times and the hold requirements for active high flip-flop.

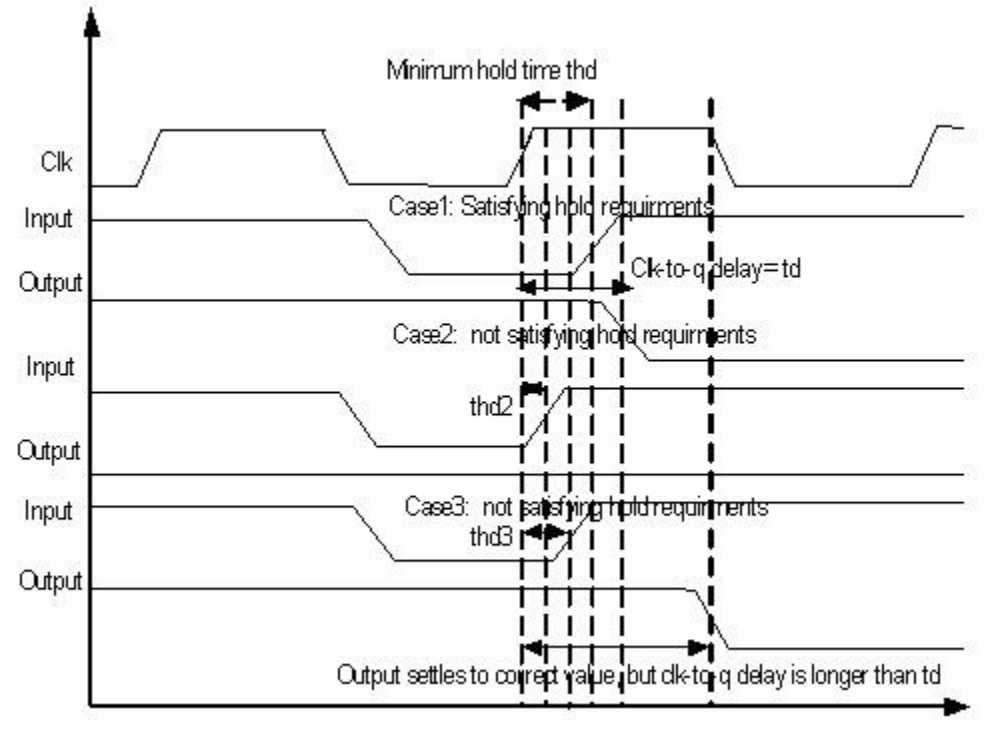

Figure 3.17 Setup time measurement for positive edge Flip flop

3.3.2.2 Estimation of Recovery and Removal Time

The recovery time is defined as the minimum allowable time between the control pin transition from active to the inactive state and the active edge of the synchronous clock signal [18]. Like the setup time a safety margin is added into the condition of recovery time. The waveform in figure 3.18 shows the recovery and removal times for active high flip-flop.

Removal time is defined as the minimum allowable time between the active edges of the synchronous clock pin while the asynchronous control pin is in transition from active to inactive state [18]. Similar to recovery time a safety margin is added to the removal time

value. Measurement of both recovery and removal times are similar to setup and hold time measurements in their use an iterative process to determine timing.

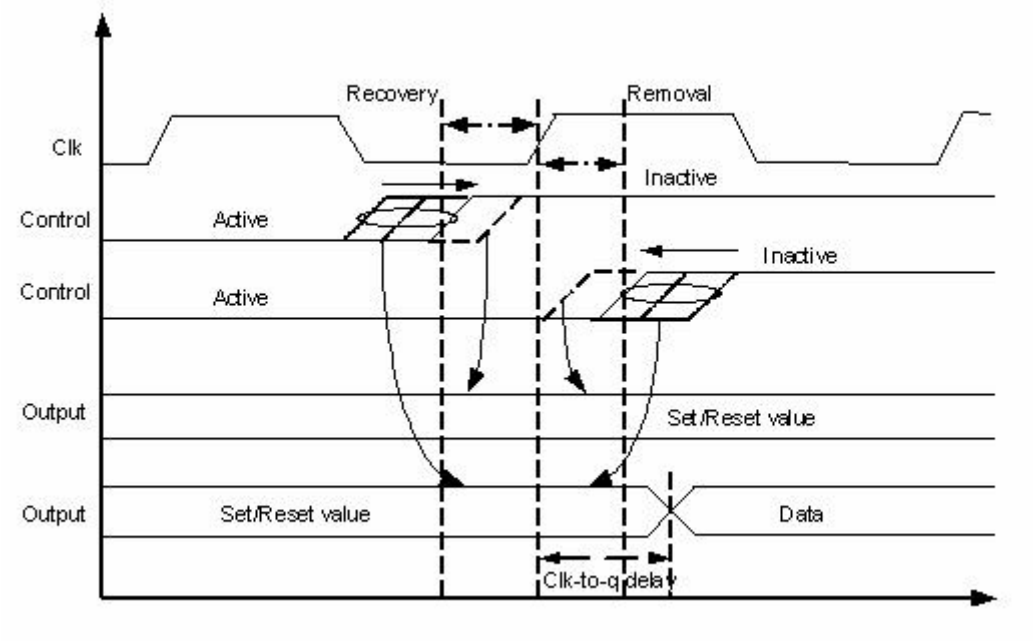

Figure 3.17 Recovery and Removal time measurement for positive edge Flip flop

The only difference between measuring recovery and removal time of a level sensitive cell to an edge triggered sequential cell is clock transition. For a level sensitive cell clock transition to inactive is the required stimulus condition whereas for an edge triggered sequential cell clock transition to active is the required stimulus condition

3.3.2.3 Estimation of Minimum and Maximum pulse widths

The pulse widths that have to be considered are Clock, Set and Reset. Static CMOS circuits can hold their output indefinitely in the absence of clock hence have no maximum clock width requirements, the minimum clock width depends on the gate

delays of the logic and hence determined by the critical paths of the circuit, which usually far exceed the minimum clock period achievable. These can be measured by decreasing the clock period until the flip-flop fails to produce a desired output. Similarly the minimum SET/RESET pulse width can be calculated by decreasing the pulse width until the flip-flop fails to SET/RESET the output. Generally these signals are used with a generous amount of time; hence their estimation need not be characterized accurately. Since Master-Slave architectures for the flip-flops are used in this library the hold times tend to be shorter or even negative, meaning that the flip-flops or latches can latch data even if the data transition occurs before the clock arrives, hence setup time measurement is critical in these circuits.

3.4 Estimation of Capacitance and Area

The input capacitance associated with each pin is the output load for the previous cell. When a gate level power synthesis tool calculates switching power or a synthesis timing analysis tool calculates the delay input capacitance is required. Input capacitance is determined by the length and width of the transistors and can also be obtained from the extracted netlist of the layout.

The capacitance can be estimated from the following

$$C_{Pin} = C_{ox} \times W \times L \tag{3.24}$$

Where

$C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}}, C_{ox}$ is the gate oxide capacitance, ε_{ox} is the dielectric constant of silicon dioxide and t_{ox} is the gate oxide thickness. W is the width of the transistor and L is the length of the transistor.

The area of the cell can be calculated from the layout of each cell and can be used by the synthesis tool to give a rough estimate of the total area of the chip excluding the routing area.

3.5 Estimation of Power dissipation

Power dissipation can be modeled in terms of 3 parameters leakage, short-circuit and dynamic power [9].

$$P_{dissipated} = P_{leakage} + P_{sc} + P_{switching} \tag{3.25}$$

Leakage power is the power dissipated by the cell when it is in stable condition i.e. there is no signal transition at the inputs or outputs of the cell [10]. Leakage power usually occurs when current carriers diffuse between the diffusion layers and substrate and this current is modeled as the sub threshold leakage per micron obtained from sub-threshold behavior of the transistors.

The sub threshold current can be expressed based on the following [20]

$$I_{ds} = \mu_0 C_{ox} \frac{W}{L}(m-1)(v_T)^2 \times e^{(V_g - V_{th})/mv_T} \times (1 - e^{-v_{DS}/v_T}) \tag{3.26}$$

Where

$$m = 1 + \frac{C_{dm}}{C_{ox}} = 1 + \frac{\frac{\varepsilon_{si}}{W_{dm}}}{\frac{\varepsilon_{ox}}{t_{ox}}} = 1 + \frac{3t_{ox}}{W_{dm}} \tag{3.27}$$

Where V_{th} is the threshold voltage, and $v_T = KT/q$ is the thermal voltage, C_{ox} is the gate oxide capacitance; μ_0 is the zero bias mobility and m is the sub threshold swing coefficient. W_{dm} is the maximum depletion layer width (T_{SI}) as in case of SOS and t_{ox} is the gate oxide thickness. C_{dm} is the capacitance of depletion layer. The power is given by leakage is

$$P_{leakage} = I_{leakage} \times V_{\sup ply} \tag{3.28}$$

The short circuit power is due to a very small current that flows when the PMOS and NMOS transistors are switching simultaneously which results in a short-circuit path from supply to ground, this current flows for a very small period of time.

Since this current is for a very small amount of time it is very negligible.

$$P_{sc} = I_{sc} \times V_{\sup ply} \tag{3.29}$$

The most dominant term is the switching power, this result from charging/discharging of load capacitance. This can be calculated from the following expression.

$$P_{sw} = \frac{1}{T}\int_0^T v(t).i(t)dt \tag{3.30}$$

Considering that during the first half of the clock cycle the load capacitor discharges through the NMOS transistor and the second half cycle the load capacitor charges through the PMOS transistor.

$$P_{sw} = \frac{1}{T}\left[\int_0^{\frac{T}{2}} V_{out}\left(-C_{load}\frac{dV_{out}}{dt}\right)dt + \int_{\frac{T}{2}}^{T}(V_{dd} - V_{out})\left(C_{load}\frac{dV_{out}}{dt}\right)dt\right] \tag{3.31}$$

Evaluating the integrals in (3.31), we get

$$P_{sw} = \frac{1}{T}\left[\left(-C_{load}\frac{{V_{out}}^2}{2}\right)\Bigg|_0^{\frac{T}{2}} + \left(V_{dd}V_{out}C_{load} - \frac{1}{2}C_{load}{V_{out}}^2\right)\Bigg|_{\frac{T}{2}}^{T}\right] \tag{3.32}$$

Applying the limits in (3.32), we get

$$P_{sw} = \frac{1}{T}C_{load}{V_{dd}}^2 \tag{3.33}$$

Since $f = \frac{1}{T}$, we can rewrite the expression as

$$P_{sw} = C_{load}.{V_{dd}}^2.f \tag{3.34}$$

Hence the switching power for the entire chip can be estimated as

$$P_{sw} = {V_{dd}}^2.f.\sum_i C_i \tag{3.35}$$

Where,

C_i is the pin capacitance of each gate which is the load capacitance for previous stage. However, the tricky part is not knowing the fraction of active gates for a particular application.

Chapter 4

Cell Library Validation

This section of the thesis discusses the procedures followed to validate the cell library. It starts with the observations made during simulations i.e. the relationship between delay and internal architecture of the cells followed by the circuits designed to validate the cell library.

4.1 Observation:

In the method proposed by Cirit [3], the cell is being characterized as a black box and the internal architecture of the cell is not taken into account. This is an easy method to estimate the timing of the circuit but does not give us an insight regarding the behavior of the cell with changing architecture. Though the Cirit model was followed to characterize the cell library, due to its simplicity, the simulations revealed a distinct relationship between intrinsic delay and the internal architecture of the cell. Specifically it was observed that the intrinsic delay fits a second order polynomial equation, increasing with the number of PMOS or NMOS either in series or parallel and the total delay varies linearly with increasing load.

4.2 Delay Estimation:

The circuit delay is due to the addition of all the capacitance nodes and resistances involved in switching. Any logic function can be synthesized from circuits which have NAND or NOR gates. First we will analyze NAND gate delays followed by NOR gates. The total fall delay for any input can be calculated using Elmore's delay model:

$$t_{df} = \sum (R_{pulldown-path} \times C_{pulldown}) \tag{4.1}$$

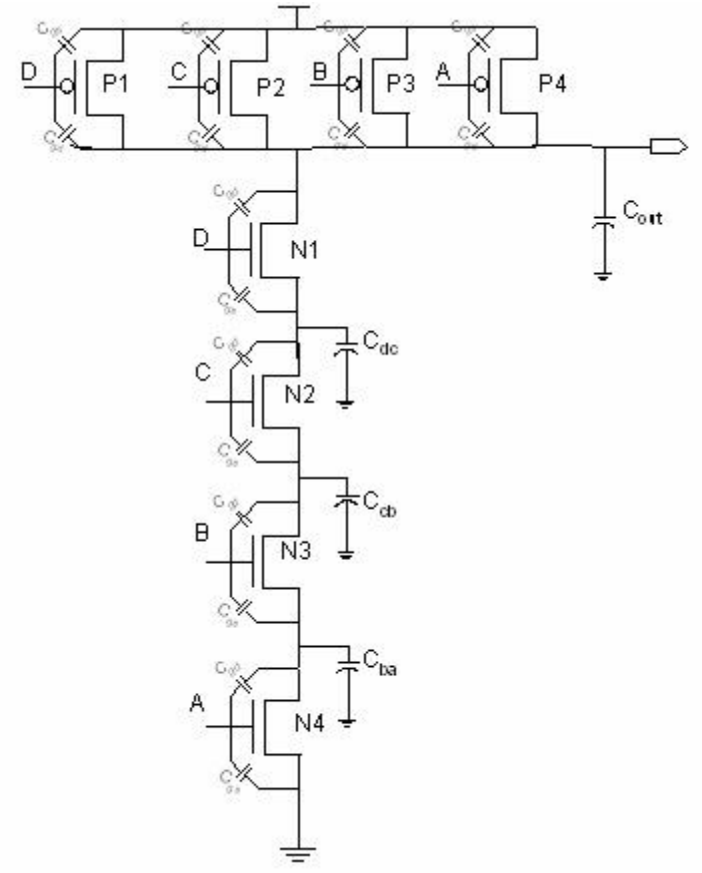

Figure 4.1 Four input NAND gate showing Parasitic Capacitances

For example the fall delay for a four input NAND gate with input A switching is [12],

$$t_{df} = R_{N4}C_{ba} + (R_{N3} + R_{N4})C_{cb} + (R_{N2} + R_{N3} + R_{N4})C_{dc} + (R_{N1} + R_{N2} + R_{N3} + R_{N4})C_{out} \quad (4.2)$$

Where,

$C_{out} = C_{gd-P1} + C_{gd-N1} + C_L$, $C_{dc} = C_{gs-N1} + C_{gd-N2}$, $C_{cb} = C_{gs-N2} + C_{gd-N3}$, $C_{ba} = C_{gs-N3} + C_{gd-N4}$.

C_{gd-P} is the gate to drain capacitance of PMOS, C_{gd-N} is the gate to drain capacitance of NMOS, C_{gs-P} is the gate to source capacitance of PMOS, C_{gs-N} is the gate to source capacitance of NMOS and C_L is the load capacitance i.e. input or gate capacitance of next stage. The value of $C_{db\text{-}N}$ and $C_{db\text{-}P}$ is assumed zero as a result of the SOS process.

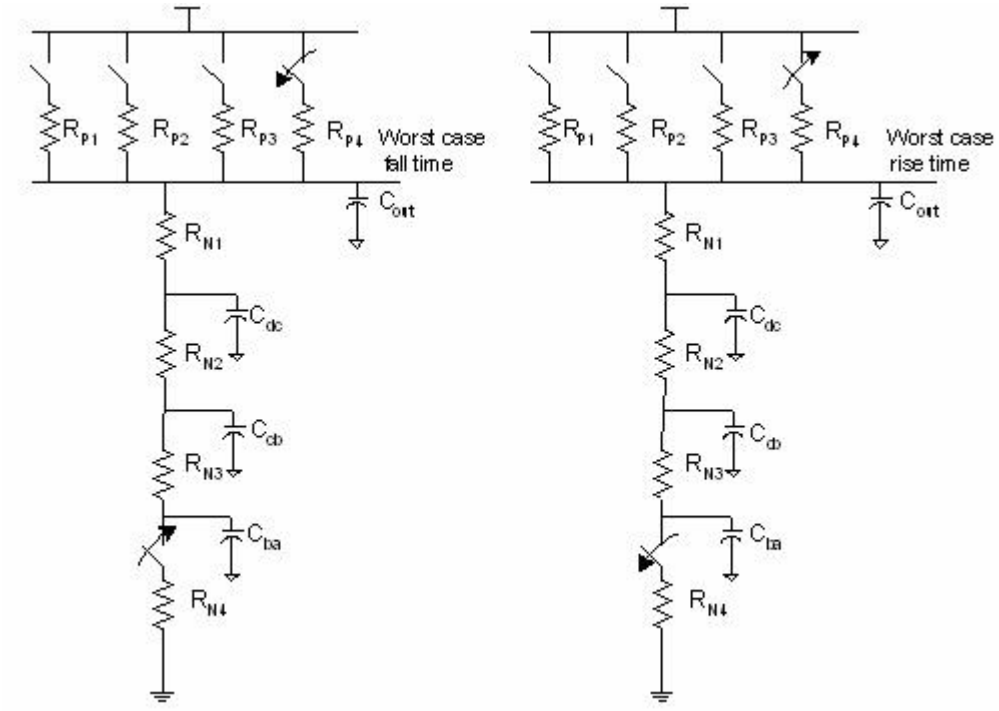

Figure 4.2 Four Input NAND gate equivalent RC Circuit

The value of rise resistance is made equal to fall resistance in the case of 4-Input NAND gate by making the width of the NMOS transistor 4 times wider since the resistance is inversely proportional to the width.

$$R = \rho \frac{L}{W} \Omega \tag{4.3}$$

Where L is the length, W is the width of the transistor and ρ is the equivalent sheet resistance.

Since the NMOS transistors are equal size $C_{dc} = C_{cb} = C_{ba} = C_n$ and $R_{N1} = R_{N2} = R_{N3} = R_{N4} = R_N$ and $R_{P1} = R_{P2} = R_{P3} = R_{P4} = R_P$. The value for C_n can be calculated as

$$C_n = C_{gs} + C_{gd} = \frac{1}{2} C_{ox}(L_n \times W_n) + \frac{1}{2} C_{ox}(L_n \times W_n) \tag{4.4}$$

$$C_n = C_{ox}(L_n \times W_n) \tag{4.5}$$

Since the transistors are scaled according to the number of inputs the capacitance increases with the number of inputs and is given by

$$C_n = f(C_{ox}(L_n' \times W_n')) = f \times C_n' \tag{4.6}$$

Where f is number of fan-in, L_n' is the length and W_n' is the width of an inverter.

Similarly the value of C_p can be calculated as

$$C_p = C_{ox}(L_p \times W_p) = C_p' \tag{4.7}$$

The value of R based on 50%-50% delay is estimated by calculating the change in delay due to change in load and can be calculated as

$$R_N = \frac{\Delta \text{Delay}_{\text{falling}}}{\Delta \text{Capacitance}} \tag{4.8}$$

Similarly,

$$R_P = \frac{\Delta \text{Delay}_{\text{rising}}}{\Delta \text{Capacitance}} \tag{4.9}$$

The values of R_N and R_P are calculated from simulation values where $R_N = \frac{R'_N}{f}$ and $R_P = R'_P$. Where, f is number of fan-in, R'_N and R'_P is inverter pull down and pull-up resistance respectively.

Replacing the Elmore delay equation with these parameters we get a solution with known parameters.

$$t_{df} = R_N C_n + 2R_N C_n + 3R_N C_n + 4R_N C_{out} \tag{4.10}$$

$$t_{df} = 6R_N C_n + 4R_N C_{out} \tag{4.11}$$

Generalizing the above equation for f number of inputs, the NAND gate will have f PMOS in parallel for pull up circuit and f NMOS in series for the pull down circuit. The width of each NMOS is scaled by a factor of f to make the pull up and pull down resistance equal. The worst case delay occurs when the NMOS near the rail switches.

$$t_{df} = f(R_N C_{out}) + (f-1)R_N C_n + (f-2)R_N C_n + \ldots + R_N C_n \tag{4.12}$$

$$= f(R_N C_{out}) + \frac{f(f-1)}{2} R_N C_n \tag{4.13}$$

$$= fR_N \left[C_{out} + \frac{(f-1)}{2} C_n \right] \tag{4.14}$$

Replacing $C_{out} = \frac{C_n}{2} + \frac{C_p}{2} + C_L$, $C_n = \mathrm{f} C'_n$, $R_N = \frac{R'_N}{f}$ and $C_p = C'_p$ we get

$$= f \frac{R'_N}{f} \left[\left[\frac{fC'_n}{2} + \frac{C'_p}{2} + C_L \right] + \frac{(f-1)}{2} fC'_n \right] \tag{4.15}$$

$$= R'_N \left[\frac{C'_p}{2} + \frac{f}{2} C'_n + \frac{f^2}{2} C'_n - \frac{f}{2} C'_n \right] + R'_N C_L \tag{4.16}$$

$$t_{df} = R'_N \left[\frac{C'_p}{2} + \frac{f^2}{2} C'_n \right] + R'_N C_L \tag{4.17}$$

Where, C_L is the load capacitor, R'_N is the fall resistance, C'_p is the PMOS gate capacitance and C'_n is the NMOS gate capacitance of the inverter in the library.

The first term in equation 4.17 is the intrinsic delay and the second term is the transition delay. From equation 4.17 for a given gate with f inputs the intrinsic delay remains constant, but the transition delay varies linearly with the load capacitance, similarly for a fixed load the transition delay remains constant. The intrinsic delay fits a second order polynomial equation with increasing number of inputs.

Similarly the Rise intrinsic delay for a four input NAND gate is as follows:

$$t_{dr} = \sum (R_{pullup-path} \times C_{pullup}) \tag{4.18}$$

$$t_{dr} = R_{P4} C_{ba} + R_{P4} C_{cb} + R_{P4} C_{dc} + R_{P4} C_{out} \tag{4.19}$$

Since $R_{P1} = R_{P2} = R_{P3} = R_{P4} = R_P$ and $C_{dc} = C_{cb} = C_{ba} = C_n$ we can rewrite equation as

$$t_{dr} = 3R_P C_n + R_P C_{out} \tag{4.20}$$

Generalizing the above equation for f number of inputs the worst case rise delay for a 4 input NAND gate occures when the transistor near the rail switches and is given by

$$t_{dr} = (f-1)R_P C_n + R_P C_{out} \tag{4.21}$$

$$t_{dr} = R_P\left[(f-1)C_n + C_{out}\right] \tag{4.22}$$

Replacing $C_{out} = \frac{C_n}{2} + \frac{C_p}{2} + C_L$, $C_n = \mathrm{f}C'_n$, $R_N = \frac{R'_N}{f}$ and $C_p = C'_p$ we get

$$t_{dr} = R'_P\left[(f-1)fC'_n + \left[\frac{fC'_n}{2} + \frac{C'_p}{2} + C_L\right]\right] \tag{4.23}$$

$$= R'_P\left[f^2 C'_n - fC'_n + \frac{fC'_n}{2} + \frac{C'_p}{2}\right] + R'_P C_L \tag{4.24}$$

$$= R'_P\left[f^2 C'_n - \frac{fC'_n}{2} + \frac{C'_p}{2}\right] + R'_P C_L \tag{4.25}$$

$$t_{dr} = R'_P\left[\frac{C'_n}{2}\left(2f^2 - f\right) + \frac{C'_p}{2}\right] + R'_P C_L \tag{4.26}$$

Where,

C_L is the load capacitor, R'_N is the fall resistance, C'_p is the PMOS gate capacitance and C'_n is the NMOS gate capacitance of the inverter in the library.

The first term in equation 4.26 is the intrinsic delay and the second term is the transition delay. Similar to fall delay, the rise delay from equation 4.26 shows that, for a given gate with f inputs; 1) the intrinsic delay remains constant, 2) the transition delay varies with the load capacitance linearly, 3) similarly for a fixed load the transition delay remains

constant, and 4) intrinsic delay fits a second order polynomial equation with an increasing number of inputs f.

The NOR gate design is similar to NAND gate. The PMOS transistors are in series and the NMOS transistors are in parallel. The NAND gate is preferred to the NOR gate due to its reduced leakage and higher speed.

From figure 4.3 it is clear that when the NMOS transistors leak more than PMOS transistors at 200C, the NOR gate has multiple paths for the leakage current to flow causing an increase in the over all leakage current and as well as power dissipation whereas NAND type structures have only one path for the leakage current to flow causing a reduction in power consumption, hence most of the designs in the cell library are NAND based.

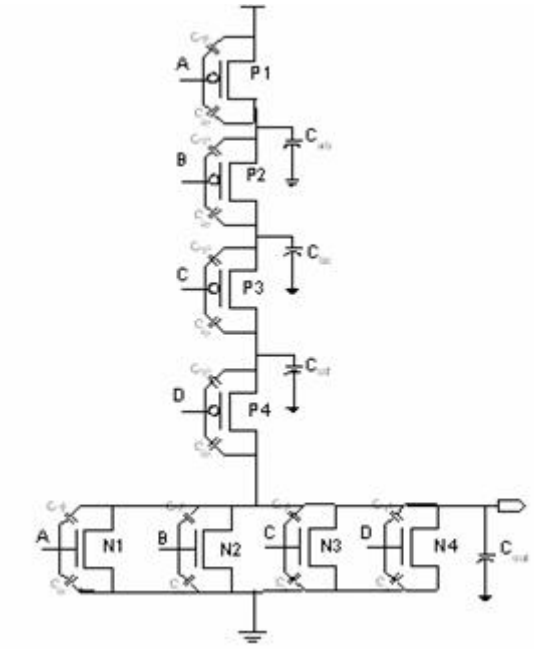

Figure 4.3 Four input NOR gate showing Parasitic Capacitances

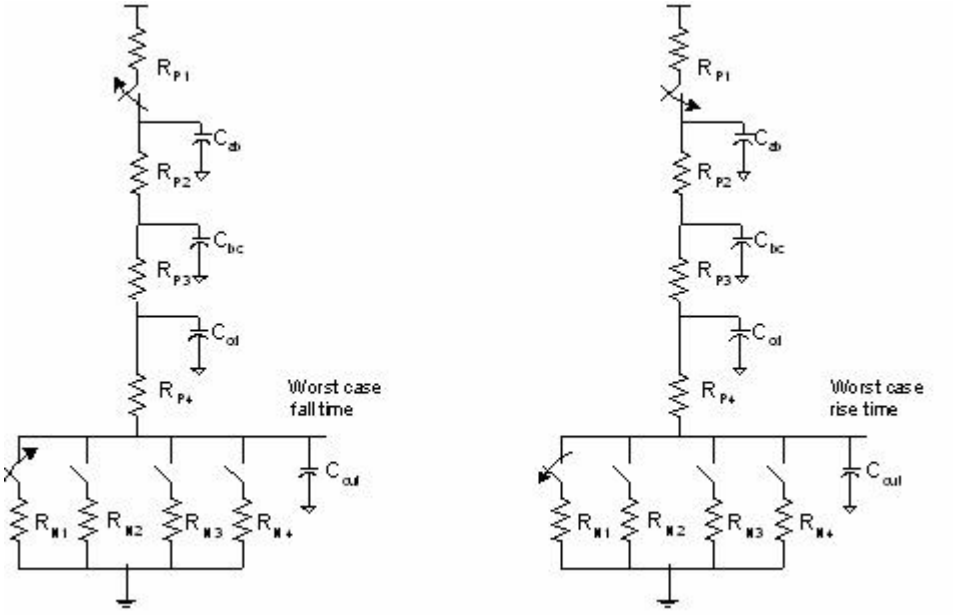

Figure 4.4 Four Input NOR gate equivalent RC Circuit

Similar to NAND gate the RC equivalent of a NOR gate is as shown in Figure 4.4. The worst case fall time delay occurs when the rail transistor switches as shown in Figure 4.4. It can be calculated from the following equation [12].

$$t_{dr} = \sum (R_{pulldown-path} \times C_{pulldown}) \tag{4.27}$$

$$t_{dr} = R_{N1}C_{ab} + R_{N1}C_{bc} + R_{N1}C_{cd} + R_{N1}C_{out} \tag{4.28}$$

Since $R_{P1} = R_{P2} = R_{P3} = R_{P4} = R_P$ and $C_{dc} = C_{cb} = C_{ba} = C_p$ we can rewrite equation as

$$t_{dr} = 3R_N C_p + R_n C_{out} \tag{4.29}$$

Generalizing the above equation for f inputs and replacing $C_{out} = \frac{C_n}{2} + \frac{C_p}{2} + C_L$,

$C_p = \mathrm{f}C'_p$, $R_N = R'_N$ and $C_n = C'_n$ we get

$$t_{df} = R'_N \left[\frac{C'_p}{2}\left(2f^2 - f\right) + \frac{C'_n}{2} \right] + R'_N C_L \tag{4.30}$$

Where,

C_L is the load capacitor, R'_N is the fall resistance, C'_p is the PMOS gate capacitance and C'_n is the NMOS gate capacitance of the inverter in the library.

Similar to a NAND gate, the fall delay equation of a NOR gate is composed of two terms, the first term in equation 4.30 is the intrinsic delay and the second term is the transition delay. From equation 4.30 for a given gate with f inputs; 1) the intrinsic delay remains constant, 2) the transition delay varies linearly with the load capacitance, 3) similarly for a fixed load the transition delay remains constant, and 4) intrinsic delay fits a second order polynomial equation with increasing inputs f.

Similarly the worst case rise delay can be calculated when the transistor near the rail turns from OFF to ON.

$$t_{dr} = \sum (R_{pullup-path} \times C_{pullup}) \tag{4.31}$$

$$t_{dr} = R_{P1}C_{ab} + (R_{P1} + R_{P2})C_{bc} + (R_{P1} + R_{P2} + R_{P3})C_{cd} + (R_{P1} + R_{P2} + R_{P3} + R_{P4})C_{out}$$

Since the PMOS transistors are equal size $C_{cd} = C_{bc} = C_{ab} = C_p$ and $R_{P1} = R_{P2} = R_{P3} = R_{P4} = R_P$ we can rewrite the equation as

$$t_{dr} = 6R_P C_p + 4R_P C_{out} \tag{4.32}$$

Generalizing the above equation for f inputs and replacing $C_{out} = \frac{C_n}{2} + \frac{C_p}{2} + C_L$,

$C_p = \mathrm{f} C'_p$, $R_P = \frac{R'_N}{f}$ and $C_n = C'_n$ we get

$$t_{dr} = R'_P \left[\frac{C'_n}{2} + \frac{f^2}{2} C'_p \right] + R'_P C_L \tag{4.33}$$

Where,

C_L is the load capacitor, R'_N is the fall resistance, C'_p is the PMOS gate capacitance and C'_n is the NMOS gate capacitance of the inverter in the library.

Similar to the fall delay, the rise delay equation is composed of two terms, the first term in equation 4.33 is the intrinsic delay and the second term is the transition delay. From equation 4.33 for a given gate with f inputs; 1) the intrinsic delay remains constant, 2) the transition delay varies with the load capacitance linearly, 3) similarly for a fixed load the transition delay remains constant, and 4) intrinsic delay fits a second order polynomial equation with increasing number of inputs f.

The above equations gives us an approximation of the total delay for different fan-in's for both NAND and NOR type structures. Any complex logic structure can be broken down into any combination of these two gates from which the intrinsic and transition delay can be calculated. For cascade structures the intrinsic delay of the first stage is added to the intrinsic delay of second stage and the transition delay of second stage accounted for.

Thus total delay can be calculated by adding the intrinsic delay with transition delay as discussed in previous chapter.

$$D_{total} = D_{Intrinsic} + D_{Transition} \tag{4.34}$$

The delay can be calculated for three different device models namely; Slow Model, Typical Model, and Fast Model. The fast model is the most optimistic delay model while the slow model is the most pessimistic delay model.

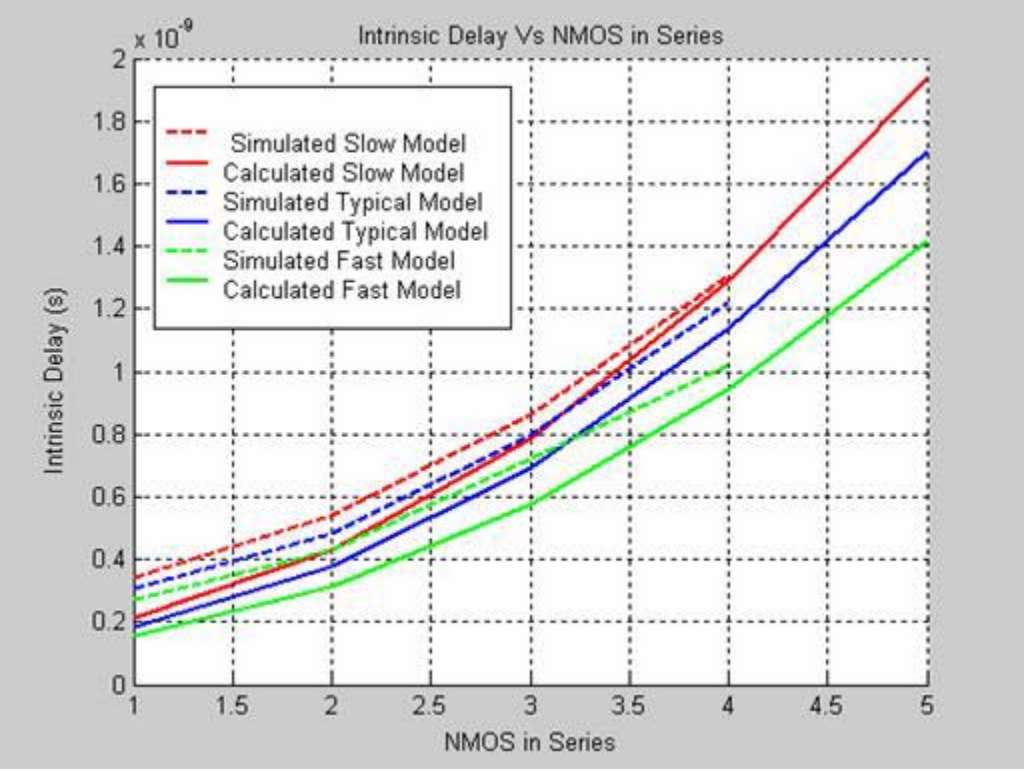

Figure 4.5 Intrinsic Fall Delay Vs NMOS in Series (195^{o}C)

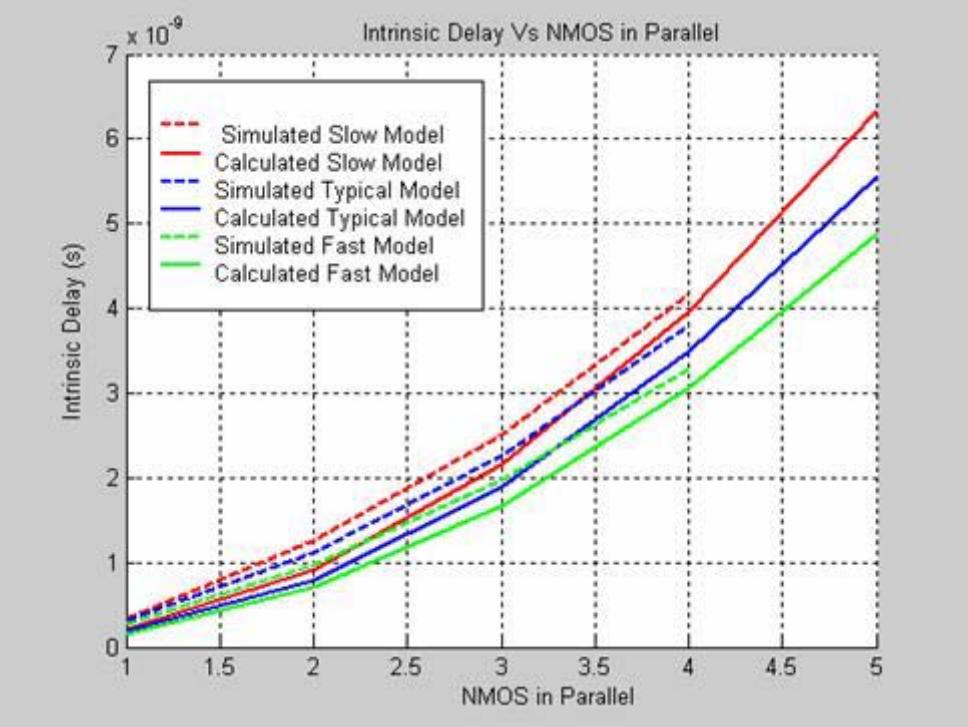

Figure 4.6 Intrinsic Fall Delay Vs NMOS in Parallel (195^{o}C)

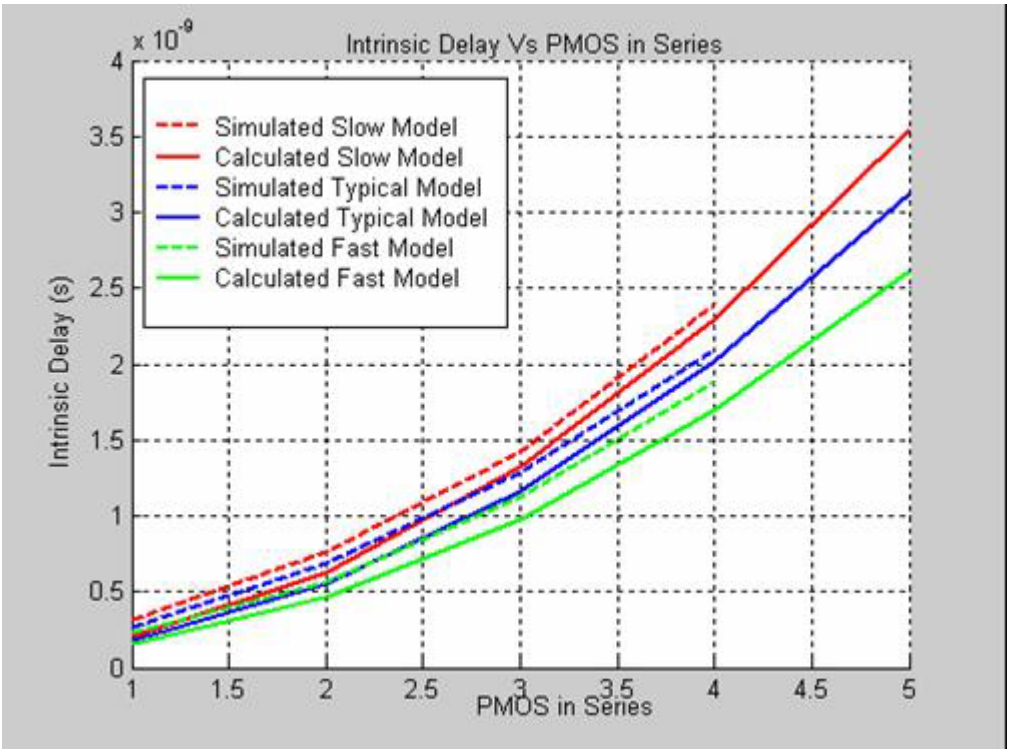

Figure 4.7 Intrinsic Rise Delay Vs PMOS in Series (195^{o}C)

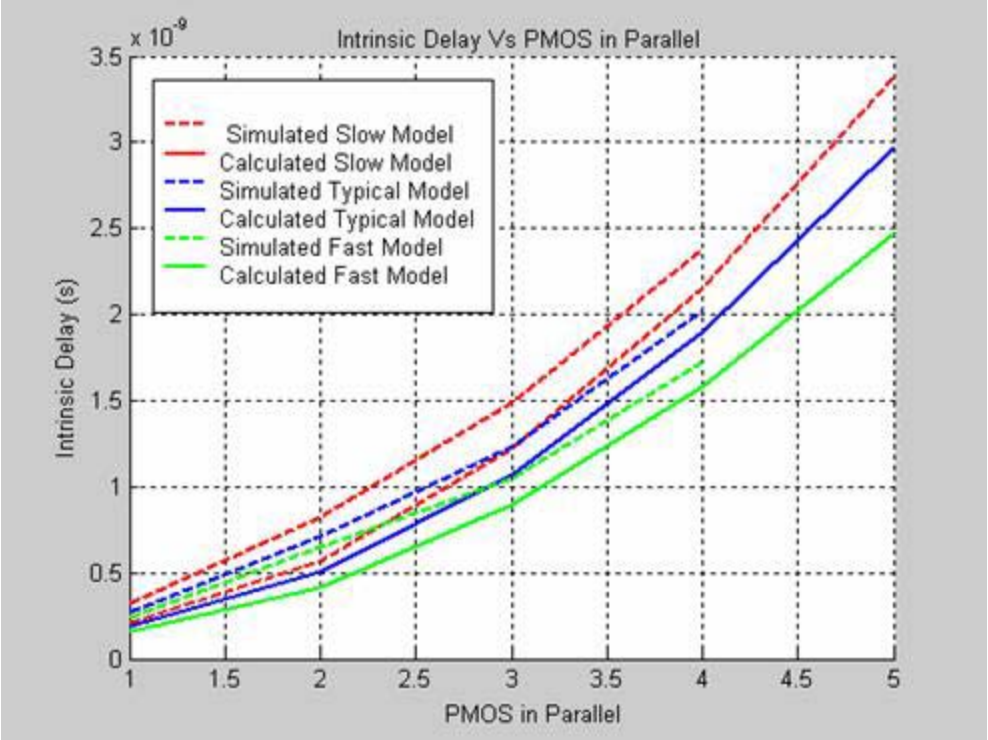

Figure 4.8 Intrinsic Rise Delay Vs PMOS in Parallel (195^{o}C)

Figure 4.5-4.8 shows the variation of intrinsic delay with the architecture of the transistors. The load connected to the gate does not affect the intrinsic delay. It is clear that the theoretical model fits the simulated model with reasonable accuracy for all the three delay models for both configurations of NMOS and PMOS.

Unlike the intrinsic delay the transition delay is a function of output loading, it varies linearly with the load and does not depend on the number of fan-in. This is achieved by increasing the widths of the transistors proportionally to the number of fan-in, such that the pull-up and pull-down resistances are equal.

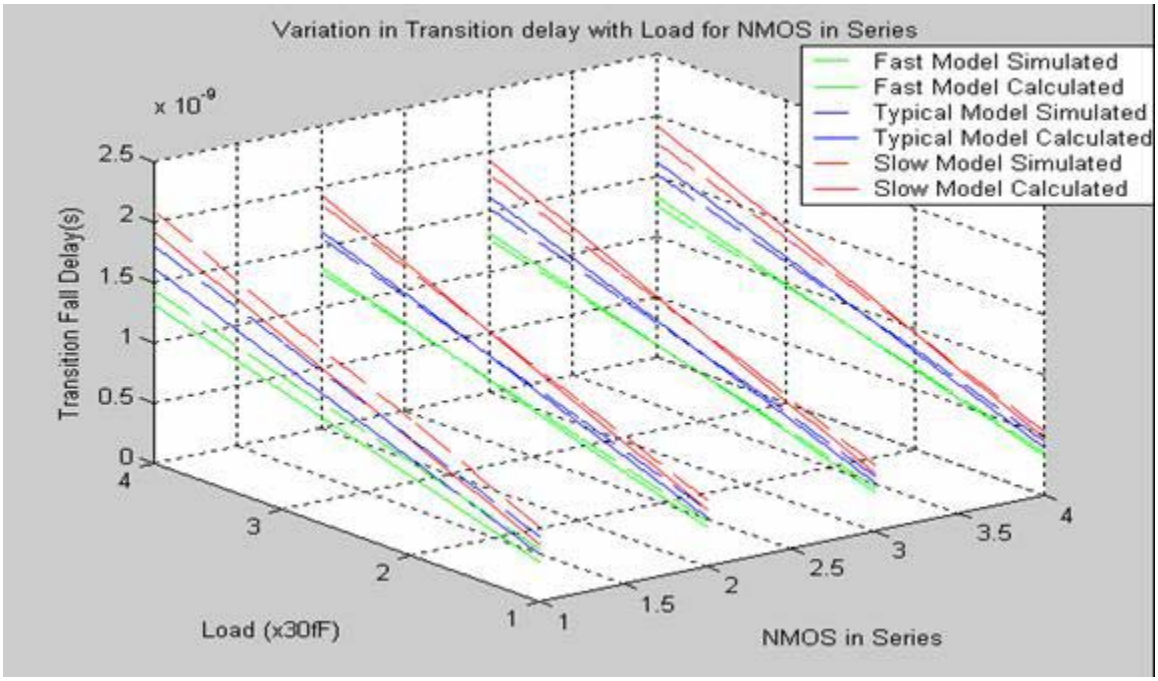

Figure 4.9 Variation in Transition delay with Load for NMOS in Series (195^{o}C)

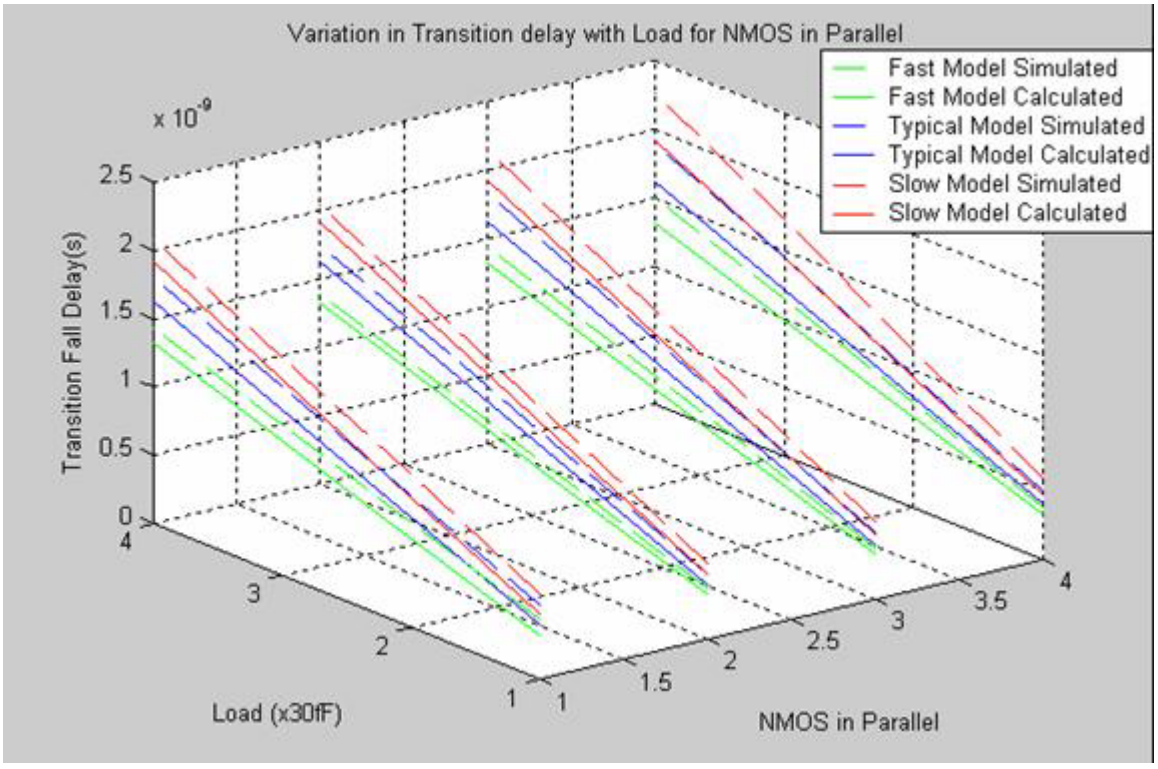

Figure 4.10 Variation in Transition delay with Load for NMOS in Parallel (195^{o}C)

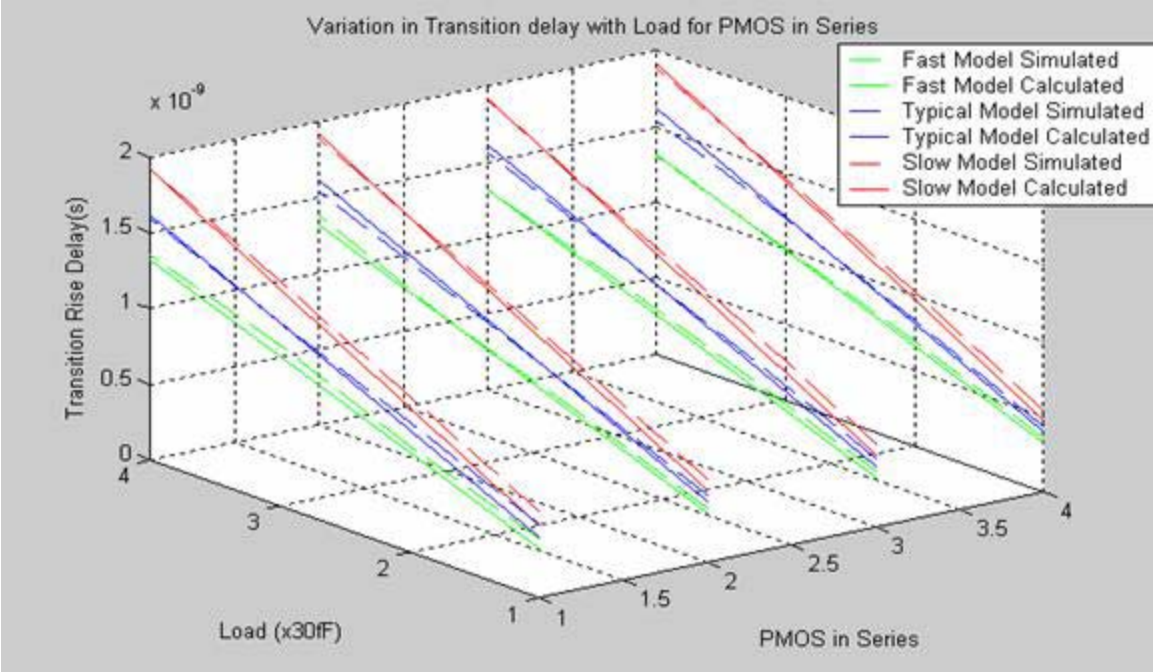

Figure 4.11 Variation in Transition delay with Load for PMOS in Series (195^{o}C)

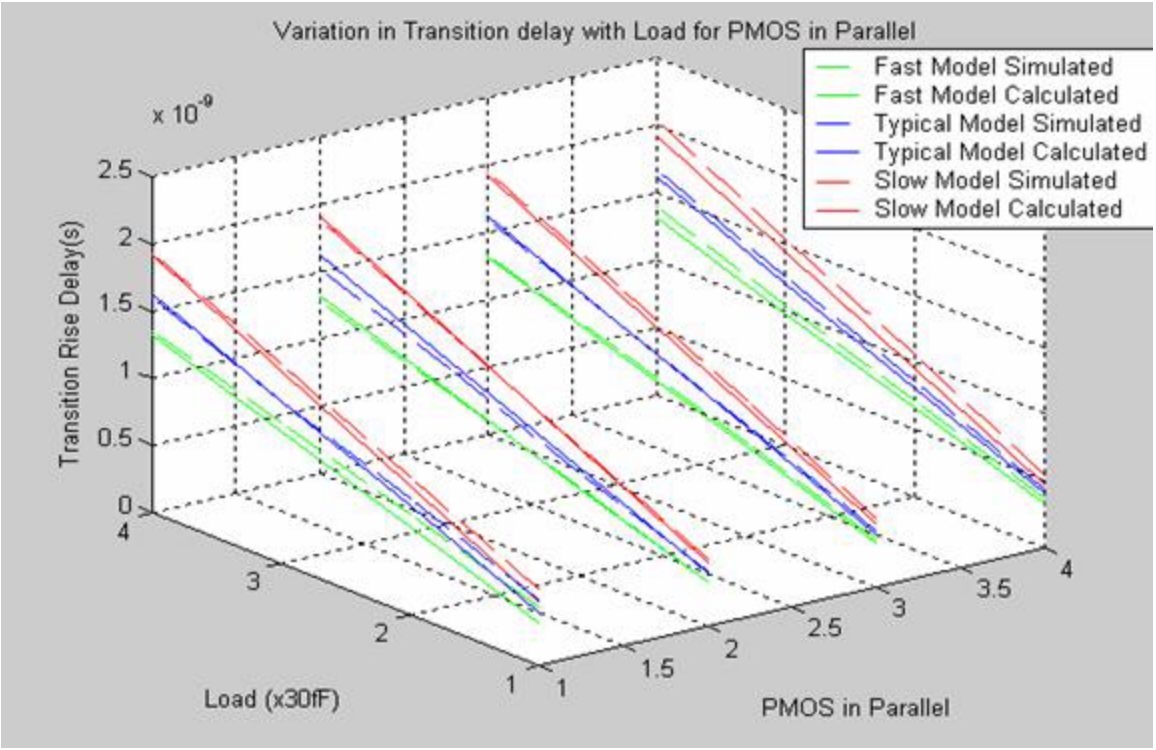

Figure 4.12 Variation in Transition delay with Load for PMOS in Parallel (195^{o}C)

Figure 4.9-4.12 shows the 3-D variation of transition delay with output load for different architectures. The transition delay increases linearly with increase in output load. The transition delay is not affected by the increase in series or parallel combination of both NMOS and PMOS. Similar to intrinsic delay the theoretical model fits the simulated model with reasonable accuracy for all the three delay models for both configurations of NMOS and PMOS. It is also clear that the NOR configurations are slower than the NAND configurations and further strengthens the choice to avoided NOR use in most of the designs. With this theoretical model in hand it is possible for us to predict the delay of a gate with any number of fan-in and with any load with reasonable accuracy. In order to extract the accurate delay parameters a number of test plans were designed. The test plans provide a methodology of extracting both intrinsic and transition delay for combinational logic and setup time and clock to Q delay for sequential logics.

4.3 Hardware Verification of Cell Library.

Hardware verification is necessary for functional verification and delay extraction of the cell library. Each cell has to be functionally verified to make sure the logical behavior of the cell is correct and the delay has to be extracted to verify the delay models developed in the previous section. Hardware Verification involves extraction of DC parameters, functional verification, extraction intrinsic and transition delay parameters for combinatorial logic and extraction of clock to q delay and setup time for sequential logic.

4.3.1 DC and Functional Verification:

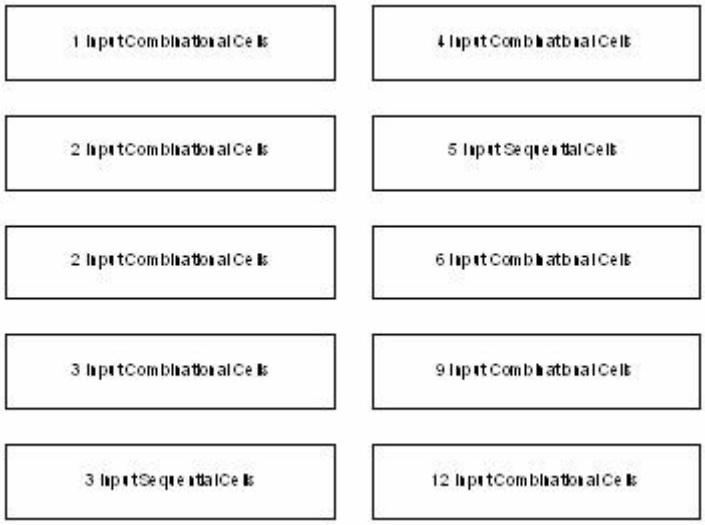

Figure 4.13 Benchmark Circuits 1 for Functional Verification

In order to perform functional verification, the cells are padded out with 2x12 pads for DC testing (Figure 4.13). Benchmark circuits 1 in Figures 4.13 and 4.14 shows how the cells are grouped according to the number of inputs. All the similar inputs are tied together so that by providing a single input it is possible to test all the cells in a particular group. Each test cell group is tied to separate power pad for static and leakage power estimation, while all the grounds are common. DC testing of the entire cell would help us to detect any functional error and also provide us with an accurate static power model for the library.

The DC testing of the library provides us with the VTC curves for the cells from which the noise margins and the β ratios can be verified.

Figure 4.14 Cadence Implementation of Benchmark circuits 1

4.3.2 Delay model verification for Combinatorial Logic

The delay model developed in the previous section fits the simulated data with a reasonable accuracy, however in order to derive an accurate delay model it is necessary to validate the cell library using hardware test plans, but it is not possible to verify the delay parameters of all the cells in the library using hardware methods as it would consume excessive chip area. Hence a test plan to extract intrinsic and transition delay parameters for combinatorial logic is developed with a limited number of cells. The resulting data can later be used to extrapolate to the remaining or balance of cells in the library. Any cell in the library can be realized by reducing to any one of these cells or cascading these cells in the test plan.

NMOS / PMOS	1/4	1/3	1/2	1	2	3	4
4	NOR4 (4 TIED)	NOR4 (3 TIED)	NOR4 (2 TIED)	**NOR4 (1 TIED)**	**AOI42 (1 TIED)**	**AO143 (1 TIED)**	
3		NOR3 (3 TIED)	NOR3 (2 TIED)	**NOR3 (1 TIED)**	**AOI32 (1 TIED)**	**AOI33 (1 TIED)**	
2			NOR2 (2 TIED)	**NOR2 (1 TIED)**	**AOI22 (1 TIED)**	**AOI23 (1 TIED)**	
1				**INV**	**NAND2 (1 TIED)**	**NAND3 (1 TIED)**	**NAND4 (1 TIED)**
1/2					NAND2 (2 TIED)	NAND3 (2 TIED)	NAND4 (2 TIED)
1/3						NAND3 (3 TIED)	NAND4 (3 TIED)
1/4							NAND4 (4 TIED)

Table 4.1 Test Plan

Table 4.1 is the test plan for characterizing the cell library, the cells in bold result in worst case delays as a result of only one input transitioning. These cells are selected for characterization of the worst case delays which are in turn used in simulations. The objective being to validate the model introduced in section 4.2 using the timing results from worst case and extrapolate to all combinatorial cells.

Validating the library insures two things, first it ensures that the silicon is properly bounded by the timing models that have been generated and secondly it enhances the performance and timing accuracy of the cell library [15]. By comparing the measured data to simulated data it is possible to tune the cell library delay model.

The test plan consists of delay chains for timing analysis as shown in figures 4.15 and 4.16, these delay chains can be used to extract both intrinsic and transition delay of the cells. The chains are designed to generate pulses for both high to low and low to high transitions; the number of cells in the delay chain is long enough to ensure accurate timing data from the delay chain. The pulses are produced by an XNOR gate as shown in Figure 4.15 that has both the delayed pulse due to the delay chain and non delayed pulse as inputs.

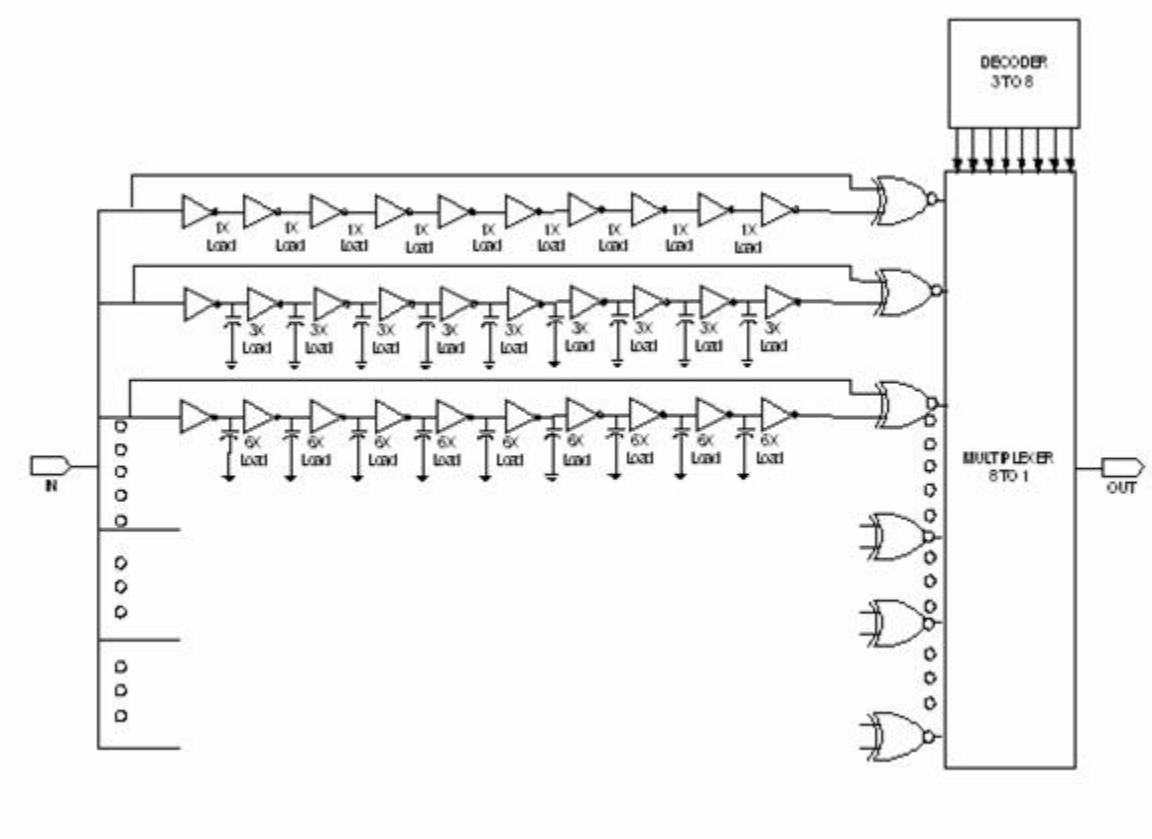

Figure 4.15 Single Input Delay Chain

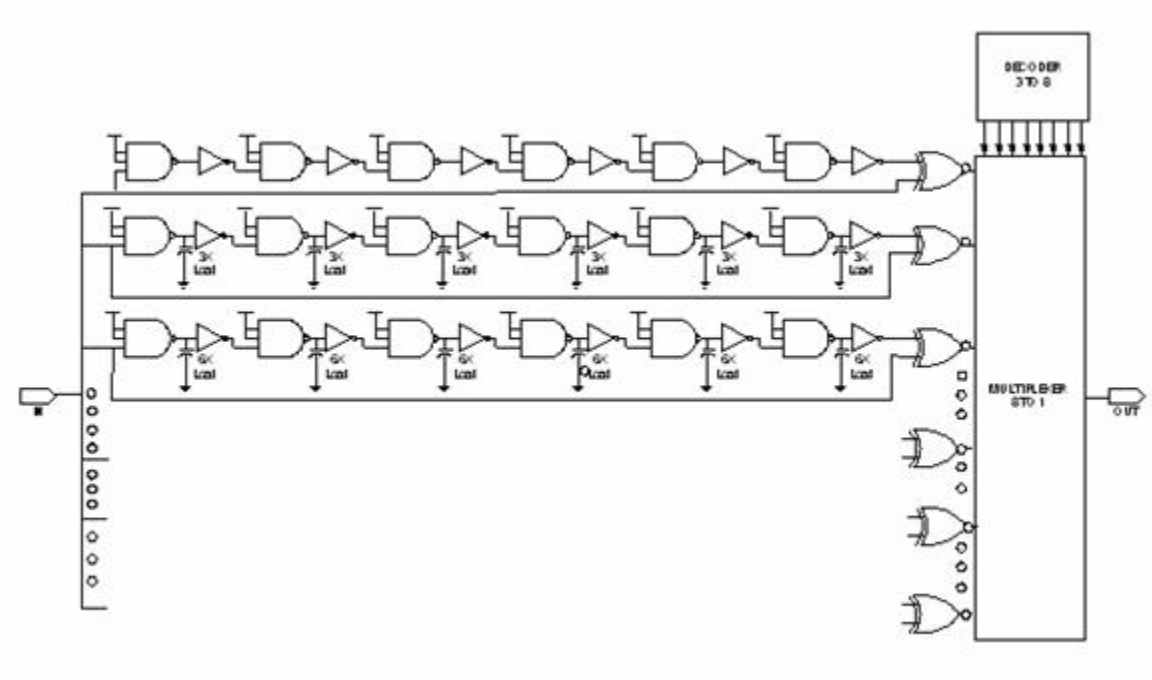

Figure 4.16 Multi Input Delay Chain

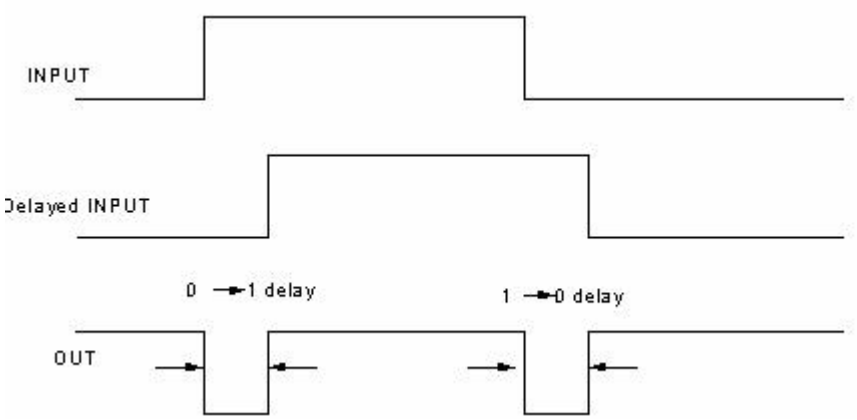

Figure 4.17 Pulse Generator Waveform

Figure 4.18 I/O Buffer Characterization

Typically there are 30 cells in the delay chain, so that the pulses are long enough in duration to be measured, single input cells are simply chained together, whereas multiple-input combinatorial cells have their inputs tied high or low except for one input whose transition results in a change in output as shown in figure 4.17. For example, the two inputs of the three input NAND gate are tied to VDD and the third input is involved in transition thus causing a state change in output. Similarly for a NOR gate all inputs except for one are tied to VSS. Delay chain of each cell has three different loading conditions; cells are loaded with 1X load i.e. loaded with another cell of same type, 3X load and 6X load by placing equivalent capacitors between each pair of cells in the chain. See figures 4.15 and 4.16. These capacitors encompass the probable different output loading of each type of cell on the fast delay extreme and are a compromise in area over broader accuracy. By not addressing heavy loading, i.e. 7X to 16X, only non-critical timing is compromised resulting in little or no impact in performance.

The I/O buffers are characterized by a direct path between two adjacent buffers from input to output as shown in figure 4.18. This path is provided for each test plan to make measurements easier and faster. These paths provide delay data as well as functional verification for the I/O pad cells. Due to three different loading conditions, each chain has different pulse widths corresponding to their loading, which can be written as

$$PulseWidth_{1\times} = Delay_{Intrinsic} + 1 \times Delay_{Transition} \tag{4.35}$$

$$PulseWidth_{3\times} = Delay_{Intrinsic} + 3 \times Delay_{Transition} \tag{4.36}$$

$$PulseWidth_{6\times} = Delay_{Intrinsic} + 6 \times Delay_{Transition} \tag{4.37}$$

Solving these three equations, it is possible to find the intrinsic and transition delay associated with each cell. Ring oscillators are also designed to verify the results from test plans. The non inverting delay chains can be tested for both high to low and low to high transitions which provides unique values for rise delay and fall delay, whereas inverting delay chains i.e. ring oscillators yield identical results for both transitions.

The pulse width is only a function of the pulse generator hence the input, output and control circuit delays are not important, and as such do not alter the pulse width. The cadence implementation of the test plans are shown in Figure 4.19.

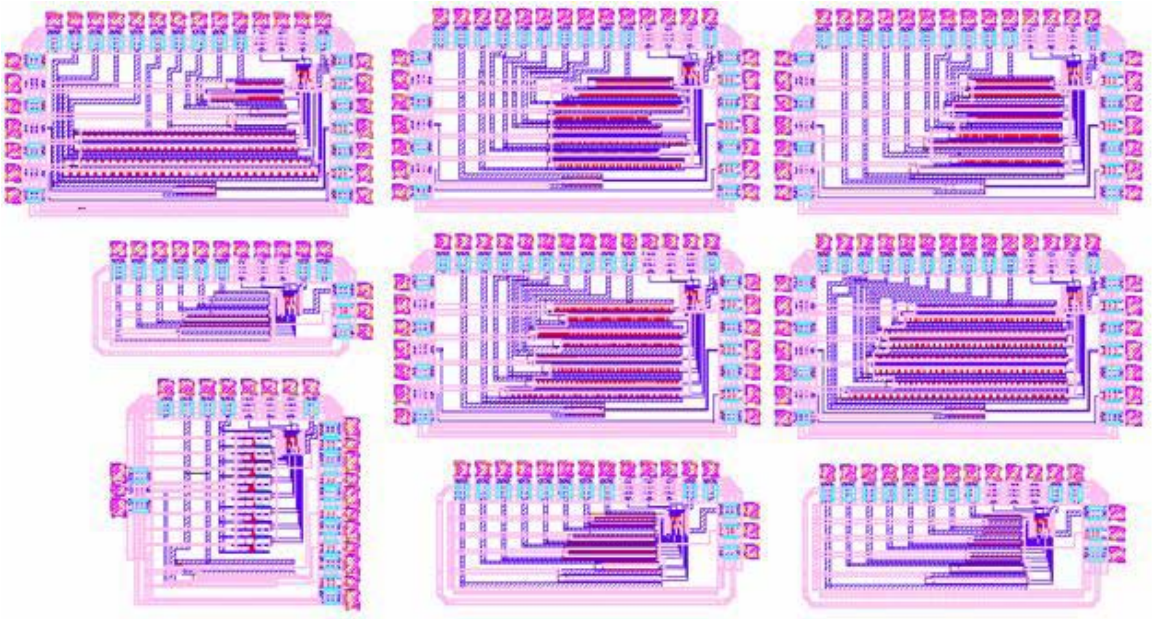

Figure 4.19 Cadence Implementation of Test Plans

4.3.3 Delay model verification for Sequential Logic

Sequential Circuits are characterized for Clock to Q delay and Setup time. Since Master-Slave architecture is used for flip-flops in this library, the hold times tend to be shorter or even negative, meaning that the flip-flops or latches can latch data even if the data transition occurs before the clock arrives, hence hold time measurement is not critical in these circuits. The library contains 8 flip flops and all of them have the same architecture, hence characterization of a single flip flop can be generalized to all the sequential logics in the library.

Characterizing sequential circuits is carried out using two benchmark circuits; the first benchmark circuit is as shown in Figures 4.20 and 4.21. The objective here is to estimate the setup time of the flip-flop with a single input, as using two separate inputs for data and clock may lead to synchronization problems.

The equal delay path aligns the clock and data, other paths have inverter chains added between clock and data to delay the clock from data. The first path fails to provide the valid transition at the output because of the alignment of clock and data which causes a setup time violation. Other paths delay the clock from data, thus providing the necessary setup time. By calculating the number of inverters used in the chain that produces the first valid output transition, it is possible to estimate the setup time.

The inverter delay chain consists of odd and even inverters to measure both rising and falling setup times. Each delay chain is connected to two flip-flops avoiding metastability issues. The output from AND gate as shown in figure 4.20 is true only when both the flip flops produce valid transitions. This is to avoid the possibility of a flip flop accidentally enter a valid transition even with setup time violations. The probability of two devices entering metastability is greater than 1 chance in a billion.

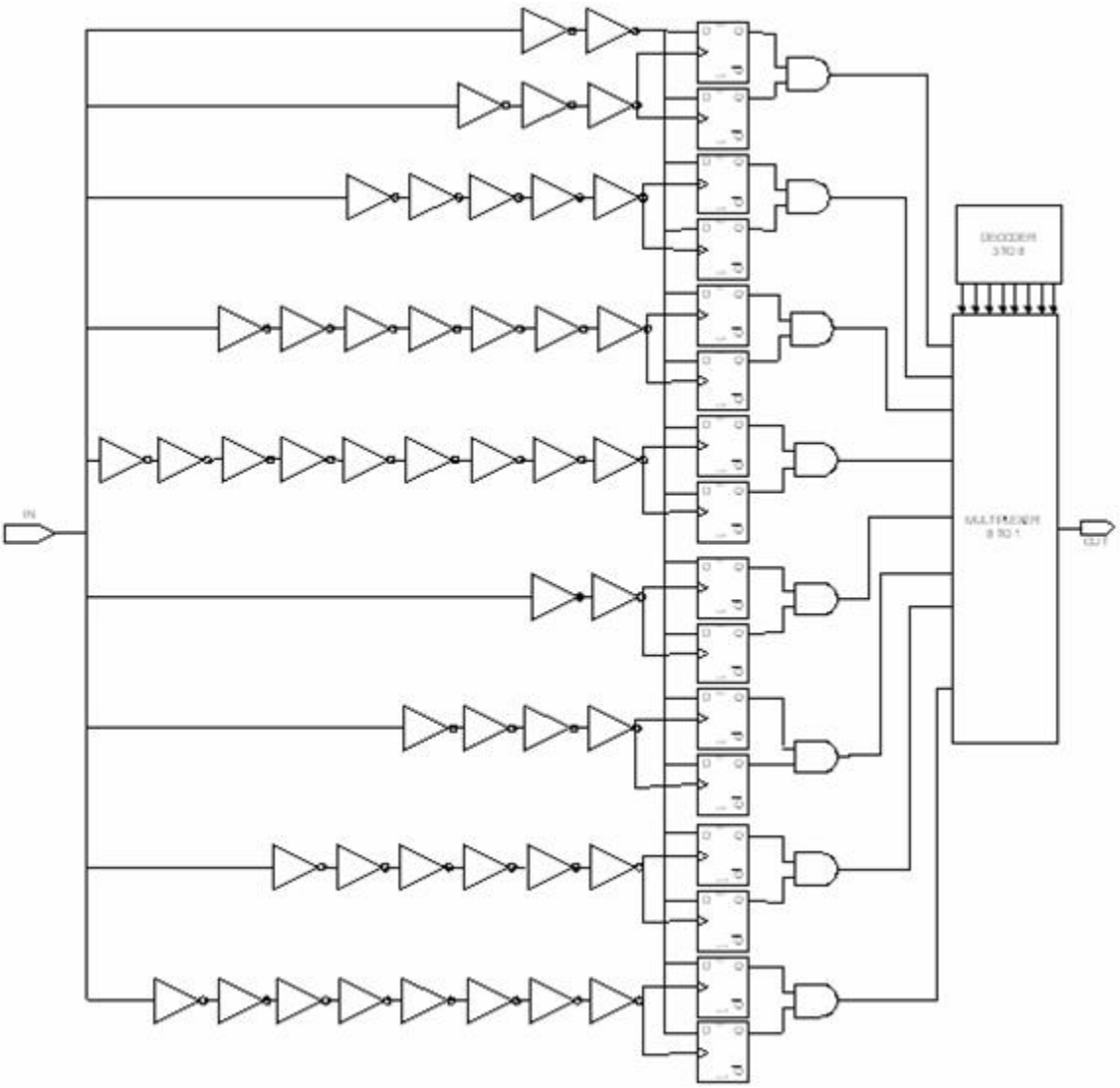

Figure 4.20 Sequential Benchmark Circuit 1

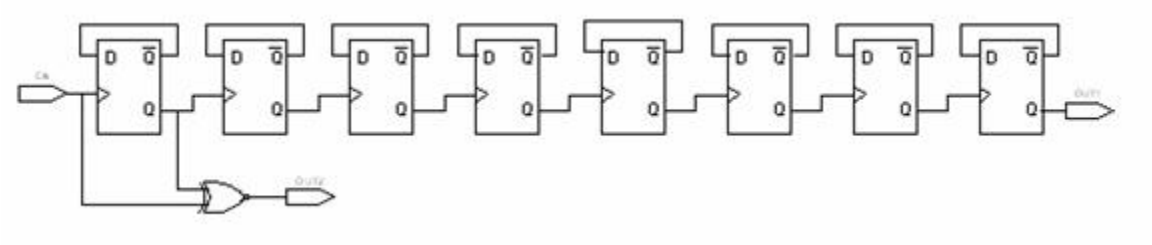

Figure 4.21 Sequential Benchmark Circuit 2

The second benchmark circuit is as shown in Figure 4.21, it is used to find clock to Q delay and setup time. The benchmark circuit 2 uses eight flip-flops in cascade so that the input time period is increased 256 times, which enables the output to be easily measured and each stage practically squares the probability of failure due to metastability causing the circuit to fail once in 2^{256} billion times [22]. The clock frequency is increased until the flip-flops fails to provide the correct output. The time period corresponding to the failing clock frequency is calculated and can be entered into equation 4.39 which describes the timing relation between flip-flop parameters and the signal delay.

$$T_{Period} = T_{Clk-Q} + T_{SU} + T_{XNOR} \tag{4.38}$$

The XNOR circuit as shown in figure 4.21 is used as a pulse generator whose pulse width corresponds to the clock to Q delay as shown in figure 4.22. Subtracting the value of clock to Q delay from the total period equation (4.39) it is possible to estimate the setup time of the flip flop. Results from benchmark circuit 1 and benchmark circuit 2 can be used to verify simulation results achieving accurate delay and setup times.

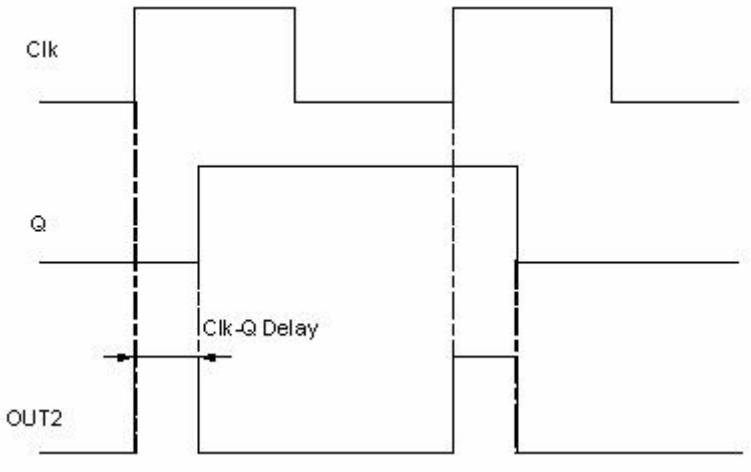

Figure 4.22 Clock to Q Delay Waveform

Chapter 5

Measurements and Results

This section of the thesis summarizes the measurement results of the benchmark circuits. The benchmark circuits were fabricated in Peregrine SOS process and tested on sn 8" Cascade Alessi REL-6100 semi automatic probe station. The measured results were compared with the simulated and calculated data from previous chapter.

5.1 Measured Delay of Combinatorial Logic:

The linear delay model parameters were extracted from the bench mark circuits as described in section 4.3.2. Five dies from 4 wafers were measured for statistical analysis. Each delay chain has 30 gates, chosen to achieve statistical significance. The variation of delay with the 20 dies is shown in figure 5.1 which roughly follows a typical Gaussian curve indicating the tightness of the process time constant.

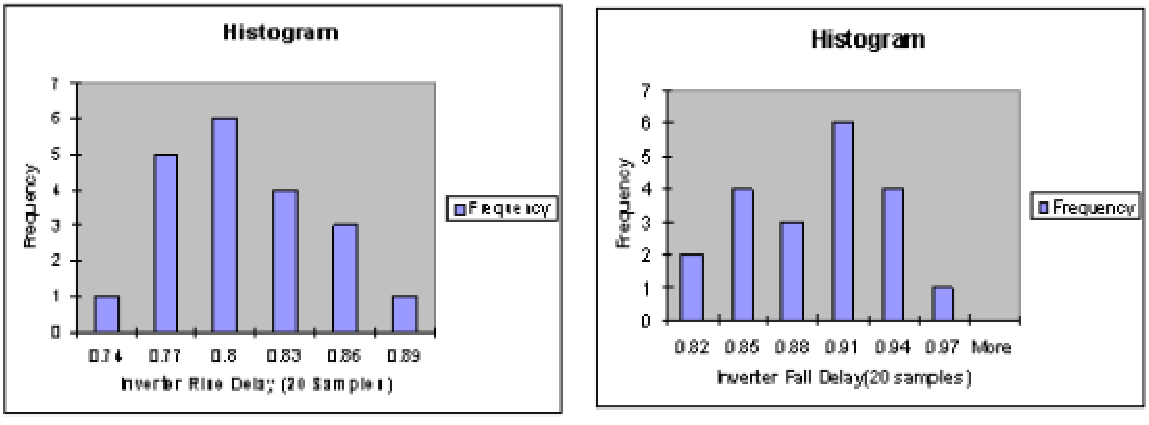

Figure 5.1 Variation of Inverter delay with die samples

The standard deviation of the rise delay was found to be 0.039 and fall delay was found to be 0.046. Table 5.1 shows the rise and fall intrinsic delays of the gates with their standard deviations.

Gate	Rise Intrinsic τ (ns)	Fall Intrinsic τ (ns)
Inverter	0.4478 ± 0.028	0.4476 ± 0.029
2 Input NAND	0.660 ± 0.17	0.5117 ± 0.06
3 Input NAND	0.99480 ± 0.15	0.64 ± 0.26
4 Input NAND	1.51290 ± 0.036	0.8405 ± 0.01
2 Input NOR	0.6554 ± 0.03	1.1445 ± 0.118
3 Input NOR	0.9361 ± 0.05	1.8032 ± 0.114
4 Input NOR	1.4399 ± 0.03	3.118 ± 0.111

Table 5.1 Measured Intrinsic delay

Table 5.2 shows the rise and fall transition delays of the gates with their standard deviations.

Gate	Rise Transition τ (ns)	Fall Transition τ (ns)
Inverter	0.33 ± 0.017	0.35 ± 0.025
2 Input NAND	0.3201 ± 0.12	0.2547 ± 0.024
3 Input NAND	0.3313 ± 0.095	0.2478 ± 0.032
4 Input NAND	0.3619 ± 0.012	0.2505 ± 0.003
2 Input NOR	0.2811 ± 0.01	0.352 ± 0.039
3 Input NOR	0.2799 ± 0.012	0.3546 ± 0.04
4 Input NOR	0.2663 ± 0.01	0.3672 ± 0.0038

Table 5.2 Measured Transition delay

A 5 sigma statistical analysis was performed on the measured data and the variation of the delay time constant was found to be as shown in table 5.3, which means 9999 in 10000 gates will have their delay time constant bound by the minimum and maximum values in table 5.3.

Gate	Rise Transition τ (ns) - Min	Rise Transition τ (ns) - Max	Fall Transition τ (ns)- Min	Fall Transition τ (ns)-Max
INV	0.225	0.475	0.215	0.455
NAND	0.27	0.39	0.126	0.412
NOR	0.23	0.33	0.2	0.5

Table 5.3 5-Sigma Delay Variation.

The measured delay data is plotted against the simulated and calculated delay data to verify the measured results. Figure 5.2-5.5 shows the variation of intrinsic delay with the architecture of the transistorized cells.

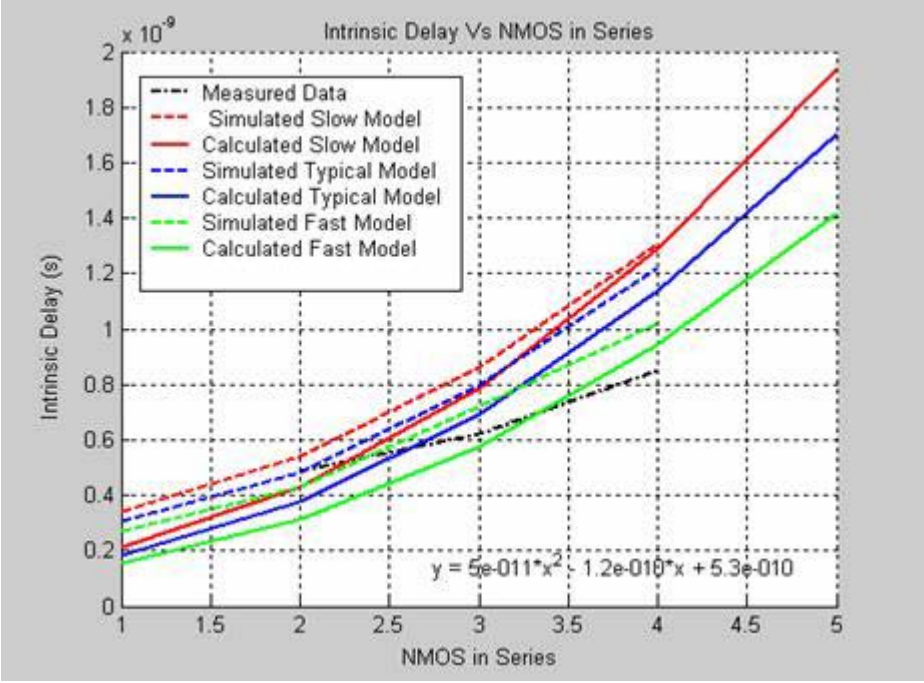

Figure 5.2 Intrinsic Fall Delay Vs NMOS in Series (195^{o}C)

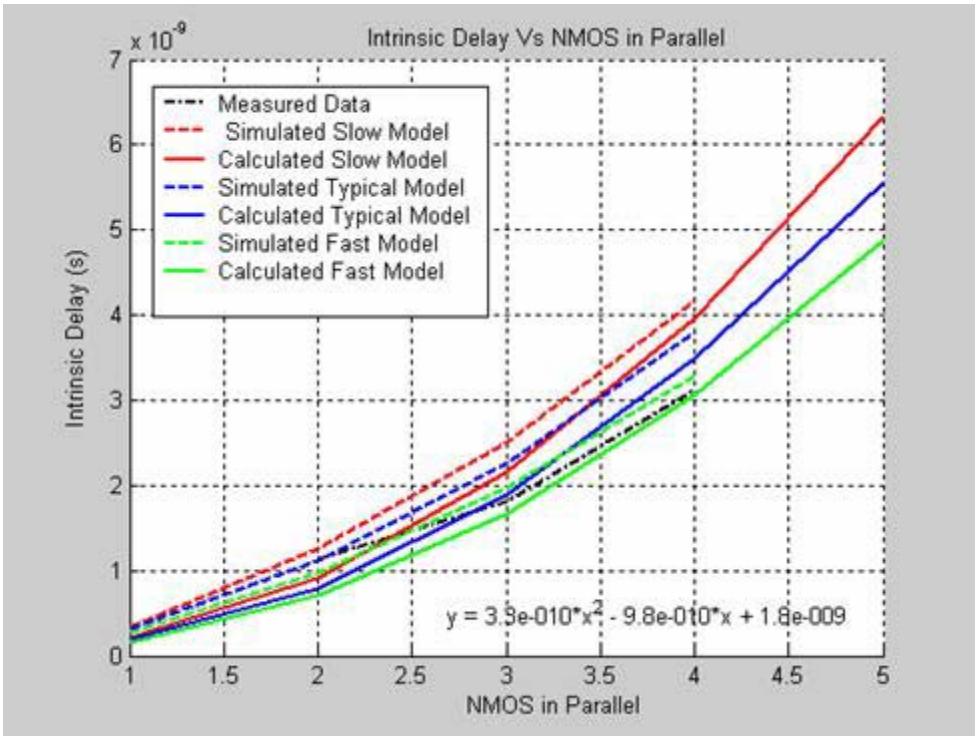

Figure 5.3 Intrinsic Fall Delay Vs NMOS in Parallel (195^{o}C)

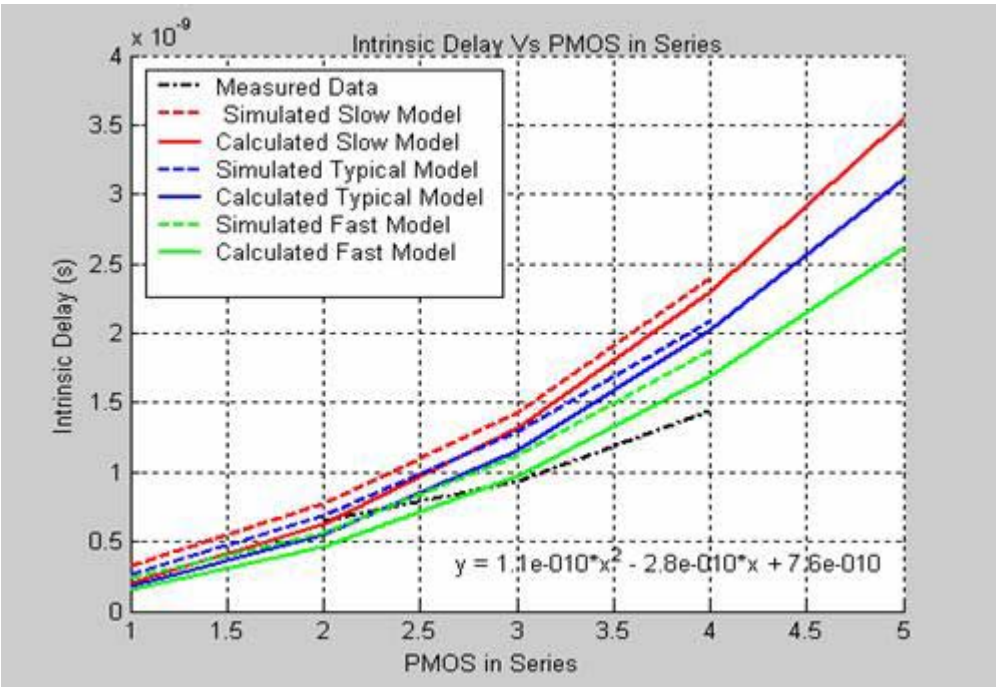

Figure 5.4 Intrinsic Rise Delay Vs PMOS in Series (195^{o}C)

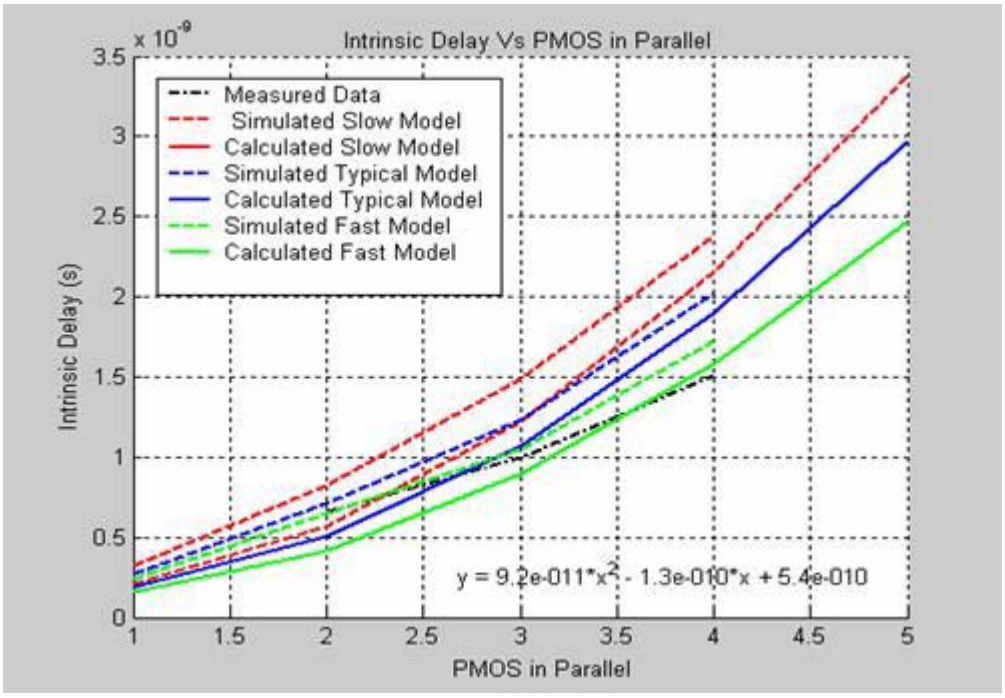

Figure 5.5 Intrinsic Rise Delay Vs PMOS in Parallel (195^{o}C)

The intrinsic delay is not affected by the load connected to the gate. It is clear that the measured data has an excellent fit with the simulated and calculated data for both configurations of NMOS and PMOS.

Similarly the measured transition delays are also plotted against the simulated and calculated delays for different loads. Figure 5.6- 5.9 shows this variation and can be extended to other loads as well.

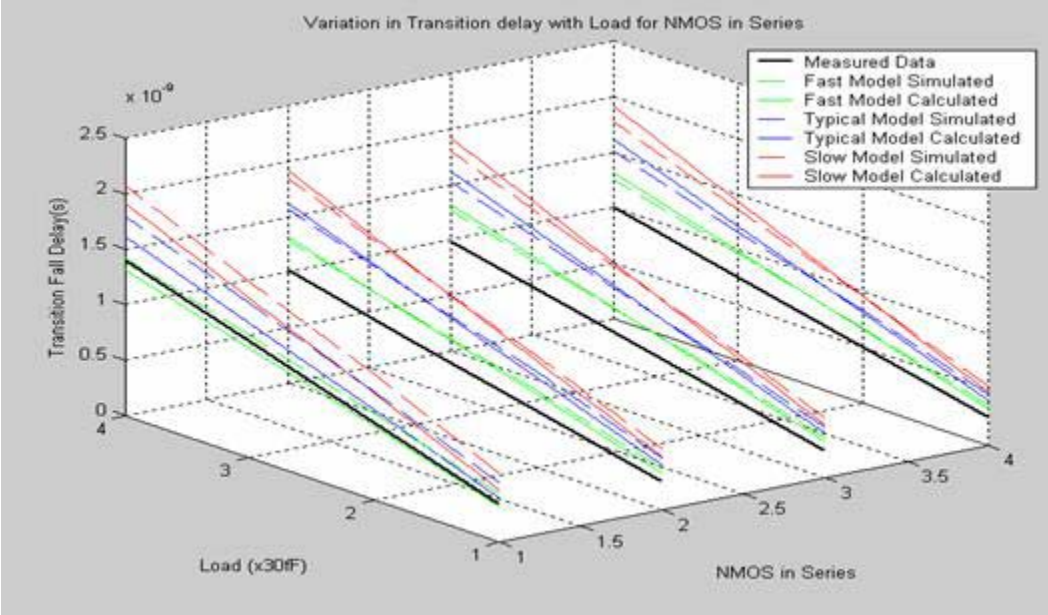

Figure 5.6 Transition Fall Delay Vs NMOS in Series (195^{o}C)

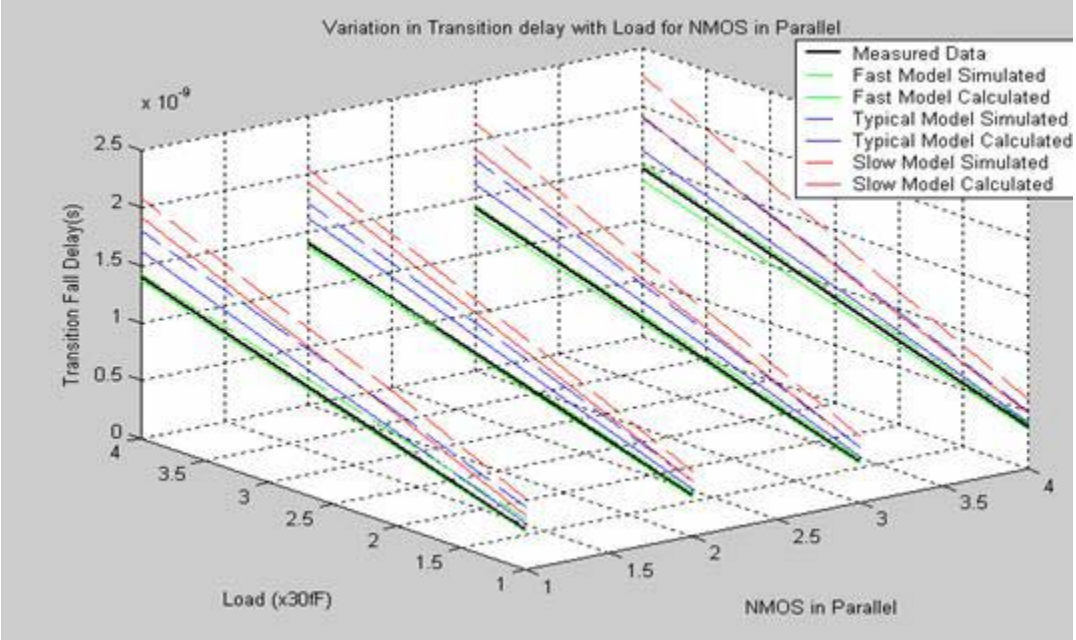

Figure 5.7 Transition Fall Delay Vs NMOS in Parallel (195^{o}C)

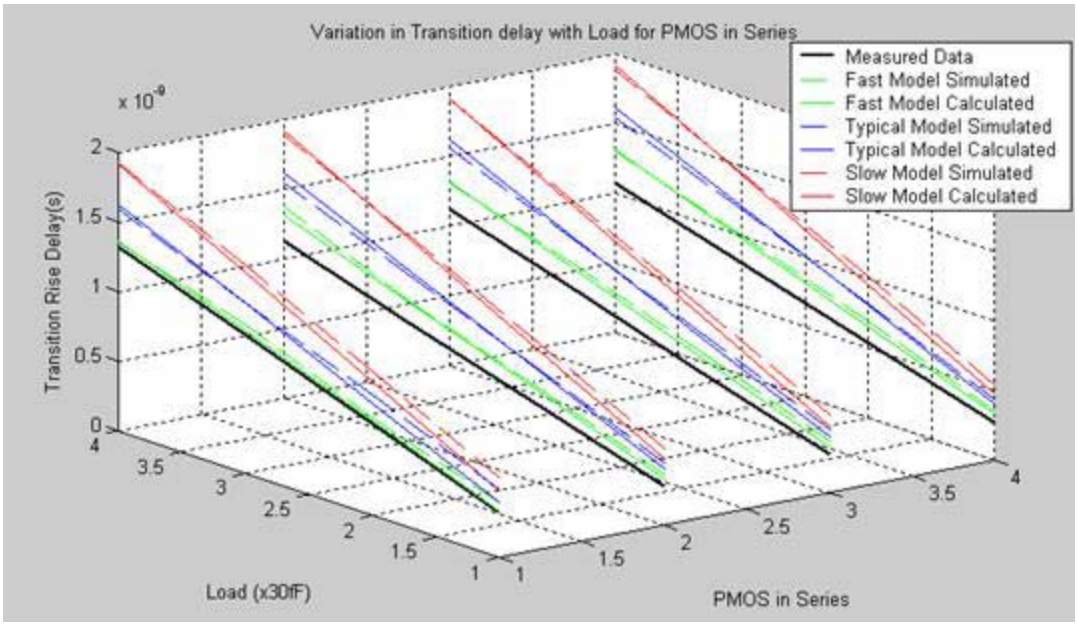

Figure 5.8 Transition Rise Delay Vs PMOS in Series (195^{o}C)

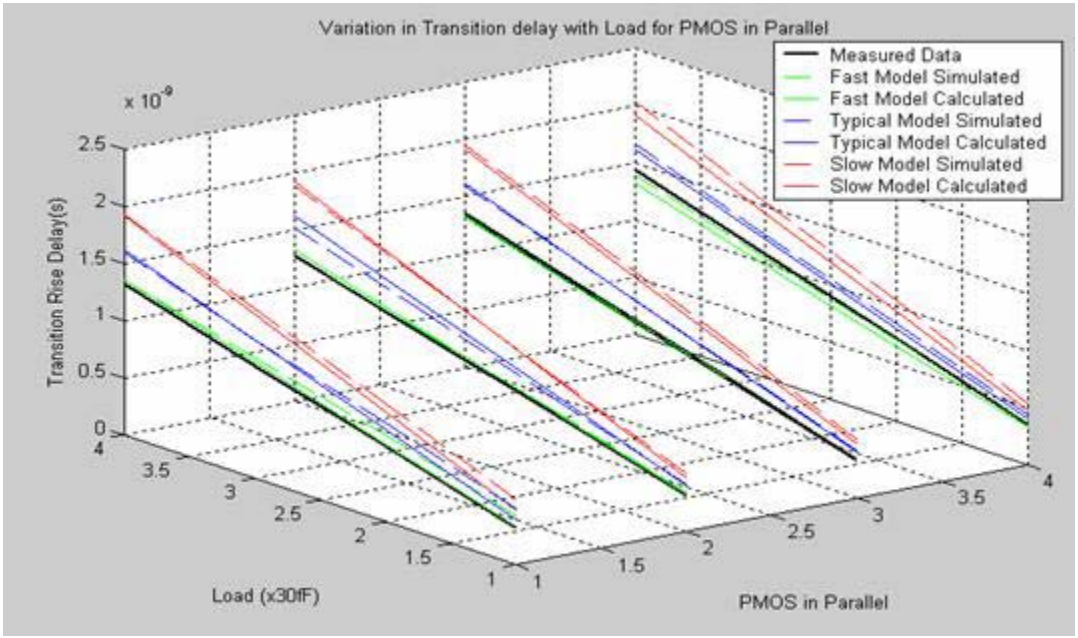

Figure 5.9 Transition Rise Delay Vs PMOS in Parallel (195^{o}C)

The transition delay increases linearly with increase in output load. Similar to intrinsic delay the measured transition data has an excellent fit with the simulated and calculated data with reasonable accuracy for both configurations of NMOS and PMOS.

5.2 Measured Delay of Sequential Logic:

Two different benchmark circuits were tested to characterize the sequential logic as discussed in section 4.3.3. From benchmark circuit1 the setup time of the flip flop is estimated using the measured data from section 5.1 and from benchmark circuit2 the clock to Q delay and setup time is measured.

5.2.1 Benchmark Circuit 1:

From benchmark circuit1 the average minimum number of inverters in chain to produce a valid output at 195^{o}C is 2, by estimating the delay of 2 inverters from the measured data in previous section, the setup time is found to be 2.56 ns**.** The setup time measured from benchmark circuit 1 has a granularity of one inverter with a 5 sigma rise plus fall time variation of less than 0.66 ns.

5.2.2 Benchmark Circuit 2:

Benchmark circuit 2 provides a way of measuring both the setup time and clock to Q delay accurately. The average Clock to Q delay is measured to be 11.9 ns. The maximum frequency at which the clock divider can produce a valid output at 195^{o}C is 75MHz, hence the time period (T_{period}) is 13.3 ns.

$$T_{SU} = T_{Period} + T_{Clk-Q} - T_{XNOR} \tag{5.1}$$

The intrinsic delay of XNOR is calculated from the inverter and NAND data and is found to be 0.96 ns. Substituting the values in equation 5.1 we get the setup time to be 2.37 ns. A T test [23] is performed to achieve 3 ‘9 estimate of the measured data.

$$t = \frac{\overline{x} - 3}{\frac{s}{\sqrt{n}}} \tag{5.2}$$

Where,

$$\overline{x} = 2.37, n = 4, s^2 = \frac{\sum x_i^2 - (\sum x_i)^2 / 4}{4-1} = 0.0097$$

Substituting the above values in equation 5.2 we get t = -12.989. Since t = -12.989 is not in the rejection region (t <-12.924), Setup time < 3ns is not rejected at a level .0005. Hence 999 of 1000 flip flops have their setup in the range 1.74ns < T_{SU} < 3ns. Similarly 999 of 1000 flip flops have their clock to q delay in the range 8.8ns < T_{Delay} < 15ns.

5.3 Power Consumption:

As discussed in section 3.4, the most dominant term is the switching power, this results from charging/discharging of the load capacitance. Table 5.4 compares the switching power consumed by the gates at 195^{o}C.

Gate	Power Consumption (uW) (195^{0}C)
Inverter	0.088
2 Input NAND	0.242
3 Input NAND	0.319
4 Input NAND	0.44
2 Input NOR	0.286
3 Input NOR	0.451
4 Input NOR	0.649

Table 5.4 Power Consumption

From the above table it is clear that NAND gates consume less power than equivalent NOR gates. Minimizing the usage of NOR Gates has improved the power efficiency of the cell library by 30%.

Chapter 6

Conclusion and Future Work

6.0 Conclusion:

Standard cell libraries have revolutionized the method of ASIC design. It has increased the efficiency and decreased turn around time compared to conventional digital circuit design. In this thesis a method to design a standard cell library that can be used to construct ultra-high temperature CMOS application specific integrated circuits (ASIC) is presented.

The library is comprised of over 90 cells including both logic and I/O pads. Each cell in the library is characterized using OCEAN script for delay using linear delay model, which allows easier characterization of the cells with a limited number of simulations, while still achieving reasonable accuracy.

A calibration model is developed which ensures accurate hardware delay calibration of the library while reducing the number of cell measurements by a factor of six. The calibration model was fabricated in Silicon on Sapphire (SOS) and tested at 195^0C to verify the delay model developed.

It was observed that the measured data had an excellent fit with the models developed, the parallel structures fit with better accuracy compared to the series structures. The

intrinsic and transition delay of combinational logic and propagation delay of I/O circuits, were estimated for 5 9s confidence levels, while setup and propagation delay of sequential logic and were estimated for 3 9s confidence levels.

The measured delay was statistically significant and 30 % faster than the typical peregrine model at 195^0C. The power consumption of the NAND gates was 30% lower than equivalent NOR gates for which their usage was maximized. The cell library works reliably from 3.3 to 1.2 supply voltage at 195^0C.

6.1 Future Work:

The future work would be extending the measured data confidence level of sequential logic from 3 9s to 5 9s by performing more statistical data measurements. More investigation needs to be done on the behavior of transistors in series. Scan cells are required for fault detection and to improve testability of complex logic designs, hence they could to be included in the library. To improve the efficiency of the library for building complex designs, mega cells such as; 1 bit register file slice, a 1-Bit ALU, p-decoder, UART, PIO/PIA Microcontroller core, fixed point DSP core, FIFO, SRAM should be included in the existing library. To achieve higher speed and lesser area the transistors can be scaled and single and double height cells can be implemented. The discussed linear delay model can be extended to non-linear delay model for achieving better timing accuracy.

References

[1] Peter McAdam and Bar-Giora Goldberg, "*CMOS SOS for MIXED Signal ICs*", Peregrine Semiconductor Corporation, June 2001.

[2] Jos B. Sulistyo and Dong S. HA, "*A New Characterization Method for Delay and Power Dissipation of Standard Library Cells*", VLSI Design, Volume:15(3), pp. 667-678, 30 January 2002.

[3] Mehmet A. Cirit, "*Characterizing a VLSI standard cell library*", Proceedings of Custom Integrated Circuit Conference, IEEE, pp. 25.7.1-25.7.4, 1991.

[4] Dhimant Patel, "*CHARMS: Characterization and Modeling System for Accurate Delay Prediction of ASIC Designs*" Proceedings of Custom Integrated Circuits Conference, IEEE, pp. 9.5.1-9.5.6, 1990.

[5] Eshraghian, K. and Weste, "*Principles of CMOS VLSI Design: A Systems Prespective*", Addison-Wesley, Reading, M.A, 1994.

[6] Synopsys Inc., "*Library Compiler: Modeling Timing and Power*", Chapter II , 2003.

[7] Jing-Yang Jou, Jing-Yuan Lin and Wen-Zen Shen, "*A Structure-Oriented Power ModelingTechnique for Macrocells*," IEEE Transactions on VLSI, Vol. 7 No. 3, pp. 380-391, 1999

[8] Jing-Yang Jou, Jing-Yuan Lin and Wen-Zen Shen, "*A Power Modeling and Characterization Method for the CMOS Standard Cell Library*," Digest of Technical Papers, International Conference on Computer Aided Design, IEEE, pp. 400-404, 1990

[9] Sung-Mo Kang and Yusuf Leblebici , "*CMOS Digital Intergrated Circuits: Analysis and Design*" , McGraw-Hill, Second Edition, 1999.

[10] Farid N. Najm "*Power Estimation Techniques for Integrated Circuits*" Proceedings of International Conference on Computer Aided Design, IEEE , 1995.

[11] Alessandro Bogliolo, Luca Benini, Burno Ricco "*Power Estimation of Cell-Based CMOS Circuits*" " Proceedings of 33rd Design Automation Conference, 1996.

[12] Dr. Louis Johnson, "*Designing Complex Gates*", Digital VLSI Design, pp 13-16, March 2003.

[13] Jiing-Yaun Lin, Tai-Chen Liu and Wen-Zen Shen, "*A Cell-Based Power Estimation in CMOS Combinational Circuits* " Proceedings of ACM, 1994.

[14] R.W. Broderson, et. al., "*LagerIV Cell Library Documentation"*, Electronics Research Laboratory, University of California, Berkeley, June, 1988.

[15] W.Agaststein, K. McFaul, P.Themins, "*Validating an ASIC Standard Cell Library"*, Intel Corporation, 1990.

[16] Rung-Bin Lin,Isaac Shuo-Hsiu Chou, Chi-Ming Tsai, "*Benchmark Circuits Improve the Quality of a Standard Cell Library* ", Proceedings of the 1999 IEEE Conference, 1999, pp 173-176.

[17] Rohini Gupta, Byron Krauter, Bogdan Tutuianu, John Wills and Lawerence T.Pileggi, "*The Elmore Delay as a Bound for RC Trees with Generalized Input Signals*" Proceedings of the 32nd Design Automation Conference, IEEE , 1995.

[18] Yunbum Jung, "*Automated Standard Cell Library Generation and Study of Cell Library Functional Content*" ,University of Michigan, pp 1-12.

[19] Denis Flandre and Jeane-Pierre Colinge, "*High-Temperature Characteristics of CMOS Devices on Silicon-on-Insulator(SOI) Substrates*" ,Micro Electronics Labratory, Universite Catholique de Louvain, 2004.

[20] Kaushik Roy, Saibal Mukhhopadhyay and Hamid MahmoodI-Memimand, "*Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits*", Proceedings of the IEEE, Vol. 91, No. 2, February 2003.

[21] Michael John and Sebastian Smith, "*Application-Specific Integrated Circuits*", Addison-Wesley Publishing Company, VLSI Design Series, June 1997.

[22] J.E. Horstmann, H. W. Eichel and R. L. Coates, "*Metastability Behavior of CMOS ASIC Flip-Flops in Theory and Test*", IEEE Journal of Solid-State Circuits, Vol. 24, No.1 Feb 1989.

[23] Jay L. Devore, "*Probability and Statistics for Engineering and the Sciences*", Brooks/Cole Publishing, Third Edition. 1991.

[24] Narendra Kayathi, "*An Extended Model for SOS for Elevated Temperature for Analog and Digital Designs*", Oklahoma State University, Jan 2005.

Appendix A

Standard Cell Library List

The naming convention followed in the cell library is divided into two categories. The sequential logics follow 7 character naming convention and combinational logics have 8 character naming convention. Each cell has a unique naming for its identification. The naming methodology and cell library list is as follows,

A.1 Naming convention for Sequential logics.

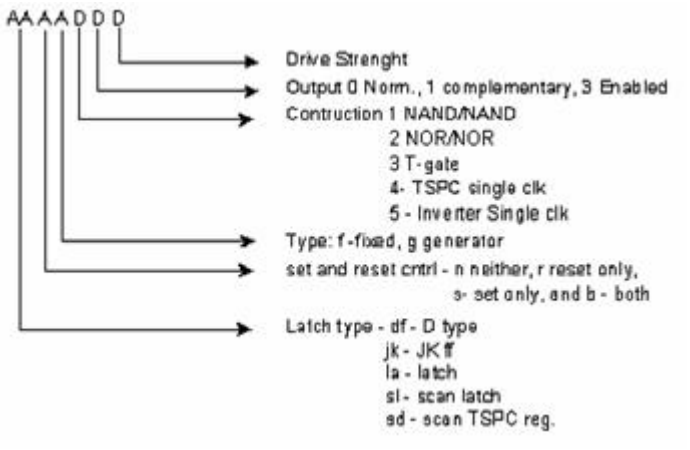

Figure A.1 Naming Convention for Sequential Logic

A.2 Naming convention for other Boolean logics.

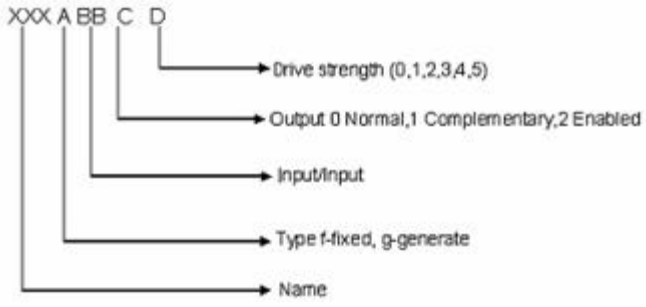

Figure A.2 Naming Convention for Combinational Logic.

A.3 Simple Gates

Cell Name	Description
andf2001	2 Input AND
andf3001	3 Input AND
andf4001	4 Input AND
aorf2201	2x2 AND-OR
aorf2301	2x3 AND-OR
aorf3201	3x2 AND-OR
aorf3301	3x3 AND-OR
aorf4201	4x2 AND-OR
aorf4301	4x3 AND-OR
aoif2201	2x2 AND-OR-INV
aoif2301	2x3 AND-OR-INV
aoif3201	3x2 AND-OR-INV
aoif3301	3x3 AND-OR-INV
aoif4201	4x2 AND-OR-INV
aoif4301	4x3 AND-OR-INV
buff1001	1X Buffer
buff1002	2X Buffer
buff1003	3X Buffer
buff1004	4X Buffer
buff1005	5X Buffer
buff1021	1X 3-State-Buffer

buff1022	2X 3-State-Buffer
buff1023	3X 3-State-Buffer
buff1024	4X 3-State-Buffer
buff1025	5X 3-State-Buffer
fuaf3001	Full adder
fusf3001	Full sub.
haaf2001	Half Adder
has2f001	Half Subtractor
invf1001	1X Inv
invf1002	2X Inv
invf1003	3X Inv
invf1004	4X Inv
invf1021	1X 3-State-Inv
invf1022	2X 3-State- Inv
invf1023	3X 3-State- Inv
muxf2101	2 In to 1 Select MUX
muxf2201	2 In to 2 Select MUX
muxf4101	4 ln to 1 select mux
muxf4201	4 In to 2 Select MUX
muxf4401	4 In to 4 Select MUX
nanf2001	2 Input NAND
nanf2002	2X 2 Input NAND
nanf3001	3 Input NAND

nanf4001	4 Input NAND
nadf2011	2 Input NAND/AND
nadf3011	3 Input NAND/AND
nadf4011	4 Input NAND/AND
aobf2001	A OR B not decoder
aabf2001	A not AND B decoder
norf2001	2 Input NOR
norf2002	2 Input 2X NOR
norf3001	3 Input NOR
norf4001	4 Input NOR
norf2011	2 Input NOR/OR
norf3011	3 Input NOR/OR
norf4011	4 Input NOR/OR
oaif2201	2x2 OR-AND-INV
oaif2301	2x3 OR-AND-INV
oaif3201	3x2 OR-AND-INV
oaif3301	3x3 OR-AND-INV
xnof2001	2 Input XNOR
xorf2001	2 Input XOR
vddf001	Pull up
vssf001	Pull down

A.4 Latches and Flip Flops

Cell Name	Description
dfnf101	D-ff with Q
dfnf111	D-ff with Q & Qbar
dfrf101	D-ff with Q & asyn. RESET
dfrf111	D-ff with Q, Qbar & asyn. RESET
dfsf101	D-ff with Q & asyn. SET
dfsf111	D-ff with Q, Qbar & asyn. SET
dfbf111	D-ff with Q, Qbar & asyn. SET,RESET
dfbf111	D-ff with Q, Qbar & asyn. SET,RESET
srbf111	Cross-coupled SET/RESET Latch NAND type
srbf211	Cross-coupled SET/RESET latch NOR type
lanf101	Latch with Q no SET and RESET
lanf111	Latch with Q and Qbar, no set and reset
larf101	Latch with Q and async RESET
larf111	Latch with Q, Qbar and async RESET
lasf101	Latch with Q and async SET
lasf111	Latch with Q, Qbar and async SET
labf101	Latch with Q ,set and reset
labf111	Latch with Q and Qbar, both set and reset

A.5 Padframe cells

Cell Name	Description	Comment
vddc000	Pos. power	vdd power
vssc000	Neg. power	vss power
padn000	Simple straight thur pad NO protection	
padi000	Input pad with protection	ESD Diodes
fillc000	Frame filler cell with Dec. Cap	
cornc000	Pad Corner Dec. Cap	
corn000	Pad Corner NO Cap	
trici050	3-State-Pad Driver.	50pf drive strength
outci050	Pad Driver	50pf drive strength

XXX- Name, n - no cap or protection, i - input protection, c cap present.

Appendix B

OCEAN SCRIPT

The spectre simulation can be automated by making spectre script files. These scripts, called **OCEAN** (Open Command Environment for Analysis) scripts have C-like syntax. They are used to save and load the simulation for future use and modification. The scripts are also used to make the synopsis and verilog library files. An example of inverter OCEAN script is as shown below:

```
/*********************************************************

  INVERTER LOGIC GATE EXTRACTION

*********************************************************/

printf("\n\n1XINVERTER LOGIC GATE EXTRACTION\n\n")

/*Standard Simulator Setup*/
/*****************************************/
simulator( 'spectre )
design(    "~/simulation_per/invf1001_test/spectre/schematic/netlist/netlist")
resultsDir( "~/simulation_per/invf1001_test/spectre/schematic" )
modelFile(
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/include.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/inSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/nlSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/rnSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/inRFSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/in68RFSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/rnRFSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/rn68RFSlow.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/nlRFSlow.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/nl68RFSlow.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/ipSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/plSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/rpSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/rpRFSlow.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/rp68RFSlow.inc" "")
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/ipRFSlow.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/ip68RFSlow.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/plRFSlow.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/SLOW/pl68RFSlow.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/NOMINAL/psNominal.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/NOMINAL/npNominal.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/NOMINAL/ppNominal.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/NOMINAL/snNominal.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/NOMINAL/mmmNominal.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/NOMINAL/type1Nominal.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/NOMINAL/type2Nominal.inc" "")
 '("/export/opt/Peregrine/Rel_2.9/fa/models/spectre/NOMINAL/type3Nominal.inc" "")
```

```
'( "/export/opt/Peregrine/Rel_2.9/fa/models/spectre/NOMINAL/type4Nominal.inc" "")
)
analysis('tran ?stop "200n"  )
temp( 195 )
save( 'v "/A" "/Y" )
/*****************************************/


/* VARIABLE SETUP */
stime = 1n          /* Time Units */
cparam = 1p         /* Cap Units */
tedge = 1n          /* Nominal input Rise/Fall Time */
CINV = 30f          /* Unit Inverter Input Cap */
Cload = CINV/cparam      /* Unit load Cap */
vrail = 3.3

desVar(    "clnum" 1          )
desVar(    "Cload" CINV       )
desVar(    "vdc" vrail        )
desVar(    "tedge" 1n         )
desVar(    "tpulse" 25n       )

declare(cl_rise[5])
declare(cl_fall[5])
declare(ss_rise[5])
declare(ss_fall[5])

/* FILE OPEN (append mode) */
dat_file = outfile("~/extracted_per/inv05.dat" "a" )

fprintf(dat_file "\ncell(invf1005) {\n")
fprintf(dat_file "  area : 2;\n")
fprintf(dat_file "  cell_footprint : \"inv05\";\n")
fprintf(dat_file "  pin(A) {\n\tdirection : input;\n\tcapacitance : 0.017;\n\t}\n")
fprintf(dat_file "  pin(Y) {\n\tdirection : output;\n")
fprintf(dat_file "\tfunction : \"A'\";\n")
fprintf(dat_file "\ttiming() {\n")

printf("\n\n***** reached this point\n\n")
/* SIMULATION & EXTRACTION */

/*****************************************/
  ;Calculate Intrinsic Rise/Fall
  printf("\n\n***** Calc i_rise/i_fall\n\n")
  desVar(              "clnum" 0          ) ;Setup zero load
  analysis('tran ?stop "200n"  )
  run()
  selectResult( 'tran)
  int_rise = (cross(VT("/Y"),0.5*vrail,2,"rising")-cross(VT("/A"),0.5*vrail,2,"falling"))/stime
  int_fall = (cross(VT("/Y"),0.5*vrail,2,"falling")-cross(VT("/A"),0.5*vrail,2,"rising"))/stime

  cl_rise[0] = int_rise
  cl_fall[0] = int_fall

/*****************************************/
/*****************************************/
```

```
  ;Calculate Rise/Fall Resistance
  printf("\n\n***** Calc rise_res/fall_res\n\n")
  av_rise = 0.0
  av_fall = 0.0
  analysis('tran ?stop "200n"  )

  for(rnum 1 4
          desVar(    "clnum" rnum    )
          run()
          selectResult( 'tran)
          cl_rise[rnum] = (cross(VT("/Y"),0.5*vrail,2,"rising")-cross(VT("/A"),0.5*vrail,2,"falling"))/stime
          av_rise = av_rise + cl_rise[rnum] - cl_rise[rnum-1]

          cl_fall[rnum] = (cross(VT("/Y"),0.5*vrail,2,"falling")-cross(VT("/A"),0.5*vrail,2,"rising"))/stime
          av_fall = av_fall + cl_fall[rnum] - cl_fall[rnum-1]
plot( v("/Y") )
  );end for rnum

  av_rise = av_rise / (4 * Cload)
  av_fall = av_fall / (4 * Cload)


/*****************************************/
/*****************************************/
  ;Calculate Slope Sensitivity
  printf("\n\n***** Calc Slope Rise/Fall\n\n")
  as_rise = 0.0
  as_fall = 0.0
/*****************************************
  ss_rise[0] = int_rise
  ss_fall[0] = int_fall

  analysis('tran ?stop "100n"  )
  desVar(             "tpulse" 20n      )
  desVar(             "clnum" 0         )
  tedge = 1n

  for(rnum 1 3
          desVar(    "tedge" tedge     )
          run()
          selectResult( 'tran)
          ss_rise[rnum] = riseTime(VT("/Y") 10n t 30n t 20 80) / stime
          as_rise = as_rise + ss_rise[rnum] - ss_rise[rnum-1]

          ss_fall[rnum] = riseTime(VT("/Y") 30n t 55n t 20 80) / stime
          as_fall = as_fall + ss_fall[rnum] - ss_fall[rnum-1]
          tedge = tedge + 1n


  );end for rnum

  as_rise = as_rise / 3
  as_fall = as_fall / 3


*****************************************/
```

```
/* DATA DUMP */
/*****************************************/
;plot(getData("/A") getData("/Y") )
  printf("Intrinsic Fall = %g\n" int_fall)
  printf("Intrinsic Rise = %g\n" int_rise)
  fprintf(dat_file "\t  intrinsic_rise : %g ;\n" int_rise)
  fprintf(dat_file "\t  intrinsic_fall : %g ;\n" int_fall)

  printf("Rise Resistance = %g\n" av_rise)
  printf("Fall Resistance = %g\n" av_fall)
  fprintf(dat_file "\t  rise_resistance : %g ;\n" av_rise)
  fprintf(dat_file "\t  fall_resistance : %g ;\n" av_fall)

  printf("Rise Sensitivity = %g\n" as_rise)
  printf("Fall Sensitivity = %g\n" as_fall)
  fprintf(dat_file "\t  slope_rise : %g ;\n" as_rise)
  fprintf(dat_file "\t  slope_fall : %g ;\n" as_fall)

  fprintf(dat_file "\t  related_pin : \"A\";\n")

/*****************************************/
/*****************************************/

/* Closing brackets */
  fprintf(dat_file "\t}\n  }\n}\n")


/* CLOSE FILE */
close(dat_file)
```

This script can be run from the CIW (from the > prompt) using the command > load "*path*/inv_test.ocn".

VITA

Venkataraman Jeyaraman

Candidate for the Degree of

Master of Science

Thesis: Design, Characterization and Automation of Ultra-High Temperature Standard Cell Library for Harsh Environments

Major Field: Electrical and Computer Engineering

Biographical:

Personal Data: Born in Madurai, India on May 21st, 1980, the son of V.Jeyaraman and S.K.Janaky.

Education: Graduated from higher secondary school in India, received a bachelors degree from University of Madras in India in May 2001 in the field of Electronics and Communication Engineering. Completed the requirements for the Master of Science degree with a major in Electrical and Computer Engineering at Oklahoma State University in December, 2004.

Experience: Worked as a Design Engineer intern at Halliburton Energy Services from May 2004 to October 2004

Worked as a Research Assistant in MSVLSI Group in Department of Electrical Engineering at Oklahoma State University from August 2002 to December 2004.