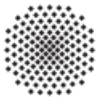

Universität Stuttgart

Institute of Computer Architecture and
Computer Engineering

University of Stuttgart
Pfaffenwaldring 47
D-70569 Stuttgart

Master Thesis Nr. 00731-001

Realistic gate model for efficient timing analysis of very deep submicron CMOS circuits

Deepthi Murali

Course of Study:	Master of Science in Information Technology
Examiner:	Prof. Dr. rer. nat. habil. Hans-Joachim Wunderlich
Supervisor:	Dipl.-Inf. Marcus Wagner Dipl. -Inf. Eric Schneider Dr. rer. nat. Michael Kochte
Commenced:	2015-09-14
Completed:	2016-03-15
CR-Classification:	B.8.2, C.4, J.6

Erklärung:

Ich versichere, nach bestem Wissen und Gewissen, dass diese Arbeit kein Material und keine Quellen enthält, die bereits für das Erlangen eines akademischen Grades unter meinem Namen oder anderen Namen an einer weiteren Universität bzw. Hochschuleinrichtung veröffentlicht wurde, außer Quellen die als solche gekennzeichnet sind. Zusätzlich versichere ich, dass keine Teile dieser Arbeit in meinem Name in Zukunft eingereicht werden, um einen akademischen Grad an einer anderen Universität oder Hochschuleinrichtung zu erlangen ohne der Zustimmung der Universität Stuttgart.

Unterschrift:
Stuttgart, den

Declaration:

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Stuttgart.

Signature:
Stuttgart,

Acknowledgement

I would like to take the opportunity to thank a number of people as I am presenting the results of my Master Thesis.

I would first like to thank Prof. Dr. Hans-Joachim Wunderlich for providing me the opportunity to do research at the institute. The regular meetings and feedback provided by him were very crucial to the completion of this thesis. Next, I would like to express my heartfelt gratitude to my supervisors Mr. Marcus Wagner , Mr. Eric Schneider and Mr. Michael Kochte. They took sincere efforts to keep me motivated so that I achieve all the goals of my thesis. Without their advice and guidance, I would have definitely been lost and I am very happy to have got them as my supervisors. I am also thankful to all other staff of the institute for their friendly attitude as it made me really comfortable to work at the institute.

I would specially like to thank the System Administrators, Mr. Helmut Häfner and Mr. Lothar Hellmeier, as they provided immediate help with all technical issues which were extremely important during the course of the thesis.

Last but not the least, I would like to thank my family and friends for extending their support throughout the duration of my thesis. Without their support, it would have been very difficult to go through all the good and bad times of my Thesis.

Abstract

The continuously shrinking technology has made it possible for designers to incorporate more functionality with better performance at a much higher density in Integrated Circuits(ICs). Fast and accurate timing simulation of such large circuit designs using ever more complex transistor models has become a challenging problem. In modern circuits, the gate delay is severely affected by process variations, environmental variations and cross talk. Moreover, technology scaling has also resulted in significant increase in interconnect parasitics (including resistors and capacitors) which can dramatically reduce the performance of a circuit.

For the circuit design validation and delay test evaluation, the industry has long relied on fast gate-level timing simulators like ModelSim to validate the designs. However, with continued scaling and steadily increasing circuit performance requirements, gate level simulators can no longer provide acceptable simulation accuracy. On the other hand, circuit level SPICE simulation provides acceptable accuracy but at a very large computational cost. To provide a suitable trade-off between the accuracy of the SPICE simulation and the speed of the gate level simulation, this thesis proposes a realistic gate model which can be used for the fast and accurate timing simulation of circuits to analyze their timing behaviour.

In this thesis, a heterogeneous gate model that combines a simple gate model like Non-Linear Delay Model (NLDMs) and an advanced current source model (CSM) using a classifier is proposed. The simple gate model allows fast timing simulation and gives acceptable accuracy in many cases while the advanced gate model always provides more accurate and reliable results, but at a much higher computational cost. The classifier is designed to choose the advanced gate model depending on special cases (eg, multiple input switching) where the simple gate model gives inappropriate results. This heterogeneous gate model is further applied to develop a circuit simulator that enables fast and accurate post-layout and delay fault simulation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Goals of this Work

The design optimizations, validation and delay test methodologies of modern day Integrated Circuits (ICs) require accurate and fast timing analysis techniques. With the steadily increasing performance requirements of modern circuits, even the smallest inaccuracy in the delay prediction can be catastrophic - for both asynchronous and synchronous circuits. Ideally, the circuit would be simulated in the most detailed level possible viz, the circuit level simulation. The most commonly used tool for circuit level timing simulation is SPICE. However, this is extremely slow. For example, to evaluate the effectiveness of delay tests, it is necessary to perform accurate timing simulation of a large number of input vector pairs.

Conventionally, the industry relied on certain gate-level simulation tools like ModelSim, for the validation of the new design and for delay test evaluation. For this, it used the delay information from the Standard Delay Format (SDF) files. However, with the continued down-scaling of technology, the delay of a logic gate has become severely affected by various new factors like cross-talk, noise, process-variations, etc. Another important effect that dominates the delay of the circuit is the impact of interconnect parasitics, which increases dramatically with growing technology generations [1]. Therefore, the very simple and abstract gate models used by gate-level timing simulation have become very inaccurate. A new gate models on transistor level must be designed that incorporate all these factors in order to give fast and accurate simulation results.

Moreover, with continuous technology scaling, circuits have become more prone to defects, e.g. a wire with an extremely large resistance or a very small resistance between neighbouring wires, which were supposed to be completely isolated. To detect such a fault, we apply a vector pair to the circuit and check if the time of the last transition at the output is smaller than the clock cycle time. This is referred to as delay fault simulation. When there are large number of vector pairs, numerous SPICE simulations of the circuit are required to be carried out and these have to be extremely fast. However, circuit simulation program like SPICE use very complex and accurate MOSFET models like BSIM4 that has hundreds of transistor parameters and hence, this method of simulation proves to be computationally very expensive. Added to this, SPICE simulation also results in the use

of a lot of CPU time and memory.

The gate model used by a circuit simulator is of critical importance as it determines the speed and the accuracy of the simulation. A gate model is basically a mathematical model that is composed of several transistor parameters and a number of equations representing the relationship between these parameters. In SPICE, these equations are used to evaluate the voltage/current at any given node. This complex modelling of gates in SPICE had a number of restrictions which included [2]:

- No approximations made to the device models
- Global convergence to the solution at every time instant

This meant that a lot of time is utilized even in the latent circuit parts resulting in exorbitantly large simulation time especially for large circuits. Thus, high levels of integration along with the shrinking technology demanded the need for a much simpler gate model which could simulate circuits much beyond the capacity limits of conventional SPICE simulators. This led to compromising the accuracy of SPICE simulations and to come up with efficient gate models with much accelerated simulation results.

Traditionally used gate models like Non-Linear Delay Models (NLDMs) are very fast but these are inaccurate due to the crude representation of the input and output signal waveforms. Also, as the effect of interconnect resistance and capacitance increases, the NLDM models do not provide good accuracy as they consider the load of a gate to be purely capacitive. Therefore, more complex models like Current Source Model (CSM) are developed which provide better accuracy by capturing and modelling the non-linear output current of the gate.

This thesis aims at developing a realistic gate model for fast post-layout simulation of circuits and to explore the various trade-off possible between the simulation accuracy, simulation speed and scalability. The proposed gate model is developed by combining a simple but fast gate model and an advanced but a slower (and more complex) gate model into one. This exploits the speed provided by the simple gate model and the accuracy provided by the advanced gate model.

1.2 Organization of the Thesis

The reminder of this thesis is organized into the following chapters.

Chapter 2 gives a brief description of the relevant concepts and definitions which will aid in the better understanding of the thesis.

The first part of Chapter 3 gives a wide insight into the different types of gate models in general as the aim of this thesis is to develop a gate model. It also analyses the advantages and the limitation of the various gate models and thus helps in the choosing of the simple and the advanced gate models which are later required to develop the proposed heterogeneous model. The latter part of this chapter analyses the various interconnect model that are used to model the interconnect resistance and capacitances.

Chapter 4 focuses on the detailed description of the design of the proposed heterogeneous gate model. It explains the various design details of the simple and advanced gate model and their implementation.

Chapter 5 explains how the proposed gate model in this thesis can be applied for the fast and accurate simulation of circuits. This chapter also discusses the method to simulate the interconnects in a circuit.

Chapter 6 explains the experimental setup used for this thesis. The first part of this chapter gives a detail description of the circuits chosen for validation of the designed gate model. The next part explains the details of the reference reference simulation with a fast gate-level and a very accurate circuit-level simulator. The next part involves the experimental setup used for the circuit simulator using the proposed gate model. The final part of this chapter discusses the simulation results of the implemented circuit simulator.

Finally, Chapter 7 discusses the possible future work and concludes the thesis.

Chapter 2

Fundamentals of VLSI Circuit Modelling and Simulation

This chapter mainly focusses on the relevant concepts and definitions which are essential for the better understanding of the ideas presented in this thesis.

2.1 Electrical Description of VLSI circuit

2.1.1 Elementary Devices

Resistor

A resistor is a device which impedes the flow of current through a circuit. The symbol of a resistor is as shown below:

Figure 2.1: Symbol of a resistor

The unit for measuring resistance in **OHM** (represented by the Greek symbol Ω).

Resistors have a linear current-voltage relationship [3] as stated by *Ohm's* law,

$$V = IR \tag{2.1}$$

where,

V is the voltage across the resistor,

I is the current flowing through the resistor.

Capacitor

A capacitor is a device that stores energy in the form of an electric field [4]. The most common type of capacitors are the parallel plate capacitors, which is made of 2 parallel plates separated by a dielectric material. When a power source forces the electric charge into the plates of the capacitor, the capacitor stores energy, resulting in a voltage V across it. The quantity capacitance is defined as the measure of electric charge (Q) required to build 1 unit of voltage (V) across the plates of the capacitor and it is given by :

$$C = \frac{Q}{V} \tag{2.2}$$

Capacitors may be polarised or non-polarised. Non-polarised capacitor is the one that has no implicit polarity and it can be connected either way in a circuit. A polarised capacitor is the one that has implicit polarity, i.e, it can be connected only in one way in a circuit.

The unit of capacitance is Farad and the symbol of a non-polarised and polarised capacitor is as shown below:

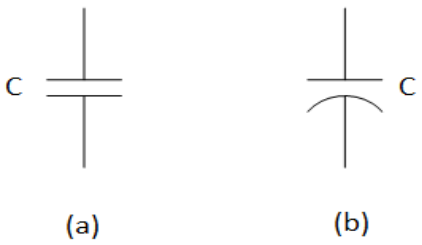

Figure 2.2: (a)Symbol of a non-polarised capacitor (b)Symbol of a polarised capacitor

A variable capacitor or a voltage controlled capacitor is the one whose capacitance can be controlled with the voltage applied across its terminals. Symbol of a variable capacitor is as shown:

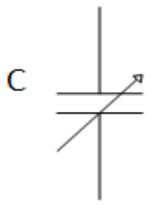

Figure 2.3: Symbol of a variable capacitor

2.1.2 Voltage and Current sources

There exists mainly 2 kinds of energy sources in electronics - voltage source and current sources.

DC Voltage source

An ideal independent voltage source provides the same voltage across its terminals irrespective of the current through its terminals. Such a voltage source may produce a constant dc output voltage.

The symbol of a DC voltage source is as shown below:

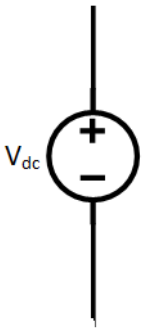

Figure 2.4: Symbol of a DC voltage source with a dc voltage of V_{dc}

DC Current Source

An ideal DC current source produces a constant dc current across its terminals no matter the voltage required across its terminals. The symbol of such a current source is as shown below:

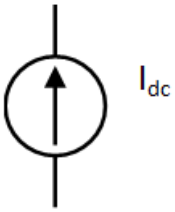

Figure 2.5: Symbol of a DC current source with a dc current of I_{dc}

Voltage controlled DC-current source

A voltage controlled dc current source is the one that produces a DC current at its terminals as a function of a voltage in the circuit. These type of current sources are referred to as dependent current sources. The symbol of such a source is as shown below:

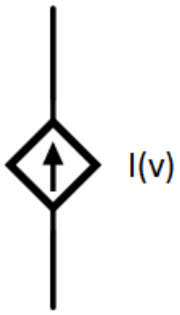

Figure 2.6: Symbol of a dependent current source I(v)

Ground

The *ground* is nothing but a circuit node to which all the voltages in a circuit are referenced to. The commonly used circuit symbols to represent the ground node are :

Figure 2.7: Symbol of ground in a circuit

2.2 MOSFETs

Metal Oxide Semiconductor Field Effect Transistor(MOSFET) is a type of voltage controlled Field Effect transistor. A MOSFET is a three terminal device with gate, source and drain and both the n-channel(NMOS) and p-channel(PMOS) MOSFETs are available. MOSFET devices exists in 2 mains forms :

- Depletion Type - In this type of MOSFETs, a Gate-Source voltage (V_{GS}) is required to turn "OFF" the device, i.e, these MOSFETs are similar to a "Normally Closed" switch.
- Enhancement Type - These type of MOSFETs require a Gate-Source voltage (V_{GS}) to turn the device "ON", i.e, similar to a "Normally Open" switch.

The symbol of NMOS and PMOS transistors are as shown below:

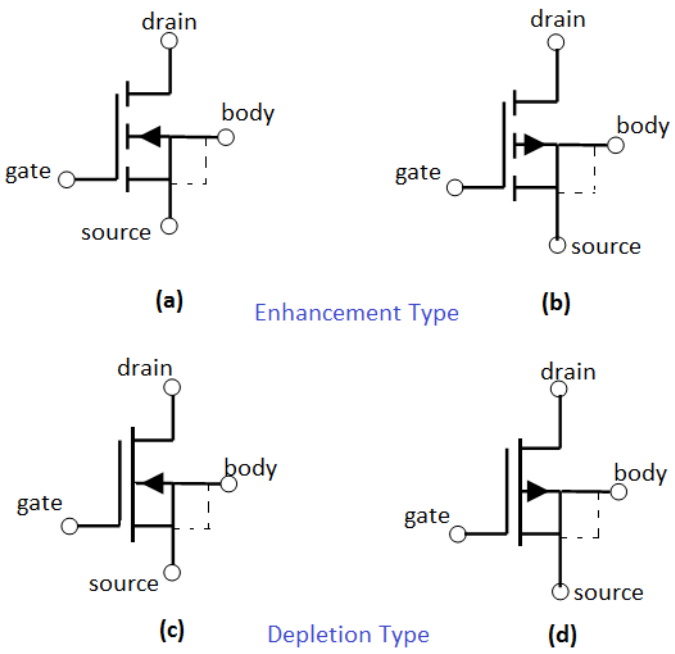

Figure 2.8: (a)n-channel enhancement MOSFET (b) p-channel enhancement MOSFET (c)n-channel depletion MOSFET (d) p-channel Depletion MOSFET

2.2.1 Physical Structure

The MOSFET strcture shown in fig 2.9 is fabricated on a p-type substrate which is marked as the Body. The source and the drain regions are created by two heavily doped n-type regions. A thin layer of an insulator (usually silicon dioxide, SiO_2) is grown on the substrate, in the areas between the source and the drain [5]. A metal layer is then deposited on the oxide layer to form the gate terminal. Metal contacts are also made at the source, drain and the substrate (also called bulk or body) to bring out 4 terminals in total.

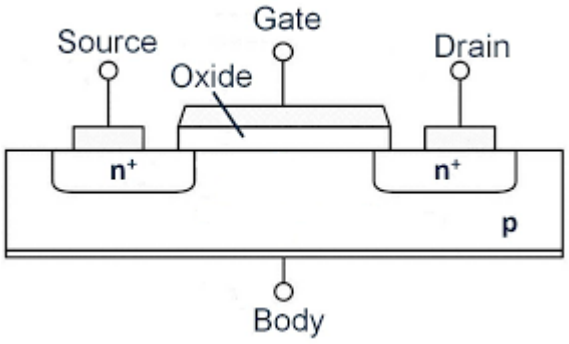

Figure 2.9: Structure of a n-channel MOSFET [6]

The substrate is generally connected to the source terminal. This is because, the substrate forms *pn* junctions with the source and the drain regions and normally, these junctions

are kept reverse biased. Since, the drain terminal is at a higher potential when compared to the source, these junctions can be cut-off by connecting substrate to the source. Hence, the substrate has no effect on the operation of the MOSFET and hence the reason why MOSFET is treated as a three terminal device.

2.2.2 Device Operation

Depending on whether the MOSFET is a depletion type or an enhancement type, the voltage applied to the gate controls the flow of current between the drain and the source. In general, conduction in a MOSFET happens when a conducting channel is formed between the drain and source regions. For an n-channel MOSFET, the channel is formed by negatively charged electrons and for a p-channel MOSFET the holes form the conducting channel.

In case of an enhancement type MOSFET, the conducting channel is very lightly doped. So, when there is no voltage applied to the gate terminal, the channel is non-conducting and hence it is in normally "OFF" state. This can be thought of as 2 back-to-back diodes that do not conduct even when a voltage (V_{DS}) is applied between the drain and the source, as the path resistance between them is of the orders of 10^{12} ohm [5]. For an n-channel enhancement MOSFET, when a positive voltage is applied at the gate terminal with respect to the source terminals (V_{GS}), negatively charged electrons are attracted towards the gate, hence enhancing the channel thickness. At a point when V_{GS} is greater than the **threshold voltage** V_{th}, the channel becomes conducting with sufficient number of charge carriers. When a small positive drain-source voltage V_{DS} is applied, current I_d starts flowing through the channel formed between the source and drain and hence the MOSFET becomes conducting. The conductance of the channel increases with the excess gate voltage (V_{GS} - V_{th}) which is also known as the **overdrive voltage**.

When the drain source voltage V_{DS} is increased further, keeping the V_{GS} a constant, drain current I_d increases. The voltage V_{DS} appears as a voltage drop across the source and drain terminals, and therefore the voltage between the gate and the points across the channel varies from V_{GS} at source terminal to (V_{GS}-V_{DS}) at the drain terminal. Thus, the voltage across the length of the channel and the depth of the channel varies. The channel is deeper at the source end and narrows down towards the drain terminal. So, with increasing V_{DS}, I_d increases. At a particular value of V_{DS}, called the **pinch off voltage**, the channel depth at the drain end decreases to almost zero. Increasing V_{DS} beyond this point has no much effect on the shape of the channel and thus the drain current I_d saturates.

For a p-channel MOSFET, the reverse is true. When V_{GS} is 0, the device is "OFF". When a negative V_{GS} is applied to the gate terminal, the holes are accumulated in the channel and the device turns "ON", while a positive V_{GS} will turn the device "OFF".

The depletion mode MOSFET device is a less common type, which is normally "ON" when there is no gate voltage applied. For a n-channel depletion MOSFET, positive V_{GS} increases the gate current while a negative gate voltage -V_{GS} depletes the majority charge carriers which are electrons and hence the drain current decreases and eventually the device turns "OFF". Similarly, for a p-channel MOSFET, the opposite holds true.

The enhancement MOSFETs are the mostly used in integrated circuits (ICs) to produce the CMOS logic gates. This is mainly because of their low "ON" resistance and extremely high "OFF" resistance and their infinitely high input resistance which makes them excellent electronic switches. CMOS stands for Complementary MOS which consists of both PMOS and NMOS within its design.

2.3 SPICE

Simulation Program with Integrated Circuit Emphasis (SPICE) is a general purpose analog electronic circuit simulator [7]. It is a circuit simulation program used in IC designing in order to check for the integrity of a design and to predict the behaviour of a circuit.

As SPICE tools provided the best possible accuracy of results, they were mainly useful in the following areas :

- Validation of timing of critical paths in circuits
- Characterization of cell libraries
- Modelling and simulation of precision analog components

SPICE simulators are essential for the evaluation, analysis and optimization of any integrated circuit (IC) design [8, 9]. The inputs to a SPICE simulators includes a description of the circuit in the form of a netlist, a set of device models which which define the electrical behaviour of the various electronic devices used and a set of input signals.

Generally, a SPICE simulation has three main steps [10] :

- Generating the required input netlist file,
- Running the SPICE simulation,
- Measurement and analyses of the simulation results.

2.4 Description of a Spice Netlist

A SPICE netlist file contains a description of an electronic circuit and consists of a number of statements defining each circuit element present in the circuit. For example, consider a RC circuit with a voltage source at the input which is as shown in the figure below:

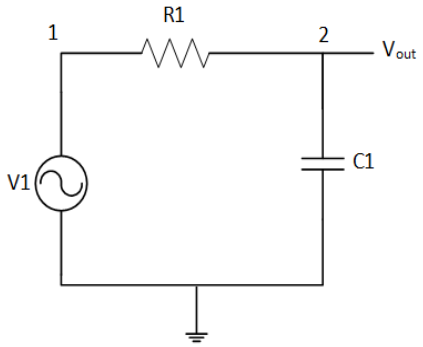

Figure 2.10: Transient Analysis of RC circuit in SPICE

Here the circuit netlist can be written as :

```
*Spice Netlist of a simple RC circuit
V1 1 0 ac 1 sin(0Voff 1Vpeak 2KHZ)
R1 1 2 30
C1 2 0 100u
.TRAN  5us  500us
.print tran v(1) v(2)
.end
```

The first line is always interpreted as a comment. **R1** and **C1** are the description of the Resistor and the Capacitor elements of the circuit. The resistor is between nodes 1 and 2 and has a value of 30 ohm, whereas the capacitor is connected between the nodes 2 and 0 and has a value of 100uF. **V1** is the AC voltage source which provides a sine wave of DC offset 0 V, a peak amplitude of 1 V and a frequency of 2kHz. The .TRAN statement is a SPICE command, which requests a transient analysis over a specified duration with a specified time-step. In this example, the duration is $500\mu s$ and the time-step is $5\mu s$. This is explained in detail in the following section. **.print** is another SPICE command used to print the voltages at any particular node. In this example, the voltages at nodes 1 and 2 are printed. **.end** marks the end of the netlist.

2.4.1 DC, AC and Transient Analyses in SPICE

Any circuit analysis of a SPICE simulator is fundamentally based on constructing and solving the circuit equations using some complex numerical techniques [11]. The most simple analysis in SPICE is the DC analysis or the operating-point analysis. The DC analysis is also an integral part in various other types of analyses. The energy storage elements like capacitors and inductors are not considered in a DC analysis. In case they exist in the circuit description netlist, they are opened or shorted, respectively. Following this, a set of equations are formulated. Any circuit obeys 3 set of equations : Kirchoff's Current Law (KCL), Kirchoff's Voltage Law (KVL), Branch Constitutive Relation (BCR). KCL states that the sum of currents at any circuit node is zero at any instant of time. KVL states that the voltage across any branch in a circuit is equivalent to the difference

between the node voltages. BCR is the equation for any electronic device which governs its equation, e.g, Ohm's law for a resistor.

Depending on the method of equation formulation in SPICE, a set of non-linear algebraic equations are formed and solved. For example, in Sparse Tableau Analysis (STA), all the formulated equations are solved simultaneously while the sparsity of the equations are exploited to the maximum. In Modified Nodal Analysis (MNA), the KVL and BCR equations are substituted in the KCL equations to form n equations in n node voltages, hence a compact set of equations are obtained. The system of equations are first linearized using the Newton's method, and then LU factorization is employed to solve the system of linear equations. The Jacobian of the system matrix for applying the Newton's method is computed and LU is factored at each iteration until the convergence is reached.

A transient analysis in a SPICE simulator determines a circuit's behaviour in the time domain. The first step of a transient analysis is the DC operating point analysis which is carried out to determine the initial conditions. Energy storage elements like capacitors and inductors present in the circuit will provide ordinary differential equations to the system of circuit equations. Thus, for a transient analysis, the SPICE simulator is required to solve a set of non-linear differential algebraic equations (DAEs). The conversion of DAEs to a set of non-linear algebraic equations is the first step in a transient analysis. This is done by integrating the DAEs in time domain. The advancement of time for integration is done by a small interval called the time step and the voltages across the inductor and the currents through capacitors are integrated using one of the several numerical integration algorithms. The most popular integration algorithms available in SPICE are Trapezoidal Rule, Backward Euler and the Gear's variable order integration. This results in a set of non-linear algebraic equations which are solved as in a DC analysis. Afterwards, the simulation proceeds by computing the next time step until the required simulation time has been reached.

2.4.2 Limitations of SPICE

With accuracy being given the highest priority, SPICE simulators have a number of restrictions which include the following:

- No device model approximation
- Global convergence to the solution at every time point
- Consistent time discretization for all circuit elements

These restrictions "imply" that at every time point, a system of equations representing the entire circuit is constructed. Accurate transient simulation requires very small time steps, even for large latent circuit parts. In other words, a SPICE simulator must simulate a large number of time steps and in each time step, a huge system of linear equations must be accurately solved. Hence, the speed, accuracy and reliability of the SPICE simulators is constantly challenged and the main contribution of this comes from the cost of evaluating even more complex equation based semiconductor device models (e.g. BSIM4 with hundreds of parameters) and the computational cost of linear solvers.

2.5 Timing Simulation in Design Flow of VLSI Circuit

The design of a VLSI Circuit ideally starts with a specification at very high abstraction level and each design step results in a circuit description at lower abstraction level, which is shown below:

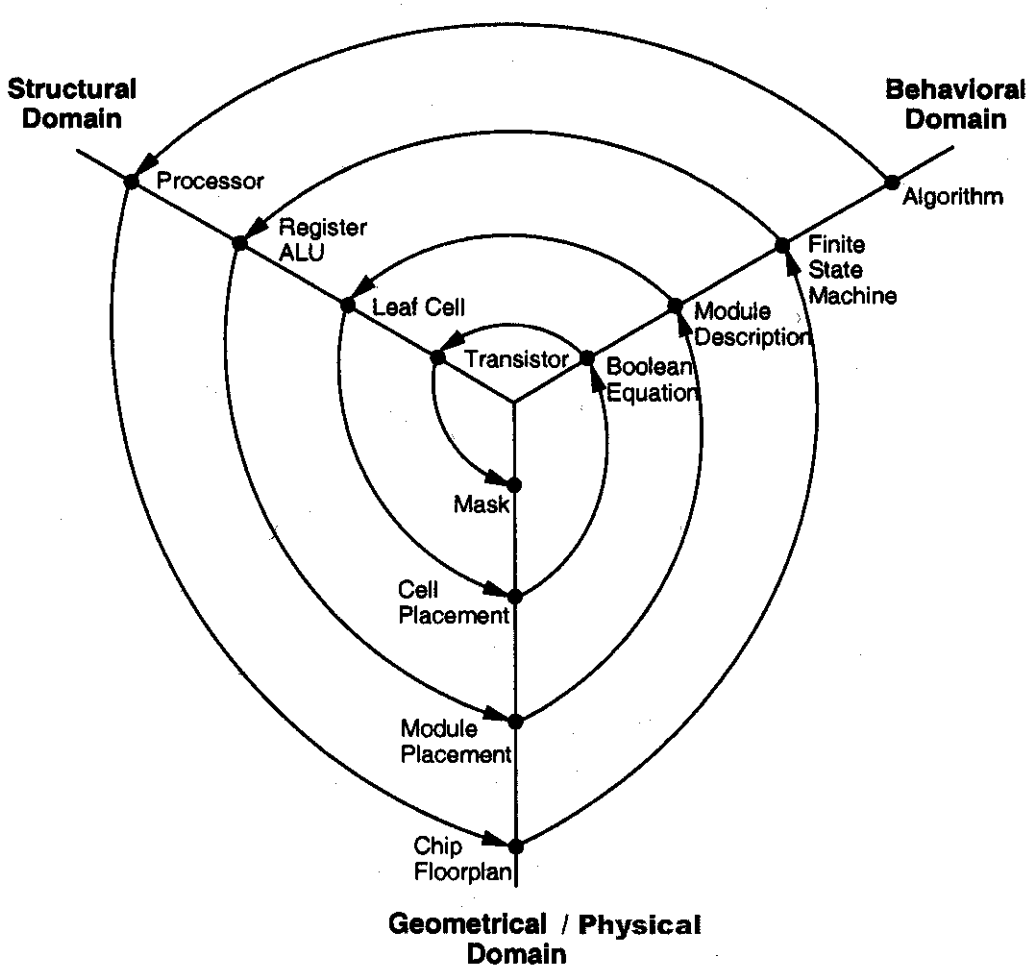

Figure 2.11: Y chart representation of IC design flow [12]

The above chart described the design flow in 3 major domains:

- Behavioural domain
- Structural domain
- Geometrical/ Physical domain

The design flow starts from the highest level and then proceeds to the lower levels (shown by arrows in the figure) [12].

The more simplified design flow of an IC basically consists of 3 main phases :

- Logic Design

- Physical Design
- Testing

The basic design flow of an IC along with the simulation steps is as shown below :

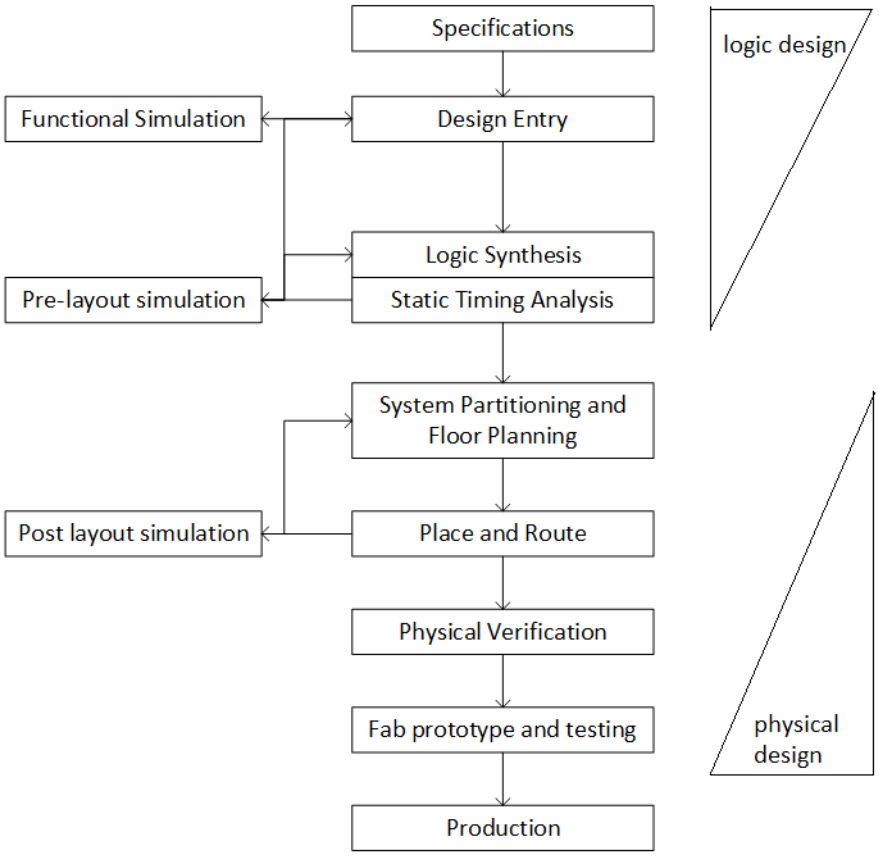

Figure 2.12: Design Flow of ICs and simulation steps involved (adapted from [13])

In the above figure, the logic design phase usually consists of the functional simulation and the pre-layout simulation where as the physical design consists of the post-layout simulation. Finally, the testing phase aims to detect any physical faults present in the design, by making use of automatic test pattern generation (ATPG) tool to generate test vectors for verifying the design.

In general, the major steps involved in a pre and post layout timing simulation [14] of a circuit is as shown in the figure below:

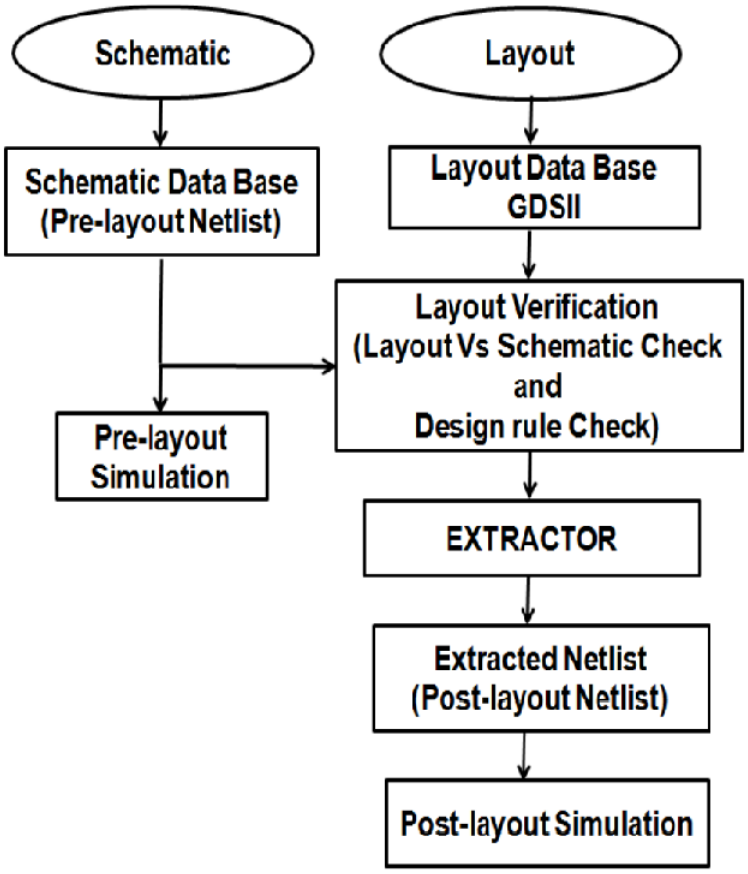

Figure 2.13: Pre and post layout Simulation Steps [14]

Once the design is completed, the schematic and the layout of the design is obtained. Initially, the design is validated by conducting tests like design rule check and layout vs. schematic test. These tests mainly test for the functionality of the circuit and it does not give the actual timing behaviour of the circuit. They do not consider the effect of parasitics as well.

Next step is the parasitic extraction of the design, which is used for the actual analysis of the circuit. The parasitic information of the circuit is used to represent the accurate electrical circuit model such that simulation of the circuit gives very accurate results considering the exact amount of extra delay caused by the parasitic resistances and capacitances. These include, the violations like set-up and hold times, glitches, etc.

In the following subsections, the details about pre- and post- layout simulations shall be explained.

2.5.1 Functional simulation and pre-layout simulation

The logic design phase with functional and prelayout simulation processes involved is as shown in the figure below:

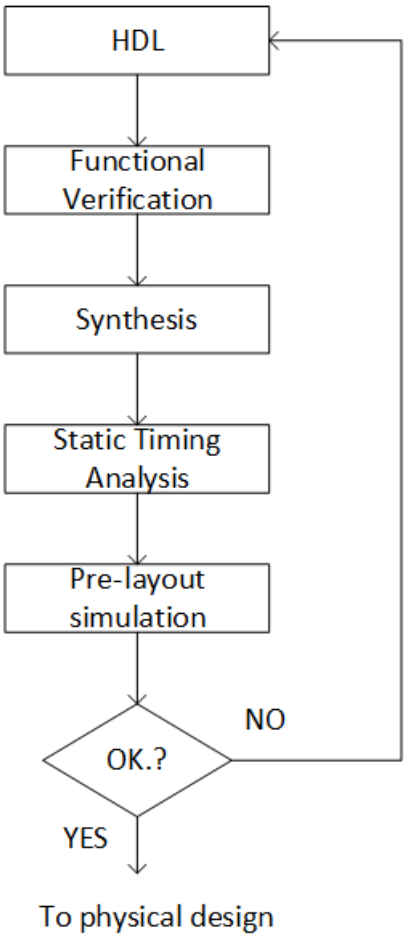

Figure 2.14: Steps of Logic phase of Design [13]

The first step of the design process after the design is finalized is the functional verification of the design. This only checks the functionality of the design. It does not provide any information about the timing of a circuit. As a next step, the prelayout netlist of the circuit is extracted to perform the pre-layout simulation. The Pre-layout simulations allow the circuit to be verified at a particular frequency, including the gate delays of the circuit. It allows the designers to identify and eliminate signal integrity, crosstalk and other electromagnetic compatibility (EMC) issues. This method is the most cost-effective way to design a circuit in relatively few iterations, as it occurs much early in the design process.

2.5.2 Postlayout Simulation

As the technology is scaling to the leading edge nodes of 28nm and below, ever more complex transistor models with hundreds of parameters are required for the accurate and reliable timing simulation of new circuit designs. These parasitic elements are now found to dominate the timing behaviour of the circuit. Furthermore, the impact of the

interconnect parasitic elements on the circuit timing has increased dramatically. So it is required to accurately simulate circuits with the interconnect parasitic resistance and capacitances, but in an efficient manner.

The steps of a typical physical design of IC involving postlayout simulation process are as shown in figure below:

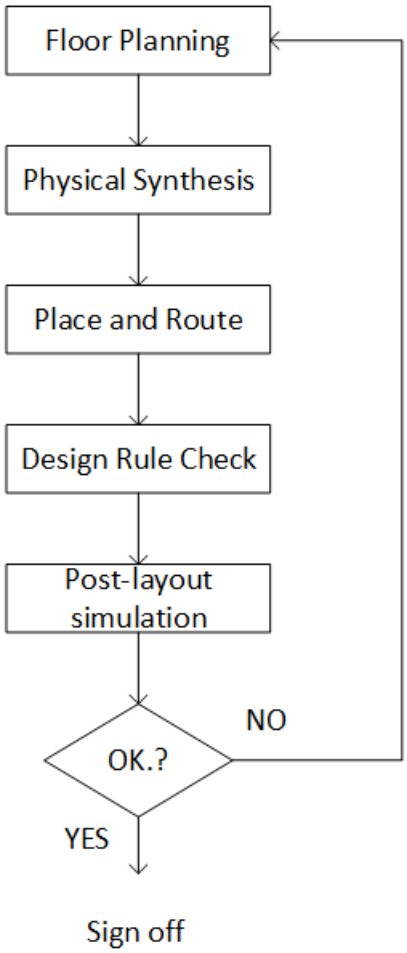

Figure 2.15: Steps of physical phase of Design [13]

The placement and routing process defines the exact location of the cells in a circuit. This also includes connecting cells with the interconnect material. This is followed by design rule check (DRC) which checks for rules against place and route so as to avoid any interference, cross-talk, etc [13]. Finally, post layout simulations are performed to test that the design meets the given timing requirements.

The more parasitic capacitances and resistances present in a circuit, the larger is the run time and memory requirements of the simulation [14]. Moreover, the parasitic extraction from the entire layout of the chip with all the connecting nets can result in an exorbitantly large netlist size. This may also affect the efficiency of the simulation, while possibly

contributing very little to the overall accuracy of the simulation results.

Also, as the resistance effect of the interconnects can no longer be ignored, the designers have moved from "C" to "RC" model for post-layout simulation.

Another considerable challenge is the presence of process variation effects. This introduces the need to simulate circuits for various process corners and this in turn increases the simulation run time and increases memory requirements.

2.5.3 Definitions related to Circuit Timing

2.5.3.1 Combinational Circuits and sequential circuits

A circuit is called a **combinational circuit** if it has combinational devices like logic gates (AND, OR, etc), MUXs and has no memory elements like Flip-flops, Registers etc [15]. The figure below shows an example of a combinational circuit.

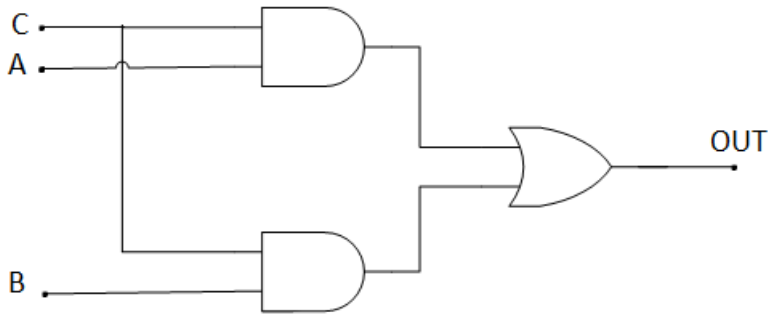

Figure 2.16: Combinational Circuit

If a circuit contains memory elements also in addition to logic gates, then such a circuit is called a **Sequential Circuit**. Example of such a circuit is as shown below:

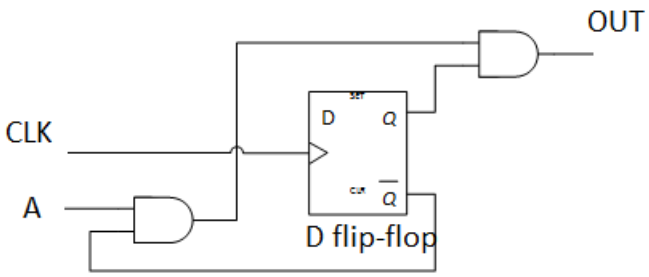

Figure 2.17: Sequential Circuit with a Flip-Flop

It is important to differentiate between the combinational and sequential circuits for timing analysis because combinational circuit timing analysis primarily deals with propagation delay of the logic gates and interconnect in the circuit only. For sequential circuit timing analysis, there are various other timing characteristics that should be satisfied which includes the setup time, hold time, and minimum clock period.

In this thesis, only combinational circuits are considered for timing analysis.

2.5.3.2 Delay Definitions

Any logic gate has some delay associated with it to transfer the voltage change at an input to the output of the gate. An ideal gate does not have any such delay, but in reality, all gates have some delay between a significant change of the gate input voltage and a significant change of the gate output voltage. This delay associated with a gate is called the **propagation delay** (or simply **gate delay**) of the gate. Consider an inverter having the following input and output waveform (with supply voltage $V_{dd} = 1V$) as shown in [16] the figure below :

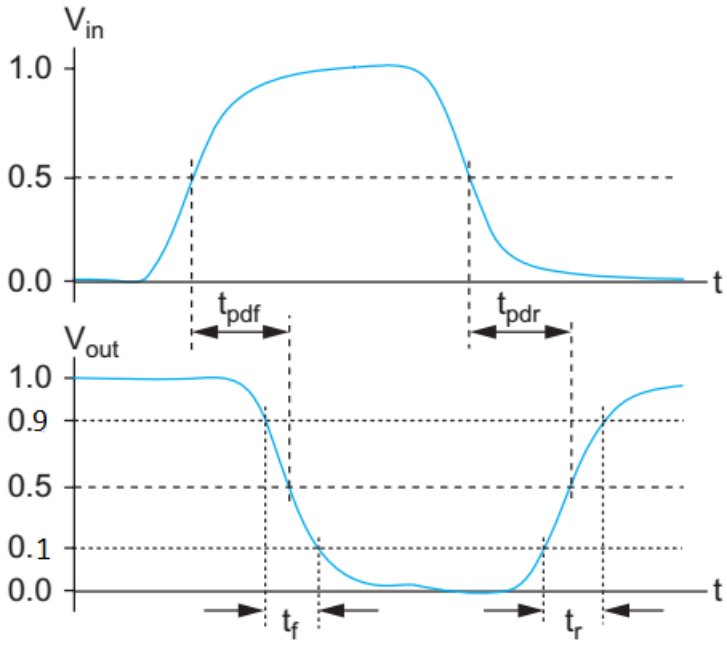

Figure 2.18: Input and output waveform of an Inverter [16]

Some common delay definitions with respect to the above are described below [17]:

t_{pdr} : Rise Propagation Delay

The Rise propagate delay is defined as the delay between 50% point of the input to the 50% point of the rising output of the gate.

t_{pdf} : Fall Propagation Delay

The Fall propagate delay is defined as the delay between 50% point of the input to the 50% point of the falling output of the gate.

$\mathbf{t_{pd}}$: Average Propagation Delay

The average propagation delay is defined as the average of the t_{pdr} and t_{pdf}. i.e,

$$t_{pd} = \frac{(t_{pdr} + t_{pdf})}{2} \tag{2.3}$$

The *transition time* is the time between two neighboring $0.1V_{dd}$ and $0.9V_{dd}$ voltage crossing points. Depending on the sign of the voltage change, it is also called rise time or fall time.

$\mathbf{t_r}$: Rise Time

The rise time is usually defined as the time taken for a rising transition to increase from the lower threshold to the upper threshold, usually $0.1V_{dd}$ to $0.9V_{dd}$ respectively, where, V_{dd} is the supply voltage.

$\mathbf{t_f}$: Fall Time

The fall time is usually defined as the time taken for a falling transition to decrease from an upper threshold to the lower threshold, usually $0.9V_{dd}$ to $0.1V_{dd}$, where V_{dd} is the supply voltage.

Chapter 3

State of the Art in Circuit Timing Simulation

3.1 Timing Model - Gate and interconnect model

Timing models typically consist of the driver model, interconnect model and the receiver model (Fig 3.1). The driver and receiver model, together referred to as the gate model, require characterization using circuit simulators, while the interconnect model can be extracted from a layout. A timing model can be represented as follows:

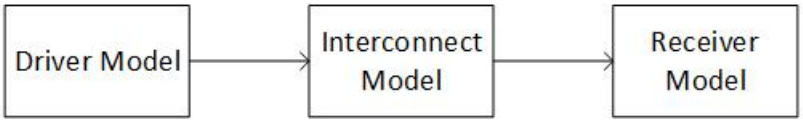

Figure 3.1: Block Diagram of a timing model

The accuracy and runtime of the timing simulation is determined by the accuracy and the computational cost required to evaluate the circuit model. The circuit model is a netlist of gate and interconnect models. To find a suitable trade-off between runtime and accuracy, gate and interconnect models at different abstraction levels have been proposed.

On a broad level, simulators are generally classified into the following categories:

- Gate level simulator
- Switch level simulator
- Transistor-level
- Circuit level simulator

The figure below depicts the accuracy-runtime variation of the above mentioned type of simulators as compared to SPICE simulators (which is a circuit level simulator) which can be treated as "Exact" Simulators.

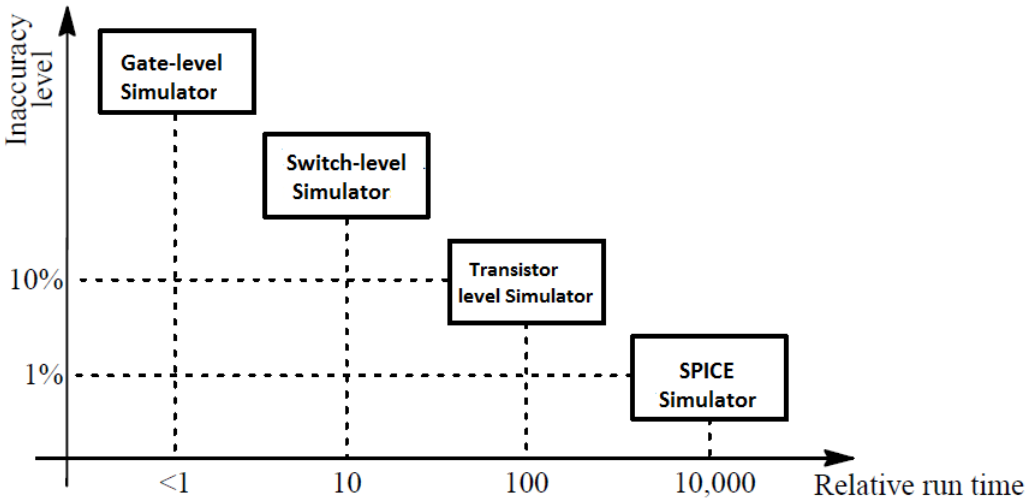

Figure 3.2: Relative accuracy and runtime of the timing simulation with gate and interconnect models of different abstraction levels, adapted from [11]

3.2 Different abstraction of simulators

In general, switch-level and gate-level simulators in the above figure sacrifice accuracy in order to achieve huge speed-up. These are mainly used for the functional simulation of the circuits, completely ignoring the timing information. "Exact" Simulators refers to those where the circuits are simulated with SPICE like tools which use complex device models (e.g BSIM4), which demands extremely large computation requirements. Hence, this type of simulation becomes practically impossible for very large circuits. Transistor-level simulator is the category of simulators which are used to determine the timing results of a circuit with a reduced accuracy than "Exact" simulators, but at a much faster speed.

3.2.1 Gate-Level Simulation

In gate-level simulators [18], the circuit to be simulated is modelled as a number of logic gates connected together. Each logic gate has a delay model that represents the delay of the gate, which is the time a transition at the input of the gate needs to propagate to the output of the gate. The simulation algorithm consists of a simple event driven mechanism. Gates whose inputs are exclusively connected to primary inputs are evaluated first and their outputs are updated at the appropriate time in future. Following these, once the outputs of the gates in the first logic level are updated, the fanout gates are simulated and their corresponding outputs are updated. Thus events are repeatedly scheduled and evaluated until the circuit is simulated for the required number of cycles. This kind of logic simulation forms the most integral part of any IC designing process.

3.2.1.1 Non Linear Delay Models (NLDMs)

Most of the cell libraries comprise of table models which specify the delay and timing checks for various timing arcs of the cell. These table models are usually referred to as Non Linear Delay Models (NLDMs). The NLDMs are the traditionally used delay models. These models are look-up table (LUT) based models which store the gate delay and the output transition time in 2 look-up tables for various input transition times (t_{in}) and output load capacitance C_{out} as the entries. The delay information is usually obtained from the Liberty (.lib) file, also called as the timing library file. However, if customised cell library is required for some particular parameter values, then the cells can be characterised by using tools like HSpice or similar tools.

The gate delay and output transition time are characterised using a circuit simulator with appropriate input stimulus to cause the output transition. These look-up tables are later used with linear interpolation technique to obtain the gate delay and output transition time while simulating the gates in a circuit.

The accuracy of the timing analysis depends on the accuracy of the cell characterisation. The delay and the output transition time LUTs constructed for NLDMs usually consist of fixed number of entries , e.g, `5 X 5` , `7 X 7` tables of (t_{in}, C_{out}) pairs [19]. The value of timing information for other (t_{in},C_{out}) pair is obtained by interpolation of the nearest neighbours.

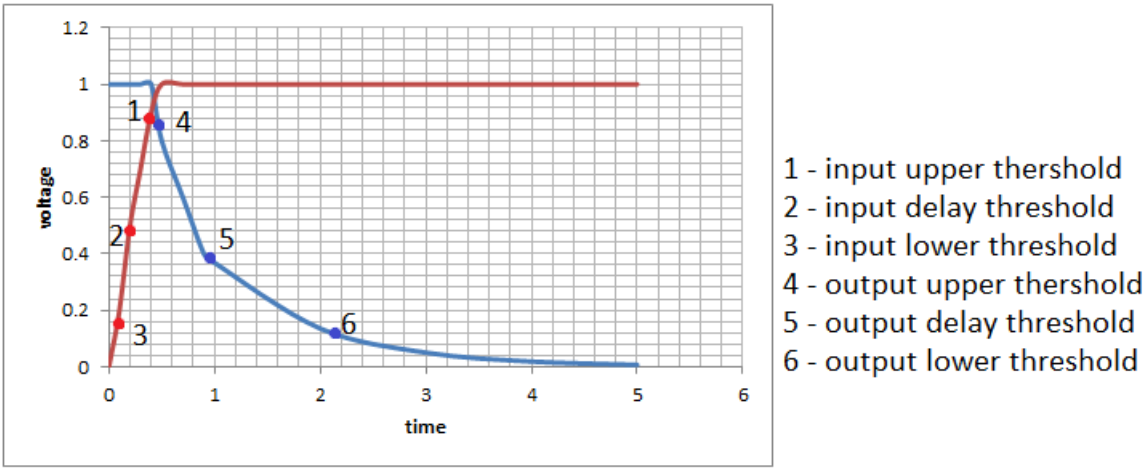

Figure 3.3: NLDM measurement points, adapted from [20]

The above picture shows the input/output measurement points. It is seen that in NLDM, only 3 cross over points are determined, 0%, 50% and the 100% points (marked as lower, delay and upper threshold in the input and the output waveforms). The gate delay is modelled as the difference between the 50% points of the input and the output waveform and the output transition time is the difference between the 0% and the 100% points of the output waveform [20].

3.2.1.2 Disadvantage of NLDMs

Once the characterisation data is available, NLDMs offer extremely fast timing analysis. However, these timing models have a disadvantage that they are insufficient to reflect the non-linearities of the circuit at lower geometry. Another shortcoming of the NLDMs is that it fails to capture the effect of miller capacitance effects between the input and output nodes. This effect dominates the delay calculation for small impedance nets. Also, NLDMs fail to capture the effect of multiple input transitions at the input of a gate. Hence, the timing analysis with this model becomes very crude in cases of multiple input switching (MIS).

3.2.1.3 CCS and ECSM Timing Model

To account for the inaccurate results provided by NLDMs, several other models like the ECSM and the CCS were developed. Effective Current Source Model (ECSM) is essentially based on current source modelling. It stores the various voltage sample values as a function of time.

The composite current source (CCS) provides all the necessary tools and guidelines for the accurate library characterization and validation. The CCS is the first in the EDA industry to deliver a complete open-source current based modeling solution for timing, noise and power [21]. Along with the various available parsers, characterization tools, validation tools and guidelines, these provide an open source Liberty modelling format which enables efficient characterization for cell library creators.

Some facts about the CCS include [22]:

1. Uses a current source for modelling the driver
2. Stores the gate output current as a function of time. It stores data specifically targeted at characterizing a cell's timing, noise, and power behaviour.
3. CCS consists of more data points as opposed to NLDM which captures only three cross-over points. Hence, CCS requires more memory than NLDMs.
4. The receiver model for the CCS consists of a 2-segment capacitor in order to model the Miller Effect
5. CCS provides much better accuracy than NLDMs
6. CCS enables both temperature and voltage scaling of detailed cell behaviour.
7. CCS addresses the existing and emerging design requirements including the physical effects of nanometer designs as well as the needs of design strategies such as multiple-voltage domains

The timing model of a cell with CCS or ECSM consists of a driver model and a receiver model along with the interconnect Model. The driver model consists of a time and voltage dependent current source in addition to the cell parasitics, and a cell output current is characterised for a given output load and input waveform The advantages of such a model are :

- Current source models are known to be very efficient in modelling the non-linear switching activity of transistors and hence prove to be very accurate.
- It is known to be better than traditional NLDM gate models in handling the complex interconnects which is prevalent in the currently existing low-power nanometer designs.
- This model provides high accuracy even when the drive resistance is much lower than the interconnect impedance as it has much more detailed information on drive resistance.

One disadvantage of these CCS and ECSM is that the characterisation is dependent on the input signal and the output load capacitance. Also, another drawback is the computation overhead caused by these models. Hence, substantial research is being made to develop these models with high accuracy and to keep the computation overhead minimal.

3.2.2 Switch-level Simulation

Switch level simulation is applied exclusively to digital circuits. Here each transistor is modelled as a voltage controlled resistive switch which is either on or off [23]. Each transistor is assigned a fixed conductance or resistance to describe the current driving capability of the transistor. This type of simulation is repeated for different values of the signals at the primary input. This type of simulation is efficient because its MOSFET model is very simple and very large circuits can be simulated. However, there are a few fundamental limitations linked to this simple modelling of MOSFET. Firstly, it gives very little timing information and secondly certain analog situations like charge-sharing, glitches, etc are worst handled.

An example of a resistive switch representation of a NOR gate followed by an Inverter gate is as shown in figure below:

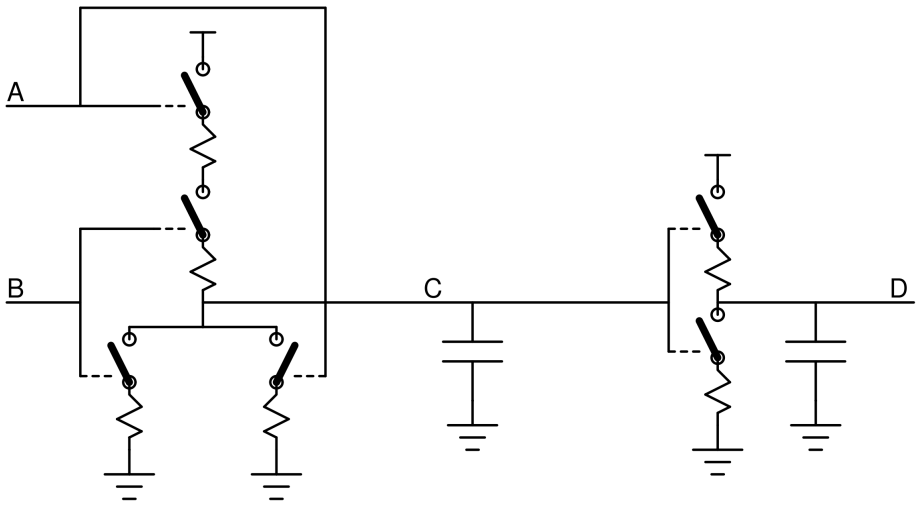

Figure 3.4: Resistive Switch Representation [24]

The very first switch level timing simulator was the MOTIS [25]. This simulator used Look-up tables to store the I-V characteristics of transistors. The charging current of the load capacitance was determined from these tables to evaluate the change of voltage at the output of the component connected block (CCB).

Following these, there were numerous simulators that were developed which approximated the device and circuit quantities by piecewise approximate functions. These included the E-LOGIC simulators [26] which used the method of nodal analysis, but considered the voltage to be discretized into different "levels". The various times at which the discrete voltage levels are reached at output node is found and simulation is done in an event-driven manner. To speed up the simulation process more, the MOS transistor models were further simplified and were considered as current-limited switches for very fast event-driven simulation.

3.2.3 Transistor level Simulation

The more complex and time-consuming type of simulators are the transistor-level simulators which model the transistors in a logic gate by describing their non-linear current and voltage equations. Thus, these simulators have much more complex gate models that provide more accurate results but these models are much simpler when compared to gate models of SPICE tools, and hence they provide faster than SPICE results. This is seen in the fig 3.2 shown above.

Some simulators based on transistor-level models include ACES and SAMSON. In ACES [27], the I-V characteristics of the transistors were stored as PWL functions, and the voltage waveform was obtained by performing explicit integration with adaptive time steps to achieve an accurate and efficient simulation. SAMSON [28] was essentially a mixed logic simulator, that used event-driven algorithms to exploit latency. This simulator provided much more accurate results than the previous ones.

3.2.3.1 Current Source Models(CSM)

In an attempt to solve the accuracy of gate model in presence of process variations, noise and voltage fluctuations, Current Source Models were developed [29].

The traditionally available NLDMs are used to abstract the delay and output transition times of cells which are later used for the timing analysis. However at geometries 90nm and below, many new effects cannot be modelled properly. Some of these challenges include :

- Miller Effect
- Multiple Input Switching (MIS)
- High Impedance interconnect
- Dynamic IR drop
- Temperature Inversion

- Process Variations

The very first CSM which was called as Blade [30], was developed using a voltage controlled current source, a capacitance and the time shift of the output voltage waveform. Since CSM is an active research topic, various models have been developed over the past years [31–34]. In [34], the logic gate was modelled to consist of a nonlinear current source, an input and output voltage dependent capacitors and a miller capacitor connected between the input and the output nodes. This model could not handle situations where a fast ramp signal is applied at the gate input a small capacitive load is present at the gate output. Thus, this work was extended to include a calibration capacitance in [35]. This was later adapted in paper to provide high accuracy even in case of multi-stage circuits [36].

3.2.4 Circuit level Simulation

Conventionally used simulators for postlayout simulation included traditional true spice simulators, that gave very accurate results. But as the circuit size grew and the parasitic effects of technology increased, these simulators demanded unimaginable runtime and memory. The memory and the CPU time requirements demanded by these exact simulators for simulating even moderately-sized circuits (having say 50,000 transistors) turn out to be unacceptable. This gave rise to a much accelerated class of fast spice simulators which provides an optimised run-time.

In general, these simulators sacrifice the accuracy of timing simulation (ideally in the range of 10% relative to exact simulators) to reduce the computational cost.

The properties of these fast-spice simulators include:

1. Use of approximate MOSFET models
2. Reduction of parasitic resistance and capacitance network to speed up the simulation process
3. Applies partition algorithms and uses partition matrix to perform hierarchical simulation
4. They seek to simplify the repeated evaluation of the non-linear analytic device models. Thus, sacrificing the accuracy of simulation for speed.
5. Employ the use of "event-driven" algorithms which incur computation only when there is activity in the circuit and only in those portions of the circuit where there is any activity and hence exploits the latency of the circuit.
6. Speedup over the SPICE simulators by using more simplified linearization or integration techniques.
7. Simulators are built using gate models that partition the circuit into "channel connected components" which are essentially sub-circuits consisting of transistors which are drain-source channel connected. The boundary of these channel connected components are either the gates of transistors, primary input of the circuits, primary outputs of circuits, power supply or ground.

These fast-spice simulators provide IC designers additional timing estimation methods which prove to be an integral part of the design methodology. They form the basis for timing and power estimation methods in the design of ICs and they have become an integral part of any design process of ASICs, memory, etc.

3.3 Interconnect Modelling

In the past history of Integrated Circuits, the interconnect wires were considered only in certain special cases or when some high-precision analysis is performed. But, as the device dimensions are constantly reducing and the speed of the circuits are aimed to be increased, the parasitic effects of the wire interconnects can no more be neglected [37]. This is further aggravated by the fact that the improving technology has enabled the manufacturing of larger die sizes economically feasible, which in turn means an increased average length of the interconnect wires and hence the increased parasitic effects of these wires. Thus, it is becoming an essential need to consider and simulate the wire interconnects as well in addition to simulating the various gates in the circuits. In other words, with continuous scaling, the delay contributed by the interconnects or the wires is becoming a significant contributor in the total delay of the circuit.

An ideal wire is the one that has no parasitic elements like parasitic resistances or parasitic capacitances associated with it and hence it has no effect on the electrical properties of the circuit [37]. Such ideal wires are considered to be equi-potential in the sense that a voltage applied to one end of the wire propagates immediately to the other end even if separated by a distance. This is the most simplistic model of a wire and it is usually considered in the early stages of design when it is more important to analyse the behaviour of the transistors. However, with constantly shrinking technology, and for the accurate simulation of circuits, it is important to consider more complex model for the wires so as to analyse their effect on the electrical properties of the circuit.

To study the effects of interconnects on the simulation of a circuit, it is important to accurately approximate the actual behaviour of the wire as a function of its parameters. These are called as interconnect models which may vary from being very simple to very complex depending on the accuracy requirements.

The basic idea of interconnect modelling is to produce a reduced order model of the interconnect networks which captures the original interconnect behaviour upto a particular frequency. Reduced order models of large RC networks can be plugged into any timing simulator and simulated more efficiently with the non-linear drivers and receivers.

Some examples of the interconnect models are:

- Lumped C model
- Lumped RC model
- Distributed RC model [37]

3.3.1 Lumped C model

The simulation of interconnects for the simple gate model is done by reducing the RC interconnect network into a lumped C model. This means that the entire RC network of the interconnect wire is represented by using a single capacitance as shown in figure below:

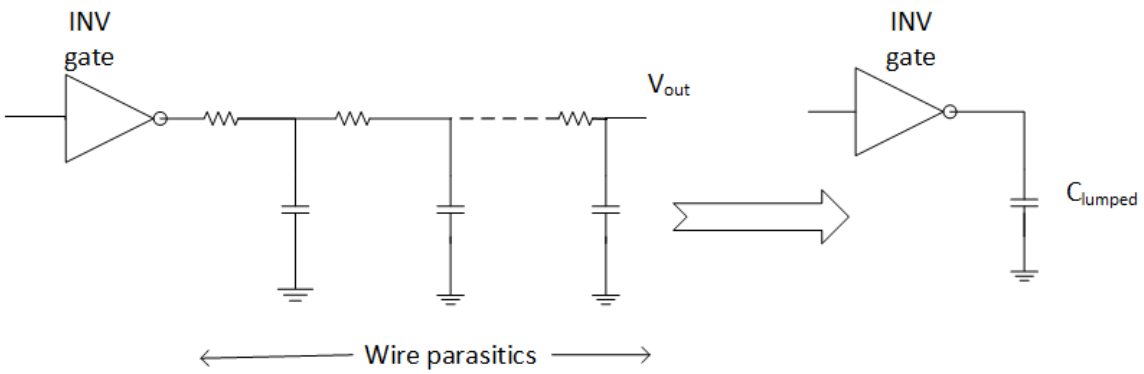

Figure 3.5: RC network reduction using Lumped C model, adapted from [37]

The lumped C model considers only the dominating capacitive parasitics of a wire and assumes the resistive components to be negligibly small. The distributive capacitances of a wire are lumped together into a single capacitance as shown in the figure above. This model still represents the wire as a equipotential region and it only changes the capacitive loading effect on the driving gate. This wire model is quite simple and yet effective and it is the most commonly used wire model in the analysis of ICs.

3.3.2 Lumped RC Model

For very small on-chip wires of the sizes of few mm length, the resistance effect is quite significant. The equipotential assumption assumed in the simple lumped C model is not adequate anymore and hence a RC model is adopted.

The simplest way to model the interconnect resistances and capacitances using the lumped RC model is to lump all resistors of the net into 1 resistor and to lump all capacitances to 1 capacitance. Thus, the model can be represented as follows:

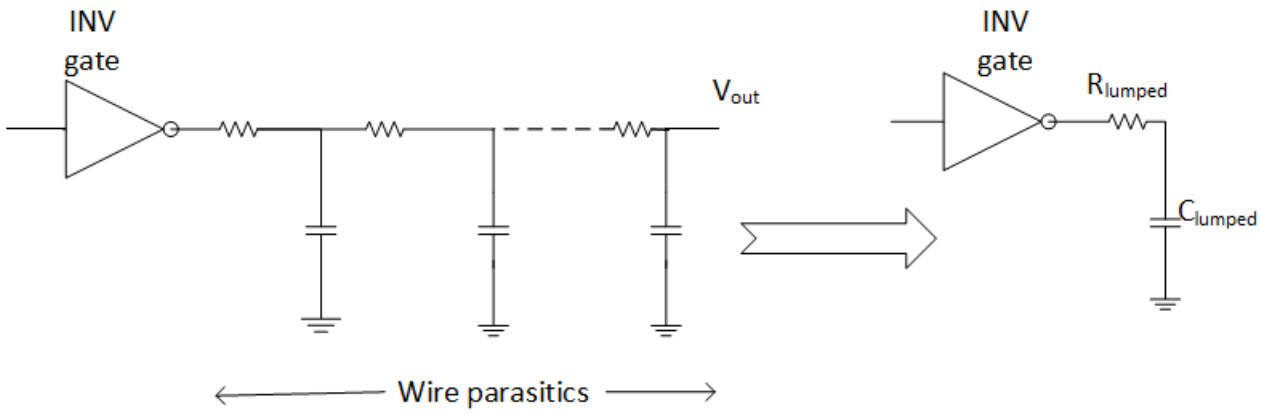

Figure 3.6: RC network reduction using Lumped RC model, adapted from [37]

The R_{lumped} and C_{lumped} represent the overall lumped resistance and capacitance of the interconnect.

Chapter 4

Design of a Heterogeneous Gate Model

This chapter presents the proposed heterogeneous gate model for fast and accurate timing simulation. An abstract description of the model is presented in section 4.1. The following section 4.2 shows the waveform model, which defines the inputs and the output of this model during simulation. The section 4.3 defines the simple gate model and the sections 4.4 and 4.5 describe the details of the advanced gate model of the proposed heterogeneous gate model.

In this thesis, we shall consider only the 2-input NOR gate and the inverter gate for all the following sections that follow. This ensures that all circuits used for the validation of this model are synthesized using these 2 gates only.

4.1 Model Description

The proposed heterogeneous gate model combines a very fast gate-level model designed for single input switching with a complex but very accurate transistor-level model.

4.1.1 Overview

The heterogeneous gate model proposed in this thesis uses a simple gate model for single input gates (Inverter Gate) and in case of a multiple input gate, it uses a classifier to determine whether a multiple input switching (MIS) case is detected at the gate input. In case of MIS scenario being detected, the advanced gate model based on current source model (CSM) is simulated to compute the next gate output transition as this model considers the effect of MIS, which the simple gate model fails to capture. Thus, the heterogeneous gate model consists of a simple gate model based on NLDM, an advanced gate model based on CSM and a classifier to choose between the 2 when a MIS case occurs. Note that this gate model takes the voltage waveform at the gate inputs and the gate output as the input and produces the gate output current waveform as result.

The idea of the proposed heterogeneous gate model can be represented as shown in the figure below:

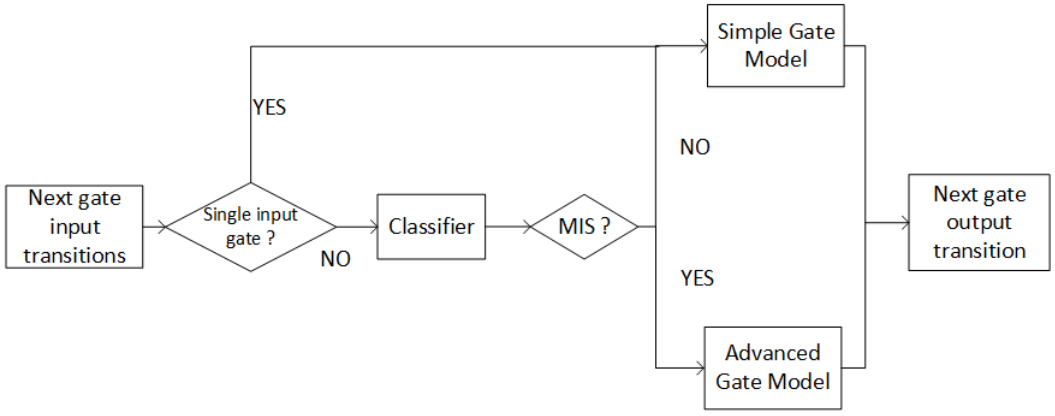

Figure 4.1: Proposed heterogeneous gate model for efficient timing simulation

The purpose of combining gate models of different abstraction levels is to explore the trade off between the speed of simulation using the simple gate model and the accuracy achieved using the advanced gate model. The classifier thus plays an important role in achieving this trade-off between the speed and accuracy of simulation.

The primary idea of using the classifier for a multi-input gate lies in the fact that although NLDM provides very fast computation results, these results are inaccurate, especially in cases where multiple input switching occurs. The timing libraries used by the NLDMs have the pin-to-pin delays of each gate characterized for single input switching case only. The impact of simultaneous switching of the gate inputs can cause significant errors in the timing estimates of the gate [38]. Since this effect of multiple input switching is not captured in NLDMs, another advanced gate simulation algorithm is used in such cases compute the next gate output transition more accurately.

4.1.2 Impact of Multiple Input Switching on Gate Delay

Given that the simple gate model calculates the next gate output transition much faster when compared to the advanced gate model, it is most desirable to simulate all gates using the simple model except for those cases, where simultaneous switching of the gate inputs occurs. To avoid the application of the advanced gate model in cases where the simple gate model provides sufficient accuracy, it is necessary to design the classifier in such a way that it chooses the time consuming advanced gate model only when it is necessary.

For any basic logic gate, we can define the *Controlling* and the *Non-Controlling* values. The controlling value completely determine the value at the output of the gate. Hence, the input transition can either be *Controlling* to *Non-Controlling* (CTN) or *Non-Controlling* to *Controlling* (NTC) depending on its effect at the output of the gate. For example,

for a NOR gate, the *Controlling* value is 1 and the *Non-Controlling* value is 0. Thus, a falling transition is a CTN transition, while a rising transition is a NTC transition.

Consider the cmos 2-input NOR gate as shown below:

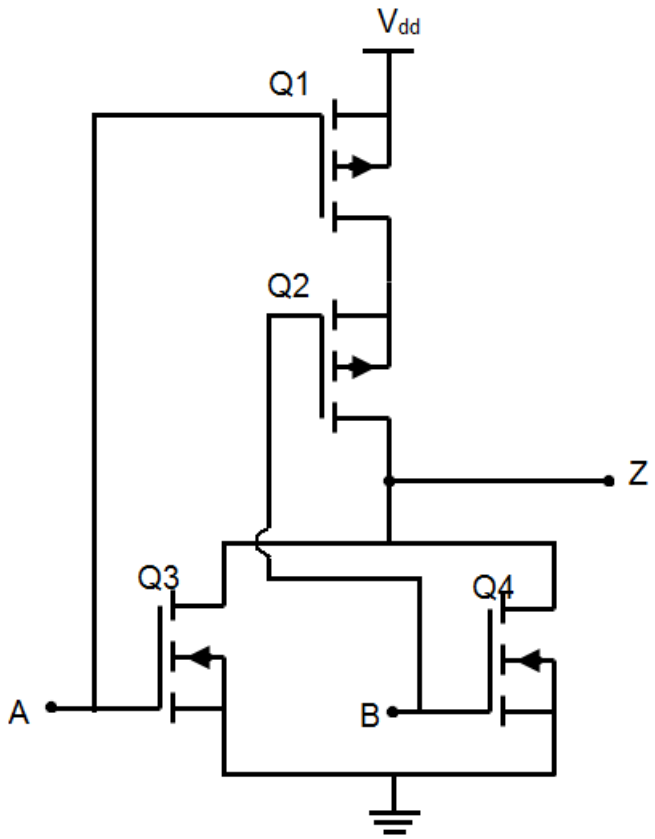

Figure 4.2: CMOS NOR gate

In the above diagram of a NOR gate, Q1 and Q2 are p-channel MOSFET transistors and Q3 and Q4 are p-channel MOSFET transistors. The inputs of the NOR gate are denoted by A and B and the output of the gate is denoted by Z.

When only one gate input (A or B) has a transition and the other input remains at the non-controlling value, the delay of the gate is called the single input switching (SIS) delay. This delay can be determined quite accurately using a simple gate model like NLDM.

When both A and B have NTC (rising) transitions, both the n-channel MOSFET (Q3 and Q4) are conducting in parallel. Therefore, the gate is discharged with twice the current. In this case, the output arrival time is determined by the earliest arriving signal. Similarly, when both inputs A and B have a CTN (falling) input transitions, then the current starts flowing only when both the p-channel MOSFET are conducting. Therefore, the latest arriving signal determines the arrival time at the gate output.

When multiple inputs of the gate switch, it is necessary to consider whether they switch in the same direction or not. If multiple inputs switch in opposite directions, they either

do not propagate a signal to the output or they produce a glitch at the output of the gate. Hence, it is more important to consider those multiple input transitions which happen in the same direction.

4.1.3 Definition of Classifier

When the gate to be simulated is a multi-input gate, initially the transitions at the input of the gate are analysed to determine whether they cause a transition at the output of the gate. If a transition occurs at the gate output due to a transition at one of the gate inputs while the other gate inputs remain stable during that time, then the gate output transition is computed according to the simple gate model.

However, if a gate output transition is caused due to the combined effect of multiple gate inputs switching almost simultaneously, the simple gate model cannot provide sufficient accuracy and the classifier has to choose the advanced gate model for the computation of this output transition. From the above section, it can be seen that the most important parameter that determines the MIS effect on the gate delay is the relative arrival time between the 2 input transitions. Hence the classifier must consider this parameter to determine MIS scenario.

To minimize the computational cost for the classification of every gate output transition, the design of the classifier is kept as simple as possible and hence the classifier evaluates the absolute difference in arrival times between the 2 inputs that switch simultaneously. If this absolute difference is less than a particular threshold, the output transition is computed according to the advanced gate model to simulate the gate. Otherwise the gate output transition is computed using the simple gate model. The design representation of the classifier is as follows:

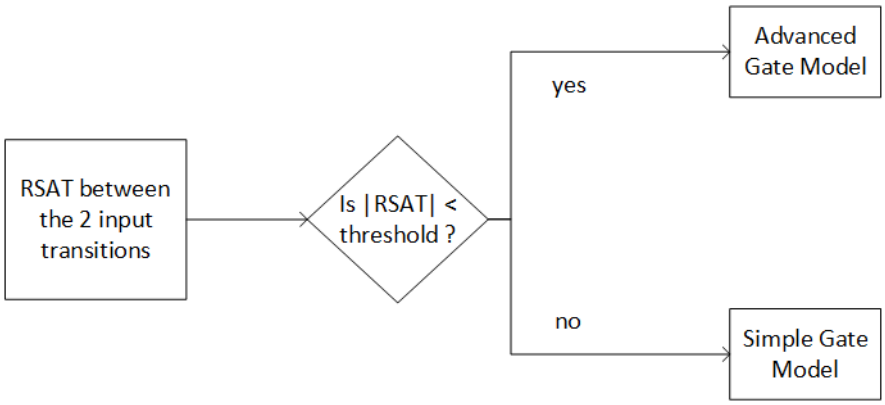

Figure 4.3: Block diagram of the efficient classifier

Suppose that the 2 inputs of a 2-input NOR gate are represented as A and B and let the output of the gate be represented as Z. Let V_{dd} denote the supply voltage of the gate. The transition time of a transition tr, where $tr \in (R, F)$, can be denoted as $\boldsymbol{T^A_{tr}}$ and represents the time taken for the input to rise from $0.1\mathrm{V_{dd}}$ to $0.9\mathrm{V_{dd}}$ in case of rising

transition (**R**) or the time taken for the input to fall from $0.9V_{dd}$ to $0.1V_{dd}$ in case of a falling transition (**F**). The arrival time of any transition on input A can be denoted as $\boldsymbol{A^A_{tr}}$ and it is defined as the time when the voltage at input A reaches $0.5V_{dd}$. Similarly, $\boldsymbol{A^B_{tr}}$ denotes the arrival time of a transition at input B. The *Relative Signal Arrival Time* (RSAT) between 2 transitions at inputs A and B is denoted by $\delta^{A,B}$ and is defined as the difference

$$\delta^{A,B} = A^A_{tr} - A^B_{tr} \tag{4.1}$$

between the transition arrival times at input A and input B.

To decide whether the simple or the advanced gate model should be used to compute the output transition, a suitable threshold for $|\delta^{A,B}|$ must be determined. This threshold is chosen based on the analysis in [38], which will be summarized in the following.

For a 2 input NOR gate, the delay of the gate when a single input has a rising transition is much larger than the case when both the inputs have rising transitions. This is because, in the latter case, multiple n-channel MOSFET transistors are turned ON for the output gate capacitance to discharge. The speed-up caused by simultaneous transition of the inputs, can be easily understood by plotting the gate delay as a function of the RSAT $\delta^{A,B}$, keeping the input transition times, $\mathbf{T^A}_R$ and $\mathbf{T^B}_R$, at some constant value. The plot is as shown below:

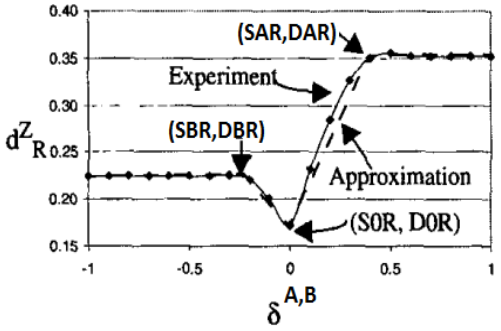

Figure 4.4: Fall propagation delay d^Z_R of 2-input NOR gate as a function of $\delta^{A,B}$ [38]

It is seen from the above graph that the speedup caused due to simultaneous switching is significant only when the difference between the arrival times of the 2 inputs (RSAT) is close to zero. Thus, an upper bound SAR and a lower bound SBR can be found such that if

$$SBR < \delta^{A,B} < SAR, \tag{4.2}$$

the MIS effect has significant impact on the delay of the gate [38]. SAR is the minimum $\delta^{A,B}$ for which the gate delay is unaffected by the transition at input B and is completed determined by the transition at input A. Similarly, SBR is also defined the same way. Hence, one of the important parameters to be considered for determining the effect of MIS on the delay of the gate is the Relative Signal Arrival Time $\delta^{A,B}$ between the transition arrival times at the gate inputs A and B.

Other factors that determine how significant MIS is on the gate output are the gate input transition times. Again from literature [38] , it is seen that for a fixed value of $\delta^{\mathrm{A,B}}$ and $\mathbf{T^B}_{tr}$, the gate delay is a function of $\mathbf{T^A}_{tr}$ which is either:

(i) monotonically increasing, or

(ii) monotonically increasing and then monotonically decreasing.

This is true for both rising and falling transitions and is illustrated as shown in the figure below, where $d_F^{Z,A}$ and $d_R^{Z,A}$ denotes the gate delay for a falling or a rising transition at input A.

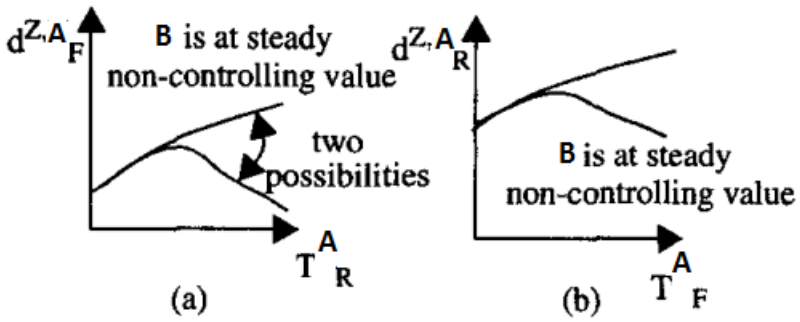

Figure 4.5: (a) Propagation delay vs. Rise time of input signal A (b) Propagation delay vs. Fall time of input signal A [38]

Similarly, when the output transition time is plotted as a function of $\mathbf{T^A}_{tr}$, for some constant $\mathbf{T^B}_{tr}$ and $\delta^{\mathrm{A,B}}$, it is strictly monotonically increasing. This is shown in the figure below, where $t_F^{Z,A}$ and $t_R^{Z,A}$ denotes the output transition time for a falling or a rising transition at input A:

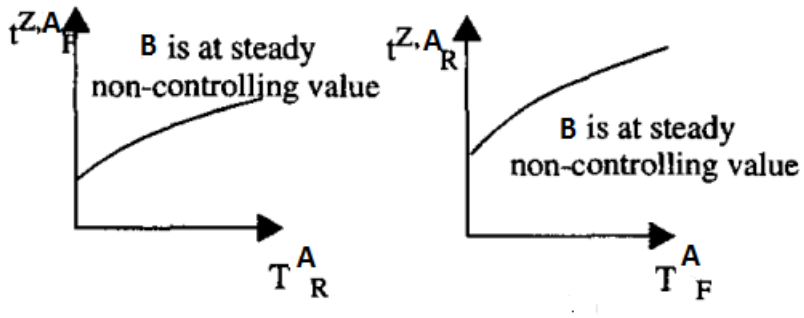

Figure 4.6: Output transition time vs. Transition time of input signal A [38]

However, the gate delay $d^{Z,A}{}_{tr}$ and the output transition time $t^{Z,A}{}_{tr}$ has the same variation with the relative arrival time between the 2 inputs.

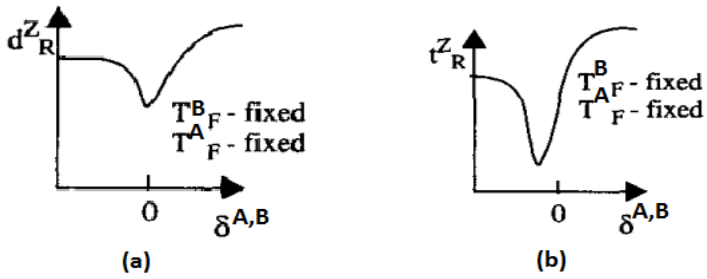

Figure 4.7: (a) gate delay vs. RSAT (b) output transition time vs. RSAT [38]

So, from the above, it can be concluded that output transition time increases with increasing input transition time, and, the gate delay is bitonic with respect to input transition. It is seen that the minimum gate delay is observed Relative Signal Arrival Time $\delta^{A,B}$ is close to zero. However, the minimum output transition time does not necessarily occur when $\delta^{A,B} = 0$

4.2 Input and Output Waveform Model

One of the most important parameters to consider while designing the heterogeneous gate model is the modelling of the input and the output waveforms. The inputs to the heterogeneous gate model are voltage waveforms and the output of the heterogeneous gate model is a current waveform. All waveforms are modelled as Piece-Wise Linear functions (PWL). A PWL function is used to construct a waveform by combining a series of straight line segments connecting a number of (x,y) pair of values which are defined by the user.

In the following, the piece-wise linear function model of a waveform will be called *piece-wise linear waveform*. For example, the PWL voltage waveform with 5 voltage levels of {0V, 0.22V, 0.44V, 0.66V, 0.88V, 1.1V} with an alternating time gap of 1us and 2us is as shown in the figure below:

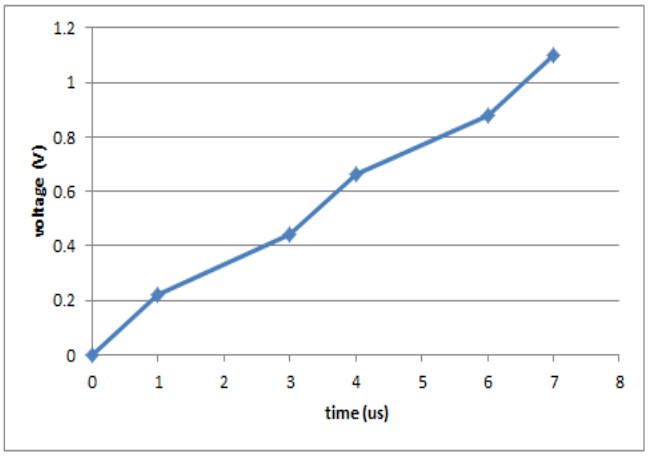

Figure 4.8: Example of a PWL Waveform

Here, the voltage is considered to be linear between any 2 consecutive voltage points. That is, between 0V and 0.22V, the voltage is considered to be linearly rising with time at the rate of 0.22V per 1us, while between the voltages 0.22V and 0.44V, the voltage is linearly rising at the rate of 0.11V per 1us. This way, the voltage waveform is linear in pieces although on the whole form 0V to 1.1V, it may not be considered linear. The waveform in the above example can be represented as :

$$\{(0, 0V), (1us, 0.22V), (3us, 0.44V), (4us, 0.66V), (6us, 0.88V), (7us, 1.1V)\} \quad (4.3)$$

The piece-wise linear voltage waveform is usually defined as a set of (time, voltage) pairs while for a current waveform, it is defined as a set of (time, current) pairs. The more pairs (x, y) are used to represent the waveform, the more accurate the shape of the waveform will be. This flexibility is particularly important given the non-linear nature of the gate output current modelled in the advanced simulation algorithm of the heterogeneous gate model. This in turn helps in finding the accurate voltage waveform at the output of the gate.

4.3 Simple Gate Model

The Non Linear Delay Model (NLDM) has proven to be very efficient if the gate output transition is caused by a single transition at one of the gate inputs while the other gate inputs remain constant. Therefore, the NLDM is chosen to be the simple gate model within the heterogeneous gate mode.

The high efficiency of the NLDM is achieved by the extensive use of Look-up tables (LUTs). Various LUTs have already been characterized for each gate representing the gate delay and the output transition time.

A NLDM cell library contains, for each cell, four lookup-tables representing the rise delay, fall delay, rise transition and fall transition from each input pin to the output pin of the gate [39].

The input waveform at any gate input is defined as a Piecewise linear waveform (PWL) with the 0% and the 100% points of the voltage waveform represented as (time, voltage) pairs. Therefore, the input transition time is the difference between the times at 0% and 100% voltage points. Let (t_1, v_1) be the (time,voltage) pair corresponding to the 0% voltage point and (t_2, v_2) be the point corresponding to 100% point. Then the input transition time is $t_2 - t_1$.

It is assumed that a load capacitance of fixed size is attached to the gate output. The size of this capacitance is computed from the interconnect parasitic capacitances, parasitic resistances and the sum of the input capacitances of all the receiving gates.

In the following, the extraction of the gate delay and the output transition times from the tables is as explained. Each table represents a two dimensional function, which maps the given pair of input transition time and output capacitance to either the gate delay or the output transition time. In the special case that a table entry exactly matches the given pair of input transition time and output capacitance, the timing value corresponds directly to the values stored in the table and hence the table becomes trivial.In case the values of the input transition time and the output capacitance do not correspond to the table entries, then a 2-D interpolation of the four nearest neighbours [40] is performed to extract the timing values.

Suppose x_1 and x_2 are the two input slope values and y_1 and y_2 are the two values corresponding to the output capacitance. Let T_{11}, T_{12},T_{21} and T_{22} be the corresponding table values. The interpolation formula for the delay values for the pair (x_0,y_0) is then defined as:

$$T_{00} = x_{20} * y_{20} * T_{11} + x_{20} * y_{01} * T_{12} + x_{01} * y_{20} * T_{21} + x_{01} * y_{01} * T_{22} \tag{4.4}$$

where the intermediate terms x_{01}, x_{20}, y_{01}, y_{20} are calculated as :

$$x_{01} = (x_0 - x_1)/(x_2 - x_1) \tag{4.5}$$
$$x_{20} = (x_2 - x_0)/(x_2 - x_1) \tag{4.6}$$
$$y_{01} = (y_0 - y_1)/(y_2 - y_1) \tag{4.7}$$
$$y_{20} = (y_2 - y_0)/(y_2 - y_1) \tag{4.8}$$

The gate simulation with the simple gate model uses the above mentioned formula for calculating the gate delay and the output transition times at the output of the logic gate. Then, the transition is appended to the PWL function, by storing the 0% and 100% cross-over points. Let d be the pin-to-pin delay of the gate from input X and ot be the output transition time, both corresponding to a particular input transition time and output load. Then the arrival time of the output transition is defined as :

$$A^{\mathrm{Z}} = A^{\mathrm{X}} + d \tag{4.9}$$

Now, since the output transition time (ot) is also known from the LUTs, the 0% and the 100% points can be easily determined as follows:

$$t_1 = A^{\mathrm{Z}} - (ot/2) \tag{4.10}$$

Similarly,

$$t_2 = A^{\mathrm{Z}} + (ot/2) \tag{4.11}$$

For a falling output transition, t_1 represents the time at which the output is at $\mathrm{V_{dd}}$ and t_2 is the time at which the output falls to 0V. In case of a rising transition at the output, t_1 represents the time at which the output is at 0V and t_2 is the time at which the output rises to $\mathrm{V_{dd}}$.

For the use of the NLDM within the proposed heterogeneous gate model, several modifications are necessary. At first, an input transition consisting of more than two time/voltage pairs must be approximated by two time/voltage pairs (t_1, v_1) and (t_2, v_2). This is done by finding the 10% and 90% voltage crossing points.

Second, the heterogeneous gate model is required to produce a current waveform at the gate output. Therefore, the gate output current during a transition must be computed. If C denotes the capacitance at the gate output and V denotes the gate output voltage, then the gate output current is $I = C * dV/dt$. Since the output voltage changes linearly with the time t, the gate output current I_1 is

$$I_1 = C * (v_2 - v_1)/(t_2 - t_1) \tag{4.12}$$

during a gate output transition from (t_1, v_1) to (t_2, v_2) and zero otherwise.

The time intervals during which the current flows are determined from the equations 4.10 and 4.11.

4.4 Advanced Gate Model for a CMOS Inverter

In [41], a very accurate current source gate model was presented, which can be used to efficiently simulate multiple transitions at the gate inputs and was selected as the advanced gate model.

Section 4.4.1 summarizes the most important cell characteristics of a CMOS inverter, which have a significant impact on the gate output current. This analysis leads to the construction of an accurate current source model in section 4.4.2. The characterization of all model parameters is finally explained in section 4.4.3.

4.4.1 Important Cell Characteristics of a CMOS Inverter

One of the most important cell characteristics is the DC current through the gate output, which depends only on the instantaneous voltages at the gate inputs and outputs.

At any time instant, the output current of the gate charges/discharges the load capacitances and the parasitic capacitances present at the output of the gate. Thus,

$$I_{\mathrm{dc}}(V_{\mathrm{i}}(t), V_{\mathrm{o}}(t)) = I_{\mathrm{load}} + I_{\mathrm{parasitic}} \tag{4.13}$$

where,

$I_{\mathrm{dc}}(V_{\mathrm{i}}(t), V_{\mathrm{o}}(t))$ is the instantaneous dc output current of the gate

I_{load} current flowing over the gate output, and,

$I_{\mathrm{parasitic}}$ is the current which flows inside the logic cell to charge or discharge the cells internal parasitic capacitances

Of critical importance for the circuit timing simulation is the current I_{load}, which is flowing over the gate output and charges/discharges the parasitic capacitances of the interconnect network and the input capacitances of all receiving gates. It is clear from the above equation that the cells parasitic capacitances can significantly reduce I_{load} and must therefore also be considered as important cell characteristics.

The output parasitic capacitance of any gate is composed of 2 main capacitances - gate-drain capacitance (C_{gd}) and the drain-bulk capacitance (C_{db}). Similar capacitances exist between any of two of the four terminals of a MOSFET, as shows for a p-MOSFET in the following figure.

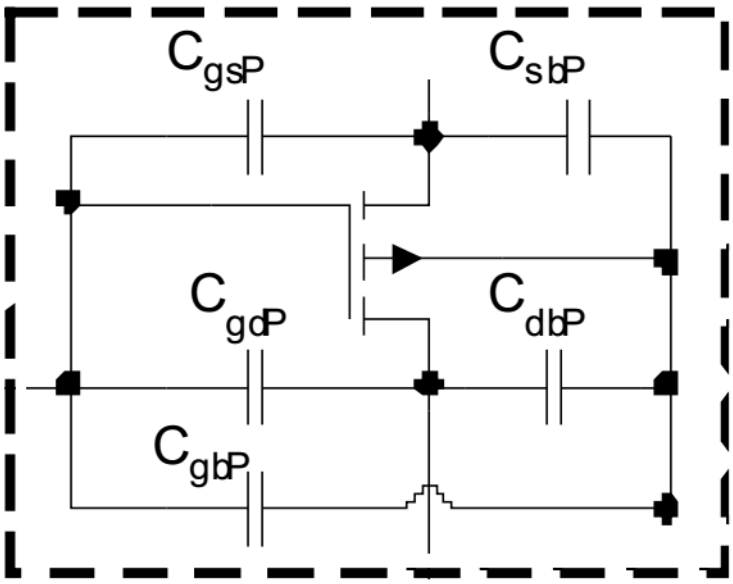

Figure 4.9: p-MOSFET parasitics [1]

The value of these parasitic capacitances depend on the terminal voltages of the gate and hence it varies during a transition at the input/output of the gate. In general, drain-bulk capacitance C_{db} depends only on the output voltage and the C_{gd} depends on the mode of operation of the transistor and hence the terminal voltages. For any switching input of the gate, C_{gd} depends on both input and output voltages which represents the miller capacitance. For the accurate simulation of a library cell, many more parasitic capacitances and resistances arise from the cell layout, as shown in the following figure.

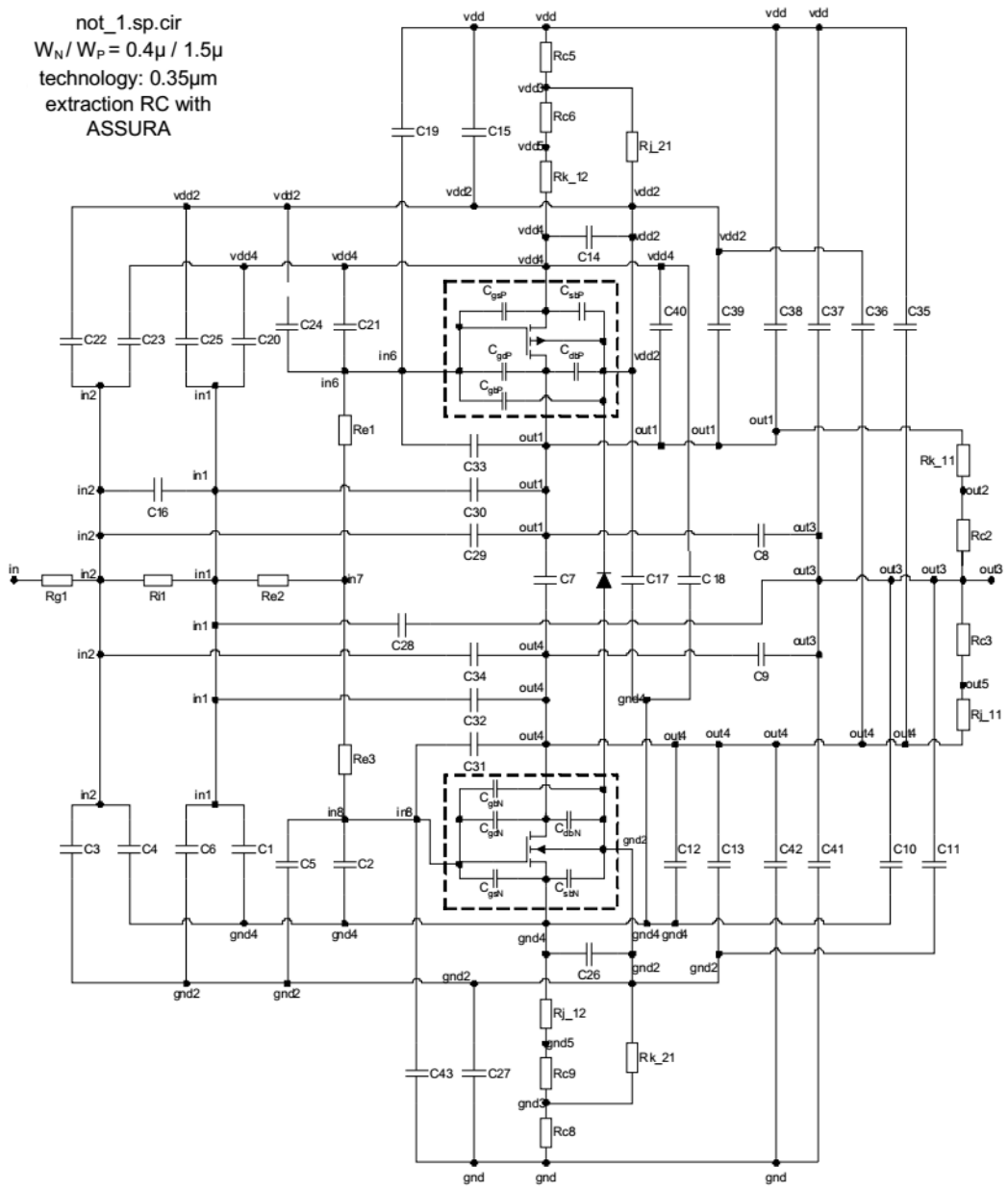

Figure 4.10: Schematic of inverter gate after layout extraction [1]

Of particular importance are:

- The capacitances between the gate input “in” and ground “gnd” or power-supply pin “vdd”, which determine the input capacitance of the inverter.

- The coupling capacitances between the input “in” and the output “out3”.
- The capacitances between the output “out3” and ground “gnd” or power-supply pin “vdd”, which determine the gate output capacitance.

This analysis leads to the current source model in the next subsection.

4.4.2 Description of Current Source Model

Any logic gate is modelled as a non-linear current source with accurate non-linear capacitances.

For an inverter gate with only one switching input, the complete gate model can be represented by the figure below :

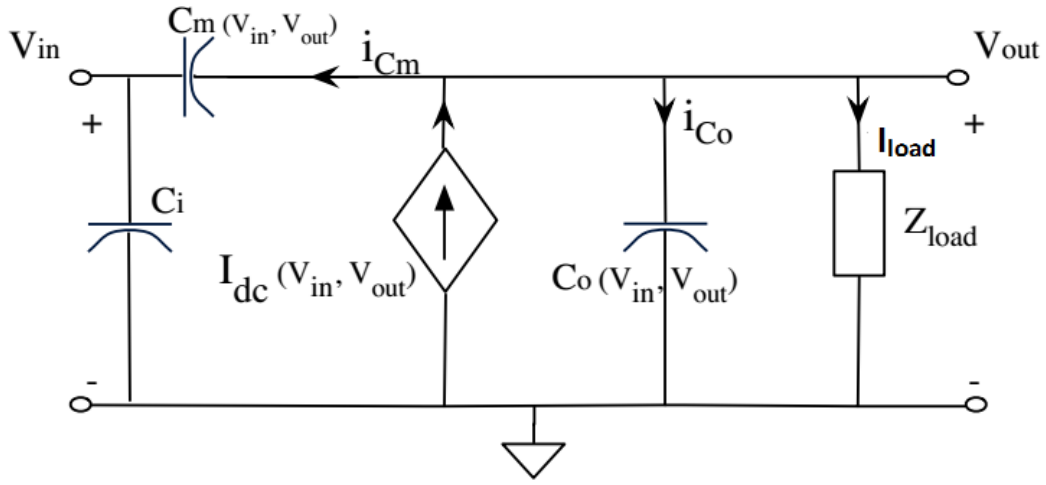

Figure 4.11: Gate model for single input switching [41]

In the above figure, C_i represents the receiver capacitance as seen by the gate input, $C_m(V_{in},V_{out})$ represents the miller capacitance at the switching gate input, $I_{dc}(V_{in},V_{out})$ represents the non-linear gate output current, $C_o(V_{in},V_{out})$ represents the sum of all parasitic capacitances at the output of the gate, and, Z_{load} represents the load connected at the output of the gate.

The miller capacitance C_m is simply a coupling capacitance between the input and output terminals of a gate. It causes an amplification effect on the coupling capacitances (gate-source, gate drain capacitance) at the input and output terminals of the gate. That is because this coupling capacitor (miller capacitor) experiences a voltage swing at the gate input and simultaneously a voltage swing of opposite direction at the gate output. The presence of the miller capacitance results in the change of the input capacitance of the gate. The importance of the miller capacitance lies in the fact that if the changed input capacitance if not captured accurately, it results in causing significant variations in the timing simulations of circuits. Hence, the transient analysis of any logic gate using the above mentioned gate model depends significantly on the Miller capacitance along with the output load capacitance at the gate output.

Using Kirchhoff's Current Law at the gate output node, the gate output DC current in Fig 4.11 at any time instant t can be written as :

$$I_{dc}(V_{in}, V_{out}) = i_{C_m} + i_{C_o} + I_{load} \tag{4.14}$$

$$= C_m(V_{in}, V_{out}) \left(\frac{dV_{out}}{dt} - \frac{dV_{in}}{dt} \right) + C_o(V_{in}, V_{out}) \frac{dV_{out}}{dt} + I_{load} \tag{4.15}$$

Therefore, the gate output current waveform at any time instant t is defined as

$$I_{load} = I_{dc}(V_{in}, V_{out}) - C_m(V_{in}, V_{out}) \left(\frac{dV_{out}}{dt} - \frac{dV_{in}}{dt} \right) - C_o(V_{in}, V_{out}) \frac{dV_{out}}{dt}. \tag{4.16}$$

4.4.3 Current Source Model Characterisation

The parameters of the current source and the parasitic capacitances depend on the gate terminal voltages. Therefore, it is necessary to measure the current and the capacitances for many different combinations of gate input and output voltages. Although this demands some time for characterization, it only has to be done once for each library cell.

The characterization of the dc output current and the parasitic capacitances of the gate was performed by a supervisor.

4.4.3.1 Characterization of DC Output Current

The dc output current corresponding to instantaneous input voltage, $V_{in}(t)$ and instantaneous output voltage $V_{out}(t)$ is determined using SPICE transient analysis. The setup for measuring the gate output current of `INV_X1` gate in SPICE is as shown below:

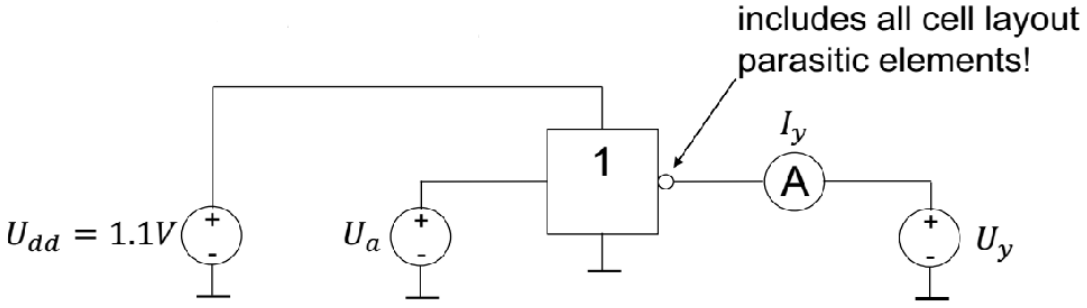

Figure 4.12: Current characterization for `INV_X1` gate

By performing DC analysis in SPICE, we can obtain the gate output current as the current flowing through a voltage source connected at the output of the gate. Thus, the instantaneous dc current at the output of a gate can be found by characterizing the gate for different input and output voltages.

The input and output voltage sources are swept from 0 to 1.1V, which is the supply voltage at a step size of 0.1V. Hence, for each combination of input and output voltage, the DC analysis is performed and the current at the output of the gate is measured. This is stored as tables in MySQL. Later, this table is used to determine the DC output current corresponding to any input and output voltage by using B-Spline interpolation. In case of a single-input gate, B-Spline interpolation is performed in 2 dimensions and for a 2-input gate, it is performed in 3 dimensions.

4.4.3.2 Characterisation of Miller Capacitance

The intrinsic capacitance of any logic gate depends non-linearly on the gate input and output voltages. This makes the characterization of the miller capacitance an important step in the implementation of the current source based gate model.

In general, the current through any voltage-controlled capacitor can be represented using the below formula :

$$I = C(v) * dV(t)/dt \tag{4.17}$$

Using the above formula, the Miller capacitance of the advance gate model can be characterised as explained. The characterization setup for extracting the miller capacitance C_m is as shown in the figure below:

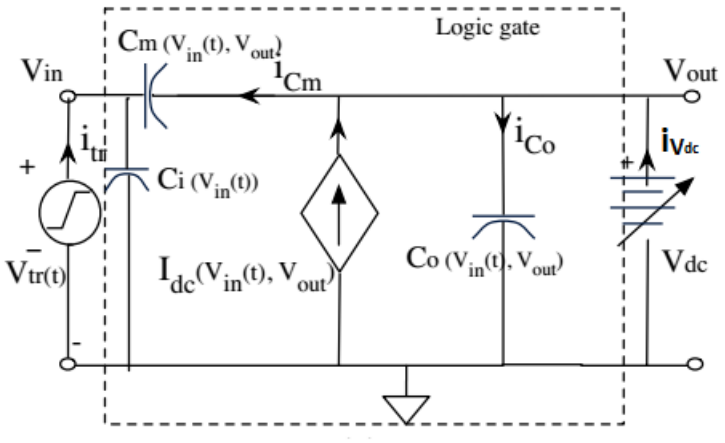

Figure 4.13: Setup to extract the miller capacitance C_m [41]

A dc voltage source is connected to the output of a gate and a ramp signal is applied to its input. The dc voltage source is then swept between 0 and the supply voltage, V_{dd}. For each value of V_{dc}, transient analysis is performed in SPICE. Applying Kirchhoff's Current Law (KCL) at the output node :

$$I_{dc}(t) = i_{C_m}(t) + i_{C_o}(t) - i_{V_{dc}}(t) \tag{4.18}$$

where, $I_{dc}(t)$ is the instantaneous gate output current at time instant t, $i_{V_{dc}}$ is the current flowing from the dc voltage source , and, i_{C_o} is the current through the equivalent grounded capacitance at the gate output.

From the above 2 equations, the miller capacitance $C_m(t)$ can be written as :

$$C_m(t) = \frac{I_{dc}(t) + i_{V_{dc}}(t)}{-k} \tag{4.19}$$

where, k is the slope of the input applied V_{in}

Thus, for a particular V_{dc}, miller capacitance value can be obtained for different time instants during the applied input transition. This is repeated for various values of V_{dc} to obtain the complete characterization of miller capacitance data for different V_{in} and V_{out} values.

4.4.3.3 Characterization of Output Capacitance

The gate output capacitance can also be characterized using the following characterisation setup. A DC voltage source is connected to the input of the gate and a ramp signal is applied to the gate gate output. The dc voltage source is then swept between 0 and the supply voltage, V_{dd}. For each value of V_{dc}, transient analysis is performed in SPICE.

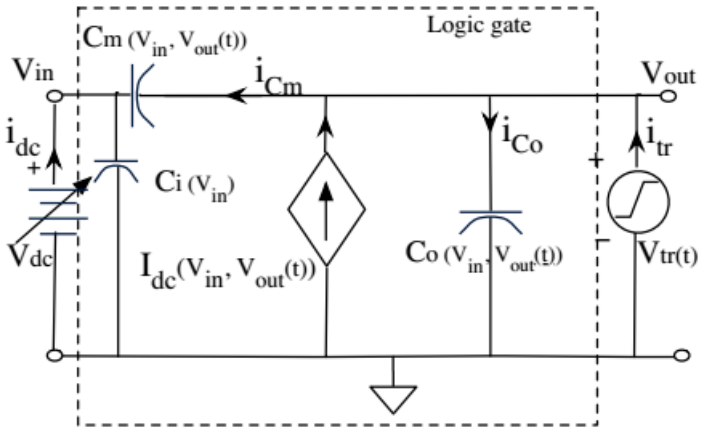

Figure 4.14: Setup to extract the output capacitance C_o of the inverter [41]

When KCL is applied to the output node, the current equation can be written as:

$$I_{dc}(t) = i_{C_m}(t) + i_{C_o}(t) - i_{tr}(t) \tag{4.20}$$

Hence, expanding the above equation, output grounded capacitance at any time instant t can be written as:

$$C_{\mathrm{o}}(t) + C_{\mathrm{m}}(t) = \frac{I_{\mathrm{dc}}(t) + i_{\mathrm{tr}}(t)}{m} \tag{4.21}$$

where, m is the slope of the voltage applied at the output, $i_{tr}(t)$ is the current through the output voltage source at time instant t.

4.5 Advanced Gate Model for 2-input CMOS gate

The advanced gate model discussed above for an inverter gate can be extended for 2-input CMOS gates as well. The number of inputs that can switch simultaneously so that a MIS case results is 2. Therefore, for a 2-input CMOS gate with MIS at both inputs, the gate can be modelled using 2 Miller capacitance for the two inputs and an equivalent grounded capacitance which is the sum of all parasitic capacitances at the gate output. This is depicted in the figure below:

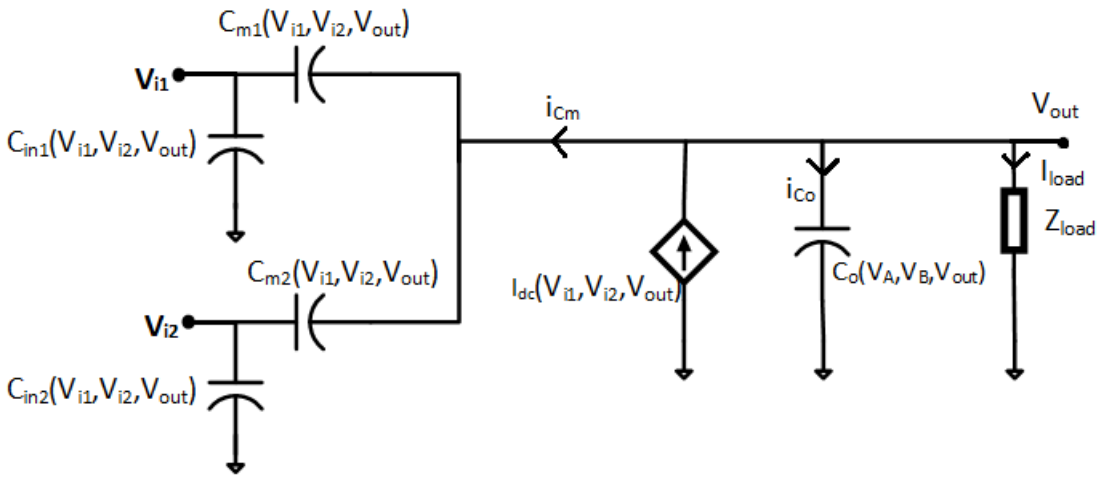

Figure 4.15: Gate model for 2-input CMOS gate with MIS (adopted from [41],[36])

The C_{m1} and the C_{m2} are the miller capacitances at the 2 inputs "i1" and "i2" of the gate. C_o is the equivalent grounded capacitances at the gate output. These capacitors are modelled in the same way as explained in section 4.4.3.

Using Kirchhoff's Current Law at the gate output node, the gate output dc current in

Fig 4.15 at any time instant t can be written as:

$$I_{dc}(V_{i1}, V_{i2}, V_{out}) = i_{C_m} + i_{C_o} + I_{load} \tag{4.22}$$

$$\begin{aligned} &= C_{m1}(V_{i1}, V_{i2}, V_{out}) \left(\frac{dV_{out}}{dt} - \frac{dV_{i1}}{dt} \right) \\ &+ C_{m2}(V_{i1}, V_{i2}, V_{out}) \left(\frac{dV_{out}}{dt} - \frac{dV_{i2}}{dt} \right) \\ &+ C_o(V_{i1}, V_{i2}, V_{out}) \frac{dV_{out}}{dt} + I_{load} \end{aligned} \tag{4.23}$$

Therefore, the gate output current waveform at any time instant t is defined as

$$\begin{aligned} I_{load} = I_{dc}(V_{i1}, V_{i2}, V_{out}) &- C_{m1}(V_{i1}, V_{i2}, V_{out}) \left(\frac{dV_{out}}{dt} - \frac{dV_{i1}}{dt} \right) \\ - C_{m2}(V_{i1}, V_{i2}, V_{out}) &\left(\frac{dV_{out}}{dt} - \frac{dV_{i2}}{dt} \right) - C_o(V_{i1}, V_{i2}, V_{out}) \frac{dV_{out}}{dt}. \end{aligned} \tag{4.24}$$

Chapter 5

Application to Circuit Simulation

The proposed heterogeneous gate model enables the fast and accurate timing simulation of VLSI circuits. The basic simulation algorithm is explained in section 5.1. The following section 5.2 details the conversion of the current waveform at the gate output into a voltage waveform using numerical integration. The simulation of interconnect parasitic elements is addressed in section 5.3. The last section section 5.4. presents an alternative simulation approach to achieve even greater accuracy.

5.1 Circuit Simulation Algorithm

This section describes the circuit simulation using the proposed heterogeneous gate model. The simulator is provided with a circuit netlist and a set of input vector pairs. An input vector pair is applied to the circuit in the form of PWL functions. The simulator is provided with the gate model and the required equations for simulating any gate as specified in chapter 4. For the simulation of circuits considering the interconnects, the simulator is also provided with the dspf file, which were obtained by VLSI interconnect parasitic extraction from the circuit layout. All the output waveforms are stored in as piecewise linear (PWL) functions.

5.1.1 Traversal of Circuit Netlist

All gates are simulated starting from the ones to which the primary inputs of the circuit are connected. Each primary input is applied as a PWL waveform with (time, voltage) pairs specified.

During the simulation, the circuit is traversed in topological order. Each gate with at least one input transition is simulated, as described in chapter 4. The result of each simulation is a current waveform at the gate output, which is converted into a voltage waveform using Euler's method of numerical integration. The output voltage waveform is stored as a PWL function of (time,voltage) pairs, which can later be used as the inputs to simulate the receiving gates. This process continues until all gates with at least one

input transition have been simulated. The flowchart of the circuit simulation algorithm is shown in Figure 5.1.

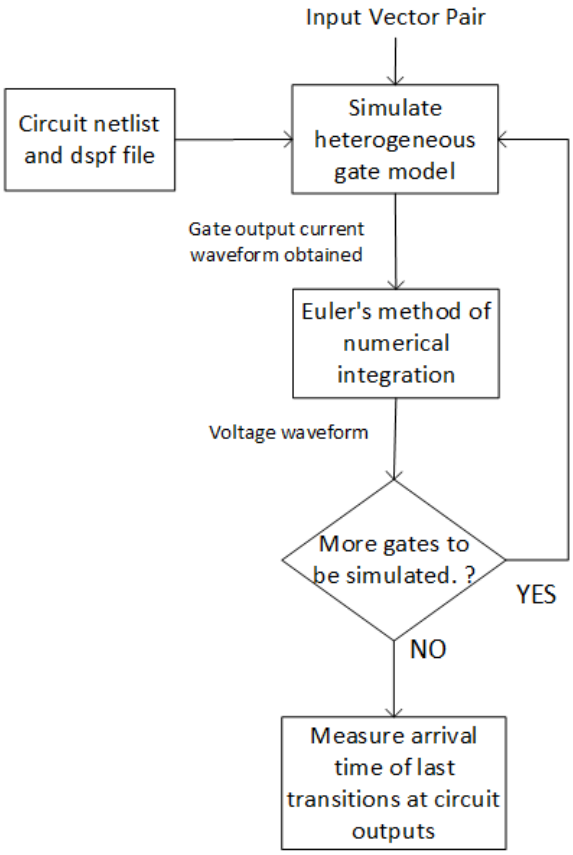

Figure 5.1: Flowchart of circuit simulation based on heterogeneous gate model

Once all the gates in the circuit are simulated, then the arrival time of the last transition at each primary output is measured to determine the delay of the circuit for the given input vector-pair. The delay of the circuit is defined as the time difference between the 50% voltage points of the primary output and the primary input.

5.1.2 Summary of Heterogeneous Gate Model Simulation

If the gate to be simulated is an inverter gate, it is simulated using the simple gate model and the output voltage waveform is obtained as explained in section 5.2.1. Note that the total output capacitance of the gate being simulated is modified by considering the lumped capacitance (as described in section 3.3.1)of the wire in addition to the sum of input capacitance of all receiving gates. This waveform obtained by simulating the gate using the NLDMs is stored as a PWL waveform where the 0 and the 100 % points are stored as (time, voltage) pairs.

In case the output transition of a multi-input gate with only one switching input must be computed, then the simple gate model is simulated as it provides sufficient accuracy in this case.

In case of multi-input gates with multiple input switching (MIS), determined from the classifier described in section 4.1.3, the advanced gate model is used to simulate the gate output transition. The output voltage waveform points are stored for numerous time points as (time, voltage) pairs as in the case of simple gate model with the difference being that in the advanced model, the non-linear output voltage waveform of the gate output is accurately captured. This accuracy is as a result of using the non-linear output current of the gate which is accurately described by the advanced gate model, especially taking the MIS into effect. These voltage waveforms are fed as the inputs to the receiving gates in the next logic levels.

5.2 Computation of Gate Output Voltage Waveform

The following subsections explains how the voltage waveform at any gate output is obtained from the current waveforms produced by the heterogeneous gate model proposed in Chapter 4.

5.2.1 Output Voltage Transition using Simple Gate Model

The simple gate model of the heterogeneous gate model is designed based on the NLDM. The output current from this model does not depend on the instantaneous output voltage. For a given current waveform $I_{load}(t)$, the voltage over the load capacitance C_{load} is

$$V(t) = \frac{1}{C_{load}} \int_0^t I_{load}(t)dt + V(0). \tag{5.1}$$

The current waveform $I_{\text{load}}(t)$ is a piecewise constant function and therefore the voltage waveform $V(t)$ is a piecewise linear function. This voltage waveform can be very efficiently computed. For example, the voltage transition in the time interval $t \in [t_1, t_2)$ is

$$V(t) = \frac{1}{C_{load}} I_{load}(t) * (t - t_1) + V(t_1) \tag{5.2}$$

Thus, the output voltage waveform can be obtained very efficiently.

5.2.2 Output Voltage Transition using Advanced Gate Model

The advanced gate model explained in the previous chapter can be used for obtaining the transient response at the output of any logic gate. The voltage waveform at the output of the gate is computed from the the current waveform, by using Euler's method of numerical

integration. As it is already known, the gate output current for any instantaneous input and output voltage can be obtained for the advanced gate model as explained earlier.

Similar to the previous subsection 5.2.1, the integral over the current waveform from time 0 to time t gives the charge $Q(t)$ of the load capacitance C_{load} at time t, assuming that the output capacitance was initially discharged ($Q(0) = 0$). Then the voltage at time t is V(t)=Q(t)/C_{load}. However, it must be considered that the gate output current depends on the output voltage. To do this, the simulation time is split into small time steps Δt and Euler's method of numerical integration is applied.

Considering the load capacitance C_{load}, we can rewrite the current given by equation 4.16 at any time instant as follows[41]:

$$I_{\mathrm{dc}}(V_{\mathrm{in}}(t), V_{\mathrm{out}}(t)) = (C_{\mathrm{load}} + (C_{\mathrm{m}} + C_{\mathrm{o}}))(V_{\mathrm{in}}(t), V_{\mathrm{out}}(t))(\frac{V_{\mathrm{out}}(t+\Delta t) - V_{\mathrm{out}}(t)}{\Delta t}) - C_{\mathrm{m}}(V_{\mathrm{in}}, V_{\mathrm{out}})(\frac{V_{\mathrm{in}}(t+\Delta t) - V_{\mathrm{in}}(t)}{\Delta t}) \quad (5.3)$$

If $V_{\mathrm{in}}(t+\Delta t) - V_{\mathrm{in}}(t)$ is represented as ΔV_{in}, then the output voltage at time $(t+\Delta t)$ is given by,

$$V_{\mathrm{out}}(t+\Delta t) = V_{\mathrm{out}}(t) + \frac{I_{\mathrm{dc}}(V_{\mathrm{in}}(t), V_{\mathrm{out}}(t))\Delta t + C_{\mathrm{m}}(V_{\mathrm{in}}(t), V_{\mathrm{out}}(t))\Delta V_{\mathrm{in}}}{C_{\mathrm{load}} + (C_{\mathrm{m}} + C_{\mathrm{o}})(V_{\mathrm{in}}(t), V_{\mathrm{out}}(t))} \quad (5.4)$$

At time t = 0, initial voltage at the input and output of the gate, V_{in} and V_{out} are known. The dc gate output current corresponding to input and output voltages are evaluated for t = 0. Thus, the voltage for the next time instant $(t + \Delta t)$ is evaluated using the above formula. This process is repeated until the V_{out} reaches the desired value. Thus, using the advanced gate model, the exact non-linear output waveform at the gate output can be determined and stored.

From equation (7) in [36], the Euler's numerical integration formula for the 2-input gate is given by:

$$V_{out}(t+\Delta t) = V_{out}(t) + \frac{Q_1(t) + Q_2(t) + Q_3(t)}{C_{load} + (C_m + C_o)(V_{i1}(t), V_{i2}(t), V_{out}(t))}, \quad (5.5)$$

where

$$Q_1(t) = I_{dc}(V_{i1}(t), V_{i2}(t), V_{out}(t))\Delta t \quad (5.6)$$

$$Q_2(t) = C_{m1}(V_{i1}(t), V_{i2}(t), V_{out}(t))\Delta V_{i1} \quad (5.7)$$

$$Q_3(t) = C_{m2}(V_{i1}(t), V_{i2}(t), V_{out}(t))\Delta V_{i2}. \quad (5.8)$$

5.3 Simulation of VLSI Interconnect Parasitic Elements

Interconnect parasitic elements, such as parasitic capacitances and and resistances, have a large impact on the circuit delay. A description of the interconnect parasitic elements of each wire is contained in the dspf (Detailed Standard Parasitic File) which is obtained by interconnect parasitic extraction from a given circuit layout.

5.3.1 Approximation using Lumped C Model

Considering the high computational cost for the simulation of complex interconnect networks, the interconnect parasitic capacitances were approximated by the lumped C model. In case of the advanced gate model, exact simulation of the RC network is done to achieve very accurate results even though the simulation time is quite significant.

In this thesis, the effective capacitance information of each wire is obtained from the dspf file. In the dspf file associated with each circuit, the net wire capacitance for each net is specified in the first line of the net description for each net. For example, consider the description of net **"n1"** of some circuit :

*NET n1 0.2586FF

So the net capacitance of the net n1 is 0.2586 FF. This capacitance is added to the sum of the input capacitances of the receiving gates.

5.3.2 Exact simulation of RC interconnects for advanced gate model

Another method chosen to simulate the parasitic RC network of interconnects, is by using the ngspice shared library provided in C++.

For any gate, the RC parasitic wire network at the output terminal can be represented as a RC network as shown in the figure below:

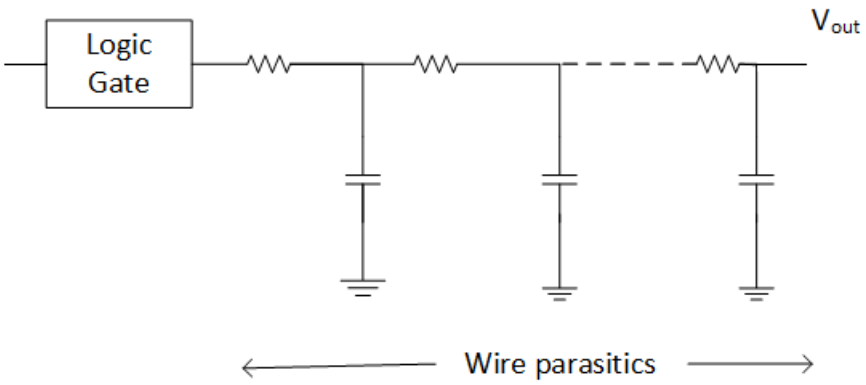

Figure 5.2: RC interconnect representation between a driving gate output and a receiving gate input (not shown)

This can be simplified by replacing the driving gate with a non-linear current source which represents the gate current I_{load}, followed by the RC interconnect network details, along with the input capacitance of the receiving gate at node. This is illustrated in the figure below for the case of two receiving gates "Gate 1" and "Gate 2":

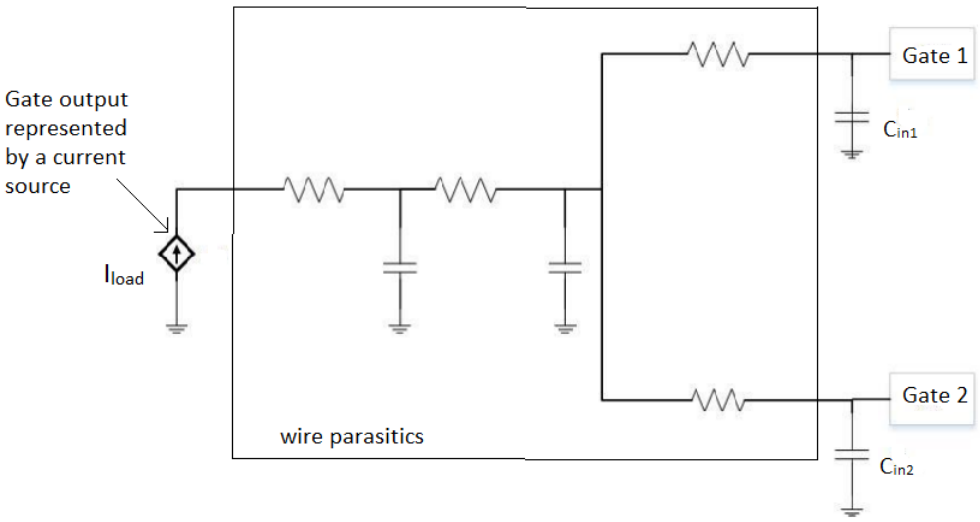

Figure 5.3: Simplified gate model for advanced algorithm

The above RC network is passed to the ngspice share library for transient analysis. The input to the circuit is a current source which is defined as "EXTERNAL" in ngspice. The value of this current source at any instant is later retrieved from the caller using callback functions defined in the library. The voltage waveform at each output node is computed and stored as a PWL waveform.

There are generally 2 methods to load the ngspice library (*dll library). Firstly, the caller can link to the library during compiling and then search for the library upon start or

secondly, it is also possible to start the ngspice shared library dynamically at runtime using the dlopen/LoadLibrary mechanism.

The input to the ngspice is in the netlist of the RC network to be simulated with an EXTERNAL current source. This simulation may be started in a separate thread and the result can be read back at each time point. The details of the functions for using the ngspice as a shared library [42] is explained in the following subsections.

5.3.2.1 Initialization of Ngspice shared library

int ngSpice_Init(SendChar*, SendStat*, ControlledExit*, SendData*, SendInitData*, BGThreadRunning*, void)

After the ngspice library has been loaded, the caller has to call this init function to initialise the simulator. This function, mainly passes the address pointers of several callback functions that may be used in the simulator to `ngspice.dll`. The int return value is not used.

NULL is allowed to be passed in case some functions are not used, except for `ControlledExit*` which is essential to exit from the library.

int ngSpice_Init_Sync(GetVSRCData*, GetISRCData*, GetSyncData*, int*, void*)

here,

`GetVSRCData*` - callback function for retrieving a voltage source value from the caller.

`GetISRCData*` - callback function for retrieving a current source value from the caller.

`GetSyncData*` - callback function for synchronization

`void*` represents a pointer to the user-defined data which will not be modified by the function but handed to the caller during callback. In case NULL is sent, the userdata from the initialization is kept, otherwise it is overridden by the data from this function

5.3.2.2 Interaction with Ngspice shared library

The RC network of interconnects to be simulated using the ngspice shared library has to passed to the library in the form of a netlist. This is done by passing the netlist as a string array.

An array of char pointers, *circarray* is allocated with the required memory and then each line of the netlist is copied to the array. The first line of the array must always be a title line and the last line should always be a NULL. Then **Circ(Circarray)** is used to parse the circuit netlist to ngspice.

As an example, the following netlist is passed to Ngspice :

```
transient analysis for DC current source and RC
```

```
I1 0 I_0:ZN external
*|I (I_0:ZN I_0 ZN O 0.000000FF 8.400 6.305)
*|I (I_1:A I_1 A I 0.565000FF 8.075 6.930)
*|S (s1_2 8.455 6.930)
*|S (s1_3 8.455 6.930)
*|S (s1_4 8.075 6.930)
*|S (s1_5 8.075 6.930)
C1_5 I_0:ZN 0 0.016620FF ic=0
C2_5 I_1:A 0 0.016620FF ic=0
C3_5 s1_2 0 0.015200FF ic=0
C4_5 s1_3 0 0.020250FF ic=0
C5_5 s1_4 0 0.020250FF ic=0
C6_5 s1_5 0 0.015200FF ic=0
R1_5 s1_5 I_1:A 5.000000
R2_5 s1_4 s1_5 5.000000
R3_5 s1_4 s1_3 1.357143
R4_5 s1_2 s1_3 5.000000
R5_5 I_0:ZN s1_2 5.000000
.TRAN 3.2e-12 8e-11 0 3.2e-12 UIC
.end
```

The initial condition (ic=0) is set to specify the initial voltage of the parasitic capacitances.

Once the netlist is parsed, the command `ng_Spice_Command("run")` is used to run the simulation in ngspice.

Afterwards, the callback functions `SendInitData` and `SendData` are used to access the simulation result synchronised with simulation time.

5.4 Using SPICE simulation as Advanced Algorithm

The simulation method described in 5.2.2 is very complex and was not fully implemented due to timing limitations. To simulate the proposed heterogeneous gate model considering all interconnect parasitic elements, another advanced algorithm was used in case of MIS effect in multi-input gates. This algorithm involved the use of ngspice shared library to simulate the multi-input gate with MIS by passing the NanGate Cell representation of the gate itself to be simulated. That is instead of the non-linear current source of the simplified gate model, the gate itself is passed to ngspice. In addition to this, NanGate cell representation of the output driven gates are also passed to represent the input capacitance of the receiving gates, instead of using a constant capacitance. For example, in case of a 2 input NOR gate with MIS which drives a NOT gate and a NOR2 gate at its output, it can be represented as follows :

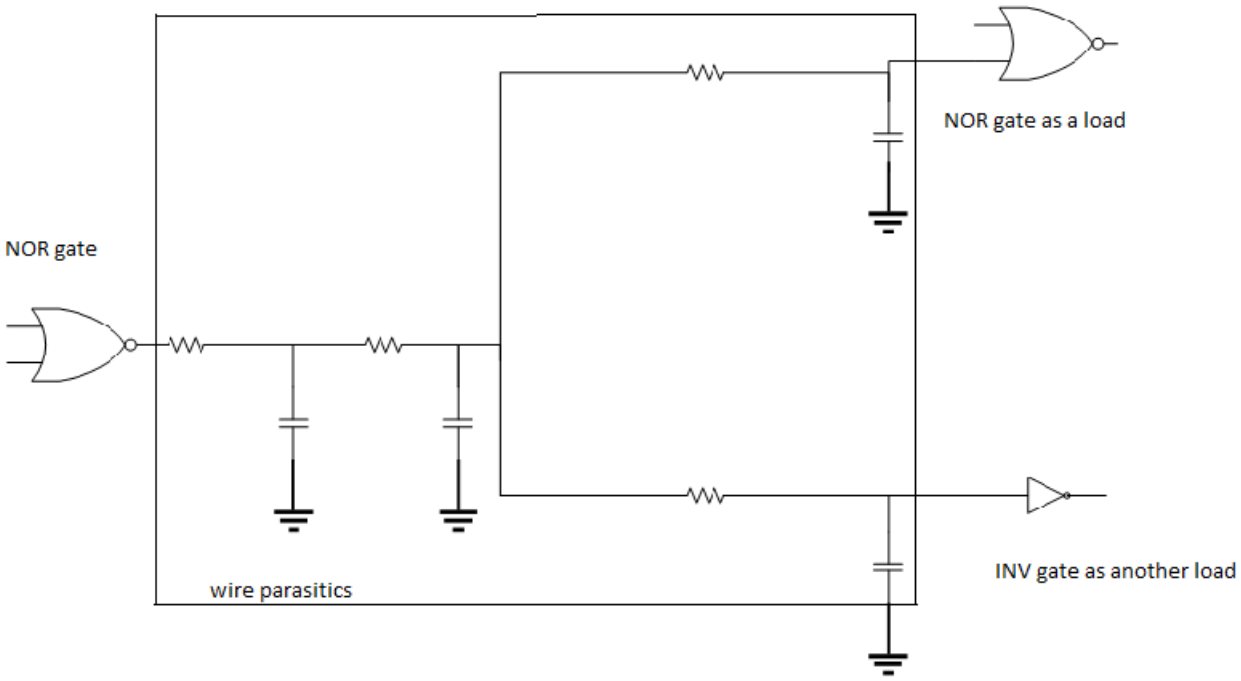

Figure 5.4: Gate model for NOR2 gate with interconnects and NanGate Cells as loads

The steps of the circuit simulator are the same as mentioned in the previous section. This method ensures more accurate results than the proposed gate model. Although, the output waveform returned by ngspice has many data points (depending on the step size of transient simulation in ngspice), in order to optimize memory requirements and computation time, the output waveform of a gate is compressed to store the voltage points with a step size of 0.1V. This means that the output voltage waveform only has points starting from 0V or 1.1V (depending on rising or falling transition respectively) with a total of 11 (time, voltage) pairs between these points. The details of the results obtained for various circuits used in this thesis are presented in the next chapter.

Chapter 6

Experimental Results

This chapter comprises of all the experiments carried out to evaluate the heterogeneous gate model for circuit simulation. It is composed of 3 parts - first part which describes the various benchmark circuits used for the evaluation of the proposed gate model. It also discusses how the input vectors are chosen for the simulation of benchmark circuits. The second part explains the reference simulations of the benchmark circuits with a gate level and circuit level simulator, and the final part, which presents the results obtained with the proposed gate model. The speed and the accuracy of the timing simulation using the proposed gate model are compared with the gate level simulator and the circuit level simulator.

The proposed gate model was implemented for a 45nm technology using the BSIM4 predictive technology models [43]. All experiments were performed on a workstation with a 3.4 GHz processor and 16 GB RAM.

6.1 Benchmark Circuits

The proposed gate model was evaluated for small and medium sized circuits to evaluate the efficiency of the proposed gate model as well as the scalability of the circuit simulation.

All circuits were synthesized using a commercial synthesis tool and mapped to only `INV_X1` and `NOR2_X1` cells from the NanGate Open Cell Library [39]. During the synthesis, all contributions of interconnect parasitic capacitances to the circuit delay were ignored. The resulting delay information (SDF-file) was used by HDLSIM for the circuit simulation without considering interconnect parasitic capacitances.

The proposed gate model was also evaluated for the circuit simulation including all interconnect parasitic elements. For that purpose, circuit layouts were created using a commercial place&route tool. The resulting delay information (SDF-file) was used by HDLSIM for the circuit simulation considering all interconnect parasitic capacitances. Finally, interconnect parasitic extraction was performed, which produced a DSPF-file (Detailed Standard Parasitic Format) that was used by HSPICE and the simulator of the proposed gate model.

The synthesis and the creation of the circuit layouts was done by a supervisor.

6.1.1 Small Circuits for Gate Model Evaluation

The proposed gate model was used for the accurate timing simulation of various circuits listed below :

- Inverter Chain
- Chain of NOR gates
- Chain of NOR and INV gates

The above mentioned circuits were used for the design and development of the circuit simulator using the gate model proposed in this thesis.

The inverter chain consisted of a chain of 7 inverter gates. The chain of NOR gates consists of 7 2-input NOR gates connected together, with the 2-inputs of each gate tied together. The circuit diagram of these 2 circuits are shown below:

Figure 6.1: Inverter chain (`inv_chain`)

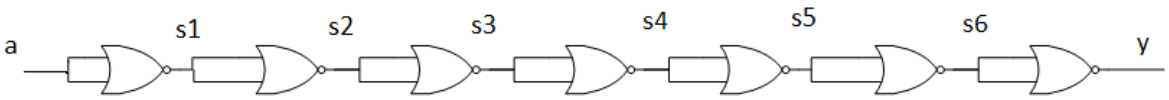

Figure 6.2: Chain of 2-input NOR gates (`nor2_chain`)

The third circuit is a chain of a NOR gate and inverter gates which is as shown below:

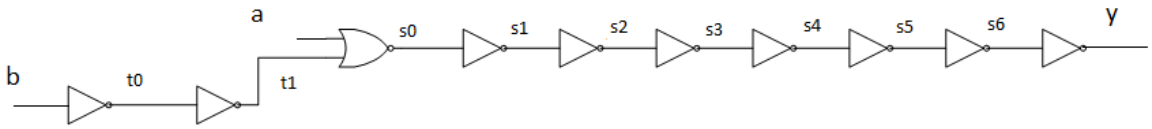

Figure 6.3: Chain of 2-input NOR gate and inverter gate (`nor2_inv_chain`)

The `nor2_chain` (fig 6.2) is the smallest circuit where the MIS case can be investigated as the inputs of all the gates are tied together. This ensures that the transitions at the gate input overlap, which gives rise to a MIS scenario.

6.1.2 Medium Sized Circuits for Simulation Scalability Evaluation

This can be extended to test the scalability of the developed gate model for bigger circuits like ISCAS85 benchmark circuits. These circuits help in validating the classifier designed as the gate model is switched more often between the simple and the advanced gate models. These circuits include:

Binary tree of 2 input NOR gates of degree 3, Ripple Carry Adder, c17, c432, c499, c880, c1355, c1908, c2670, c3540, c5315, c6288, c7552

The above mentioned ISCAS85 circuits are a set of combinational circuits which were proposed in the 1985 International Symposium on Circuits and Systems. These combinational circuits have since then been used for various research work and they provide a basis for the validation of results in the area of test generation. The **c17** circuit is the smallest of the ISCAS85 circuits and it is composed of 6 NAND gates. The c17 circuit has 5 primary inputs (I_1,I_2,I_3,I_4,I_5) and 2 primary outputs (o_1, o_2). The circuit netlist of c17 circuit is as shown below:

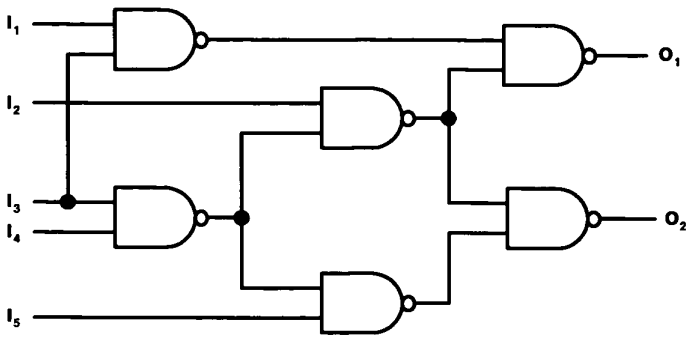

Figure 6.4: Circuit netlist of ISCAS85 c17 circuit [44]

The table below shows a brief description of the other ISCAS85 circuits:

Table 6.1: ISCAS85 circuit characteristics

Circuit Name	Circuit Function	#gates	#PI	#PO
C432	Priority Decoder	160	36	7
C499	ECAT	202	41	32
C880	ALU and Control	383	60	26
C1355	ECAT	546	41	32
C1908	ECAT	860	33	25
C2670	ALU and Control	1193	233	140
C3540	ALU and Control	1669	50	22
C5315	ALU and Control	2307	178	123
C6288	16-bit multiplier	2406	32	32
C7552	ALU and Control	3512	207	108

In the above figure, the column "#PI" represent the number of primary inputs of the circuit and the column "#PO" represent the number of primary outputs of the circuit. The column "#gates" shows the number of gates in the benchmark circuit before synthesis.

6.2 Input Vector-Pairs used for Simulation

The accurate timing simulation of circuits using the proposed gate model involves the application of certain input transitions to the primary input of the circuit and measuring the arrival time of the last transition at the primary outputs of the circuits.

The input vector pairs for the simulation of various circuits were generated using a commercial Automatic Test Pattern Generation (ATPG) tool and provided by a supervisor.

Each input vector-pair creates a set of transitions at the primary inputs of the circuit, which are used during circuit manufacturing test to distinguish between the correct and a faulty behaviour of the circuit. In this thesis, many input vector-pairs were created for each benchmark circuit, which for example sensitize the critical path of the circuit. In addition, there are also vector pairs that can activate possible transition faults in a faulty circuit. A fixed number of such input vector pairs are pre-calculated for each circuit and these are stored in a MySQL database and utilised to evaluate the proposed gate model for the respective circuits.

6.3 Reference Simulations

All benchmark circuits were simulated with all available input vector-pairs using a fast gate-level simulator HDLSIM and a very accurate commercial circuit-level simulator

HSPICE. Each simulation is performed twice. In the first run, all interconnect parasitic capacitances are ignored. In the second run, the parasitic capacitances of the interconnects are considered by both HDLSIM and HSPICE. At the end of each simulation, the arrival time of the last transition at each primary output is measured and stored in a MySQL database. Furthermore, the runtime of the circuit simulation is also measured and also stored in the database.

6.3.1 Using Circuit-Level Simulator HSPICE

To evaluate the accuracy of the proposed gate model, the precise arrival time of the last transition at each primary output is required for comparison. For this purpose, a very accurate commercial SPICE software HSPICE [45] was used to simulate all input-vector pairs for all circuits, both without and with the interconnect parasitic capacitances.

Each simulation is performed twice. In the first run, all interconnect parasitic capacitances are ignored. In the second run, the DSPF-file obtained by interconnect parasitic extraction is used to consider the impact of the interconnect parasitic elements on the circuit delay.

At first, an input vector pairs to be applied to the primary inputs of the benchmark circuit is loaded from the database. By defining suitable voltage sources for all primary inputs, all the primary input transitions are applied simultaneously and with an input transition time of 1 ps. Afterwards, the SPICE netlist is extended by including the 45nm predictive technology files for the transistors, the NanGate Open Cell Library description of the `INV_X1` and `NOR2_X1` cells and specifying the `.tran` statement for transient simulation. Finally, a set of `.measure` statements are added to measure the arrival time of the last transition at each primary output.

The interconnect parasitic elements are obtained from the detailed standard parasitics file (DSPF) file after interconnect parasitic extraction from the circuit layout.

To automate this process, a bash script was created to perform the following steps for each input vector pair:

- Generation of the circuit netlist
  - Inclusion of all library files and the 45nm technology files required for simulating the circuits.
  - Definition of voltage sources for each primary input according to chosen input vector pair.
  - Netlist of the circuit to be simulated.
  - SPICE command for transient analysis of the circuit.
  - Measurement of the arrival time of the last transition at all primary outputs of the circuits.
- SPICE simulation of the circuit netlist generated.
- Storing of the results from SPICE simulation in database.

6.3.2 Using Gate-Level Simulator HDLSIM

To evaluate the speed of the simulation using the proposed gate mode, all input vector-pairs were simulated with a very fast in-house gate-level simulator HDLSIM. For each input vector-pair the simulation was done using the SDF-file after the synthesis, which ignored interconnect parasitic capacitances. The simulation was then repeated with the SDF-file obtained by the commercial place&route tool, which considered all parasitic capacitances and resistances. The runtime and the arrival time of the last transition at each primary output was provided by a supervisor and stored in the MySQL database.

6.4 Circuit Simulation Results

The proposed heterogeneous gate model was implemented in the circuit simulator CIRSIM. Similar to the computation of the reference results in the previous section, all input vector-pairs for a benchmark circuit were simulated using CIRSIM, with and without the consideration of the interconnect parasitic capacitances.

In the following, the abbreviation CS is used to identify the results using CIRSIM and the abbreviation SP denotes the results obtained by HSPICE. Similarly, the abbreviation HS is used to identify the results for HDLSIM.

After each simulation, the arrival time of the last transition at each primary output of the benchmark circuits was measured. These measurements are compared to the reference SPICE results. The absolute difference between these 2 values determine the error of the designed gate model.

For a given input vector-pair, the simulation error of CIRSIM and HDLSIM at the i-th primary output is defined as

$$\epsilon_{CS,i} := |T_{SP,i} - T_{CS,i}| \tag{6.1}$$

$$\epsilon_{HS,i} := |T_{SP,i} - T_{HS,i}|, \tag{6.2}$$

where $T_{SP,i}$, $T_{CS,i}$ and $T_{HS,i}$ denote the arrival time of the last transition at primary output i obtained by HSPICE, CIRSIM and HDLSIM, respectively.

The average (maximum) simulation error $\bar{\epsilon}_{CS}$ ($\hat{\epsilon}_{CS}$) of CIRSIM is defined as the average (maximum) value of $\epsilon_{CS,i}$ computed over all primary outputs and input vector-pairs for each circuit. The average simulation error $\bar{\epsilon}_{HS}$ and maximum simulation error $\hat{\epsilon}_{HS}$ of HDLSIM are defined in the same way. The average and maximum simulation errors of CIRSIM and HDLSIM are presented for each benchmark circuit in the following subsections in the form of tables.

To represent the average runtime of the simulation of a vector pair, three columns namely, t_{HS}, t_{CS} and t_{SP} are also presented. t_{CS} represents the average simulation runtime with designed CIRSIM, t_{HS} represents the average runtime with HDLSIM and t_{SP} represents the average runtime required with SPICE.

6.4.1 Results without Interconnect Parasitic Capacitances

The accurate simulation of the smaller circuits like `inv_chain`, `nor2_chain` and the `nor2_inv_chain` were performed using the heterogeneous gate model proposed in this thesis. The result summary for these circuits are shown in the table below :

Table 6.2: Average simulation results for circuits without interconnects

circuit	$\bar{\epsilon}_{HS}(ps)$	$\hat{\epsilon}_{HS}(ps)$	$\bar{\epsilon}_{CS}(ps)$	$\hat{\epsilon}_{CS}(ps)$	$t_{HS}(ms)$	$t_{CS}(ms)$	$t_{SP}(ms)$
inv_chain	18.92	20.32	13.57	14.73	0.0026	2.92	35.00
nor2_chain	6.81	12.05	4.52	5.33	0.0035	108.21	60.00
nor2_inv_chain	23.33	24.21	14.68	15.60	0.0028	4.26	55.00

The circuits `inv_chain` and `nor2_inv_chain` do not have any MIS scenario and hence the classifier chooses the simple gate model using the NLDM to simulate all gates in both the circuits. Therefore, the runtime of the circuit simulator (t_{CS}) as seen from the above table is much less than the SPICE runtime (t_{SP}).

For the `nor2_chain`, all the gates have MIS at the input and hence the classifier chooses the advanced gate model for simulating the gates. The average simulation error ($\bar{\epsilon}_{CS}$) of the designed circuit simulator is much better than the average simulation error of HDLSIM ($\bar{\epsilon}_{HS}$).

6.4.2 Results with Interconnect Parasitic Capacitances

This subsection presents the experimental results for the simulation of the proposed heterogeneous gate model while considering all interconnect parasitic capacitances. For this purpose, the interconnect parasitic capacitances were approximated by the lumped C model, as explained in subsection 5.3.1. The result summary is as shown below:

Table 6.3: Average simulation results with CSM as advanced gate model

circuit	$\bar{\epsilon}_{HS}(ps)$	$\hat{\epsilon}_{HS}(ps)$	$\bar{\epsilon}_{CS}(ps)$	$\hat{\epsilon}_{CS}(ps)$	$t_{HS}(ms)$	$t_{CS}(ms)$	$t_{SP}(ms)$
inv_chain	23.05	23.46	19.74	21.53	0.0030	4.64	50.00
nor2_chain	14.80	22.20	4.09	6.63	0.0031	112.65	75.00
nor2_inv_chain	27.65	28.50	21.13	21.94	0.0027	4.79	65.00

The above table shows that for the `nor2_chain` circuit, where the advanced gate model based on CSM is used, the average error of propagation is very small with on average only 4.09 ps difference to HSPICE result. However, the speed of simulation has to be optimised further. There are several factors like the timestep used for simulation, the memory handling, etc., that determines the runtime of the advanced gate model using the CSM. Once these optimisations are carried out, a better runtime can be achieved using the advanced gate model which in turn would reduce the computational cost required for the simulation of the proposed heterogeneous gate model.

Thus, it is evident that by improving the accuracy of the advanced gate model with considerably efficient runtime, more accurate results can be produced with the heterogeneous

gate model and this can be extended for larger circuits.

6.5 Simulation Results using SPICE as Advanced Algorithm

This section presents the results for an alternative simulation approach, which performs a SPICE simulation of the gate using the Ngspice shared library in case of multiple input switching. In particular, all interconnect parasitic elements between the driving and the receiving gates are considered as explained in section 5.4. The summary of the results using this circuit simulator which uses the NLDM as simple gate model and the ngspice shared library as the advanced gate model is as shown in the table below:

Table 6.4: Average simulation results with ngspice shared library as advanced gate model

circuit	$\bar{\epsilon}_{HS}(ps)$	$\hat{\epsilon}_{HS}(ps)$	$\bar{\epsilon}_{CS}(ps)$	$\hat{\epsilon}_{CS}(ps)$	$t_{HS}(ms)$	$t_{CS}(ms)$	$t_{SP}(ms)$
nor2_tree	4.31	10.89	14.32	31.24	0.0042	104.93	51.43
inv_chain	23.05	23.46	19.74	21.53	0.0030	4.64	50.00
nor2_chain	14.80	22.20	13.19	18.62	0.0031	952.62	75.00
nor2_inv_chain	27.65	28.50	21.13	21.94	0.0027	4.79	65.00
c17	6.66	14.80	6.95	15.16	0.0040	23.52	47.62

The use of ngspice shared library with nangate cells as the advanced algorithm in the heterogeneous gate model has definitely increased the accuracy of the results. As it is evident from the table, the circuits `inv_chain` and the `nor2_inv_chain` chain do not involve any MIS scenario. Hence, they are simulated using the simple gate model where the average error of simulation has better accuracy than HDLSIM and the speed of simulation is also much faster than SPICE.

The `nor2_chain` circuit which encounters MIS at every NOR gate, the advanced algorithm using the ngspice shared library is used. This has definitely improved the accuracy of the simulation results over HDLSIM. However, the accuracy achieved is limited due the output waveform from the ngspice being compressed into a PWL function with lesser number of data points. This shows that the accuracy of the gate model greatly depends on the accuracy of the advanced algorithm while simulating gates with MIS. However, as far as the simulation runtime of CIRSIM is concerned, it is much larger than the SPICE simulation time as a consequence of the use of ngspice shared library. This is mainly due to the fact that several files must be loaded from the NFS server. For example, to load the description of the predictive technology model for the n-MOSFET and p-MOSFET transistors and also several files for the description of the nangate library cells are required. It is well known that the average access time for each file is in the order of tens of milliseconds and this had to be repeated for the simulation of each gate.

The other circuits to be simulated include the `nor2_tree` and the ISCAS85 benchmark circuit. As the table shows, the accuracy of the results for `nor2_tree` is quite bad as compared to HDLSIM. This is the circuit where the classifier switches between the simple and the advanced gate models. The inaccuracy of the results is partly because of

the waveform being compressed after simulating a gate with the ngspice shared library. This error is further increased by the crude approximation of the output waveform for the application of the simple gate model. However, using a sufficiently small time step and with further optimizations, the error introduced by this approach is expected to be minimal.

The results for the c17 circuit seems to give almost the same accuracy as HDLSIM with a reasonably faster simulation speed over the SPICE results, thus proving the claim for a simulator which is faster than SPICE and with good accuracy. The accuracy of simulations can be further improved by using efficient optimisation techniques for the waveform obtained from ngspice.

6.6 Summary of results

This chapter evaluated and compared the accuracy and runtime of an highly optimized in-house gate-level simulator, a highly optimized commercial SPICE simulator with the newly developed simulator CIRSIM, which simulates the proposed heterogeneous gate model. The results show that the idea of combining a simple gate model and a more complex time-consuming advanced gate model gives accurate results. However, the runtime of the simulation hugely depends on the number of gates being simulated using the advanced gate model. The fewer the number of gates simulated using the advanced gate model, the higher is the speed-up achieved over the SPICE calculations. While using the current source model as the advanced gate model, the runtime of simulation is much faster than that achieved using the ngspice shared library. However, there is more scope for optimisations to improve the runtime of the circuit simulator and to make it more efficient than SPICE runtime.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis presented the idea of a realistic heterogeneous gate model by integrating a simple gate model based on NLDMs and an advanced current source based gate model. The simulation of any gate with NLDMs proved to be very fast when compared to SPICE but the accuracy with these models proved to be quite unsatisfactory. One of the main factors that the NLDMs failed to consider was multiple input switching at the gate inputs. In order to handle this, a more advanced algorithm to simulate the gate which is based on a non-linear current source and parasitic capacitances was implemented. In addition, another advanced algorithm using the ngspice shared library was implemented to achieve even greater accuracy. However, this approach could not be fully optimized due to timing limitations.

The heterogeneous gate model proposed in this thesis was implemented in the newly developed circuit simulator CIRSIM. The experiments conducted demonstrate the high efficiency and flexibility of the proposed heterogeneous gate model.

SPICE simulation of large circuits with thousands of transistors, proves to be very time consuming and hence this proposed gate model may save a huge chunk of the computation time with a relatively reduced overall accuracy than SPICE (around 15% or less).

7.2 Future Work

The results of simulation clearly shows that the accuracy of simulation using the heterogeneous gate model depends on the accuracy and runtime of the advanced gate model chosen. This broadens the scope for a numerous optimisation techniques can be employed to improve the accuracy and runtime of the advanced gate model. Furthermore, the proposed gate model can be improved by employing more complex classifiers using Machine Learning. Once accuracy close to SPICE results is achieved, this gate model can also be extended to incorporate glitches and glitch filtering.

Bibliography

[1] I. Brzozowski, "The miller effect in digital cmos gates and power consumption analysis," *Signals and Electronic Systems (ICSES)*, no. 1, pp. 1–6, 2012.

[2] M. Rewienski, *Simulation and Verification of Electronic and Biological Systems*. Springer Netherlands, 2011.

[3] "Basic electronic components." `http://www.marinetech.org/files/marine/files/Curriculum/TriggerFish/Electrical/Components%20updated2.pdf`. Accessed : Mar 2016.

[4] "Fundamentals of electronic circuit design." `http://www-mdp.eng.cam.ac.uk/web/library/enginfo/electrical/hong1.pdf`. Accessed : Mar 2016.

[5] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*. Oxford University Press, Inc. New York, NY, USA ©2007, fifth ed., 2004.

[6] "Computational nanoelectronics and emerging devices group - nano-fets." `https://www.ece.nus.edu.sg/stfpage/elelg/research.html`. Accessed : Mar 2016.

[7] L. W. Nagel and D. Pederson, "Spice (simulation program with integrated circuit emphasis)," Tech. Rep. UCB/ERL M382, EECS Department, University of California, Berkeley, Apr 1973.

[8] L. W. Nagel, *SPICE2: A Computer Program to Simulate Semiconductor Circuits*. PhD thesis, EECS Department, University of California, Berkeley, 1975.

[9] W. T. Weeks, A. J. Jimenez, G. W. Mahoney, D. Mehta, H. Qassemzadeh, and T. R. Scott, "Algorithms for astap–a network-analysis program," *IEEE Transactions on Circuit Theory*, vol. 20, no. 6, pp. 628–634, 1973.

[10] C. R. Kime, "Getting started with hspice:a tutorial." `http://bear.ces.cwru.edu/eecs_cad/tut_hspice_xor.pdf`, 1998. Accessed : Mar 2016.

[11] C. Visweswariah, "Ibm research report, electrical and timing simulation," tech. rep., EECS Department, University of California, Berkeley, 1998.

[12] "Introducton to vlsi systems." `http://emicroelectronics.free.fr/onlineCourses/VLSI/ch01.html`. Accessed : Mar 2016.

[13] "Introduction to asics." `http://www.ni2designs.com/downloads/docs/Intro_to_ASICs.pdf`. Accessed : Mar 2016.

[14] B. Jain and K. Khare, "A comparative study of methodologies to optimize post-layout challenges," *International Journal of Computer Applications*, vol. 124, no. 8, pp. 27–30, 2015.

[15] "Timing tutorial." `http://www.wright.edu/~tdoom/courses/CEG360/review/Timing_Tutorial.pdf`. Accessed : Mar 2016.

[16] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. USA: Addison-Wesley Publishing Company, 4th ed., 2010.

[17] "Delay calculations." `http://home.anadolu.edu.tr/~mmatanak/lec5Delay.pdf`. Accessed : Feb 2016.

[18] B. Scheff and S. Young, *Gate-level Logic Simulation*. Prentice-Hall, 1972.

[19] J. Jiang, M. Liang, L. Wang, and Y. Zhou, "An effective timing characterization method for an accuracy-proved vlsi standard cell library," *Journal of Semiconductors*, vol. 35, no. 2, p. 025005, 2014.

[20] "Comparing nldm and ccs delay models." `http://www.paripath.com/blog/characterization-blog/comparing-nldm-and-ccs-delay-models`. Accessed : Feb 2016.

[21] "Ccs timing : Technical white paper." `https://www.opensourceliberty.org/ccspaper/ccs_timing_wp.pdf`, Synopsys inc. Accessed : Mar 2016.

[22] "Liberty : Ccs, ecsm or ndlm." `http://chitlesh.ch/wordpress/liberty-ccs-ecsm-or-ndlm`. Accessed : Mar 2016.

[23] R. E. Bryant, "Mossim: A switch-level simulator for mos lsi," *18th Conference on Design Automation*, pp. 786–790, 1981.

[24] R. Kao, "Piecewise linear models for switch-level simulation," tech. rep., Western Research Laboratory, 1992. Accessed : Mar 2016.

[25] L. Pillage, R. Rohrer, and C. Visweswariah, *Electronic Circuit & System Simulation Methods*. McGraw-Hill, Inc. New York, NY, USA, 1999.

[26] A. R. N. Young Hwan Kim, Seung Ho Hwang, "Electrical-logic simulation and its applications," *IEEE Transactions on Computer-Aided Design of ICs and Systems*, vol. 8, pp. 8 – 22, January 1989.

[27] A. Devgan and R. A. Rohrer, "Adaptively controlled explicit simulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, 1994.

[28] K. A. Sakallah, W. Stephen, F. Ellow, and I. Eee, "Samson2 : An event driven vlsi circuit simulator," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. C, no. 4, 1985.

[29] "Effective current source model." `http://www.cadence.com/Alliances/languages/Pages/ecsm.aspx`, cadence design systems, inc.

[30] J. F. Croix, D. F. Wong, and J. Croix, "Blade and razor : Cell and interconnect delay analysis using current-based models," *Design Automation Conference*, pp. 386–389, 2003.

[31] K. Tseng and N. Verghese, "A robust cell-level crosstalk delay change analysis," *Computer Aided Design*, pp. 147–154, 2004.

[32] C. Amin, C. Kashyap, N. Menezes, K. Killpack, and E. Chiprout, "A multi-port current source model for multiple-input switching effects in cmos library cells," *2006 43rd ACM/IEEE Design Automation Conference*, pp. 247–252, 2006.

[33] H. Fatemi, "Statistical logic cell delay analysis using a current-based model," *Design Automation Conference*, pp. 253–256, 2006.

[34] S. Nazarian, H. Fatemi, and M. Pedram, "Accurate timing and noise analysis of combinational and sequential logic cells using current source modeling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 1, pp. 92–103, 2011.

[35] M. Mahmoud, A. Wassal, A. El-rouby, and R. Guindi, "Current source based standard-cell model for accurate timing analysis of combinational logic," *Electronics, Circuits, and Systems (ICECS)*, pp. 281–284, 2013.

[36] K. Chen and Y. Hwan Kim, "Current source model of combinational logic gates for accurate gate-level circuit analysis and timing analysis," *VLSI Design, Automation and Test (VLSI-DAT)*, pp. 1–4, 2015.

[37] "The wire." `http://bwrcs.eecs.berkeley.edu/Classes/icdesign/ee141_f01/Notes/chapter4.pdf`, September 1999. Accessed : Mar 2016.

[38] L.-C. Chen, S. K. Gupta, and M. A. Breuer, "A new gate delay model for simultaneous switching and its applications," in *Design Automation Conference*, pp. 289–294, ACM, 2001.

[39] "Nangate freepdk45 generic open cell library." `http://si2.org/openeda.si2.org/projects/nangatelib/,Aug-2009`. Accessed : Mar 2016.

[40] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Pratical Approach*. 2009.

[41] A. Goel and S. Vrudhula, "Current source based standard cell model for accurate signal integrity and timing analysis," *Design, Automation and Test in Europe*, pp. 574–579, 2008.

[42] P. Nenzi and H. Vogt, *Ngspice Users Manual*, February 2014.

[43] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," *Quality Electronic Design, 2006. ISQED '06. 7th International Symposium on*, pp. 7–12, 2006.

[44] S. Kang, S. Chun, Y. Kim, and G. Kim, "Method of efficiently compressing and decompressing test data using input reduction," October 2005. US Patent App.

[45] Synopsys, *HSPICE ® User Guide : Simulation and Analysis*, 2008.