1

of/de

1

PM-1 3½"X4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

PIONEERS IN METHYLENE BLUE TESTING SINCE 1974

MicroD

1569 COUNTRY ROAD 8, BARRINGTON, NEW HAMPSHIRE 03006
TEL. 603 438 7887 FAX 603 438 7887 TLE 603888488

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

A Timing Macro Model for Performance Optimization of CMOS Logic Circuits

by

Roger Chi Fai Shum, B. Eng.

**A thesis submitted to the Faculty of Graduate
Studies and Research in partial fulfillment
of the requirements for the degree of**

Master of Engineering

**Ottawa -Carleton Institute for Electrical Engineering
Department of Electronics
Faculty of Engineering
Carleton University
Ottawa, Canada
February 1992**

© 1992, Roger C. F. Shum

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-75959-3

Canada

**The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis**

**A Timing Macro Model for Performance Optimization of CMOS
Logic Circuits**

**submitted by Roger Chi Fai Shum, B. Eng.
in partial fulfillment of the requirements for
the degree of Master of Engineering**

**Prof. M. C. Lefebvre
Thesis Supervisor**

**Prof. J.S. Wight
Chairman, Department of Electronics**

**Carleton University
February, 1992**

Abstract

Propagation delay is one of the crucial design parameters of high performance VLSI circuits. The purpose of a timing model is to estimate the delay accurately in a fraction of time taken by circuit simulation. Timing models can be used in performance optimization and timing verification. This thesis discusses various aspects concerning the development of timing model, as well as the design and implementation of a new timing model based on the macro-modeling paradigm. The proposed timing model targets simple logic gates in CMOS technology. Two new algorithms are developed to systematically simplify the model equations to a user specified level of accuracy. Three waveform representations - the ramp function, exponential and piecewise linear - are examined. The ramp function representation is found to be the most suitable for the proposed model. A comparison between the proposed model and SPICE is also presented.

Acknowledgments

I would like to thank his supervisor, Prof. Martin C. Lefebvre, for his invaluable advice, guidance and support. His effort in reviewing this thesis is much appreciated.

I am very grateful to the faculty, staff and graduate students of Department of Electronics for providing a superb environment in which to pursue scientific research. I also appreciate the patience of my fellow SUN network users, who endured the extensive CPU consumption required to complete this work.

The author would like to acknowledge the financial support received from: the Natural Sciences and Engineering Research Council of Canada, Bell-Northern Research Ltd., Ottawa, and the Department of Electronics, Carleton University. The SUN SPARCstations used for the prototype development and experimentation were graciously provided by the Canadian Microelectronics Corporation.

Finally, I would like to thank my parents for their support. Without their encouragement and help, this work would not have been possible.

Table of Contents

Abstract	iii
Acknowledgments	iv
Table of Contents	v
List of Figures	vii
List of Tables	ix
Chapter 1 Introduction	1
1.1 Perspective.....	1
1.2 Objective.....	3
1.3 Outline of the Thesis.....	3
Chapter 2 Related Work	5
2.1 Nodal Analysis Approach.....	5
2.2 The RC model.....	7
2.3 The Analytical Approach.....	11
2.4 The Macromodeling Approach.....	14
2.4.1 MOGLO	14
2.4.2 Matsen's Model.....	18
2.5 Summary.....	23
Chapter 3 Macromodeling Problem Space Definition	24
3.1 Abstraction Of Gate Topology.....	24
3.2 Waveform Representation.....	26
3.2.1 Step Function Representation.....	26
3.2.2 Ramp Function Representation	26
3.2.3 Piecewise Linear Waveform Representation.....	29
3.2.4 Piecewise Non-linear Waveform Representation.....	31
3.3 Loading.....	33
3.4 Input Combination	34
3.5 Summary.....	35
Chapter 4 Overview of the New Proposed Model	36
4.1 Definition of Model Parameters.....	36
4.1.1 Gate Topology	36
4.1.2 Waveform Representation	36
4.1.3 Loading.....	37
4.1.4 Input Vector.....	37
4.2 Development of Model Equations	37
4.2.1 General Form	37
4.2.2 Computation of Coefficient Values.....	38
4.2.3 Model Accuracy	39

4.2.4	State-Space Graph Formulation	40
4.2.5	Algorithm 1	41
4.2.6	Algorithm 2	43
4.3	Summary	46
Chapter 5	Simulation Result	48
5.1	Experimental Framework	48
5.2	Summary of Simulation Results	51
5.2.1	Effect of Independent Variables	51
5.2.2	Effect of Input Vectors	53
5.3	Summary	59
Chapter 6	Macromodeling Results	60
6.1	System Overview	60
6.2	Comparison of the Two Simplification Algorithms	62
6.3	Comparison with SPICE	63
6.3.1	Configuration of Testing System	63
6.3.2	Results	64
6.4	Summary	67
Chapter 7	Conclusions	68
7.1	Summary	68
7.2	Contributions	69
7.3	Future Work	70
Appendix A	Curve Fitting	71
Appendix B	Test Circuits	75
Reference		78

List of Figures

Figure 1.1	A cell in macromodeling and its environment.....	2
Figure 2.1	The RC model for an Inverter.	8
Figure 2.2	Input and Output Waveform for (a) a fast input; (b) a slow input.	12
Figure 2.3	The Switching Time of a Rising Output.	16
Figure 2.4	A Sample Waveform in the Matson's Model.....	19
Figure 2.5	Matson's Model of a General Gate.	20
Figure 3.1	Step Function as Waveform Representation.	27
Figure 3.2	Ramp function representation using two points from the waveform.....	28
Figure 3.3	Two-segment piecewise linear representation of a waveform purposed by Deng[6].	29
Figure 3.4	Two-segment piecewise linear waveform representation using threshold voltages as break points.	30
Figure 3.5	Waveform representation using a ramp and an exponential tail	31
Figure 3.6	Piecewise non-linear waveform representation for a rising output.	32
Figure 3.7	Components of the Loading.....	33
Figure 3.8	Input condition for case 1 for (a) 3-input NAND gate and (b) 3-input NOR gate.	35
Figure 3.9	Input condition for case 2 for (a) 3-input NAND gate (b) and 3-input NOR gate.	35
Figure 4.1	State-space graph to illustrate the problem of removing terms.	41
Figure 4.2	A typical path of nodes that may be travelled by Algorithm 1.....	44
Figure 4.3	A typical path of nodes that may be travelled by Algorithm 2.....	46
Figure 5.1	Block Diagram showing the configuration of the system.....	48

Figure 5.2	Assumed Layout for the transistors.	49
Figure 5.3	Circuits used to compute (a) the min. rise/fall time and (b) the max. rise/fall time.	50
Figure 5.4	The effect of W_n , CL and Tr on the delay of an inverter:	52
Figure 5.5	The effect of W_n , CL and Tr on the output fall time of an inverter:	54
Figure 5.6	The behavior of delay of NAND and NOR gates:.....	55
Figure 5.7	The output of a 4-input NAND gate with a fast and a slow input.	56
Figure 5.8	Voltage waveforms for the 4-input NAND gate using: (a) input vector (1), (b) input vector (2), (c) input vector (3).	57
Figure 5.9	Comparison between the propagation delay with the three input vectors: (a) input vector (1) and input vector (2); (b) input vector (2) and input vector (3); (c) input vector (1) and input vector (3).	58
Figure 6.1	Overview of the system for generating the model equations.....	60
Figure 6.2	Configuration of the program for testing the models.....	64
Figure 6.3	Output waveform and two of its representation for the circuit in Ex3.	66

List of Tables

Table 6.1	Comparison between Algorithm 1 and Algorithm 2 for the ramp function waveform representation.....	62
Table 6.2	Comparison of the complexity of the three waveform representations.	65
Table 6.3	Experimental Results	65
Table 6.4	Comparison of the accuracy of the three waveform representations with SPICE.....	67

Chapter 1

Introduction

1.1 Perspective

An important problem in VLSI CAD tool research is the development of accurate methods for estimating the analog behavior of digital logic gates, or timing models. A timing model can be used for both timing verification and performance optimization. It has to be sufficiently accurate to ensure that the actual circuit behaviour is not too far from its expected behaviour. It must also be computationally efficient in order to be feasible for the analysis and optimization of large VLSI circuits.

Timing models have been the subject of many research efforts. Existing timing models can be divided into four categories: (i) nodal analysis approaches, (ii) RC models, (iii) analytical approaches and (iv) macromodeling approaches. The nodal analysis approaches formulate a set of equations by performing nodal analysis on the circuit. This set of equations is solved to obtain the timing performance of the circuit. The RC timing models consider the MOS transistor as a perfect switch in series with a resistor and a capacitor connected to ground. RC models compute timing performances by solving the circuit as an RC circuit. Analytical approaches use the device equations of the MOS transistor to derive equations for the dependent variables of the timing model.

This thesis presents a new timing model, based on the macromodeling paradigm. For the purpose of macromodeling, a digital cell can be viewed as a black box, as shown in Figure 1.1. It is assumed that each cell only interacts with its immediate neighbors and

that this interaction is unidirectional. It can therefore be characterized mathematically as boundary conditions.

Figure 1.1 A cell in macromodeling and its environment.

C_L , the load capacitance and $V_{in}(t)$, the input waveform of the switching input(s) and S_{in} , the state(s) of the stable input(s) are the boundary conditions imposed on the cell by its environment. C_L consists of the parasitic wiring capacitances and the input capacitance of the fan-out cells. $V_{in}(t)$ depends on the drive capability of the logic cell driving $V_{in}(t)$ and the input capacitance of the cell itself. C_{IN} , the input capacitance and $V_{out}(t)$, the output waveform are two interface parameters generated by the cell which may affect other cells. T_D , the propagation delay is an internal attribute which cannot affect other cells.

T_D , $V_{out}(t)$ and C_{IN} are the dependent variables of the model. There are two types of independent variables: (i) the boundary conditions, $V_{in}(t)$ and C_L , and (ii) internal parameters. Internal parameters include the sizes of the transistors, the parasitic capacitance at each node, and the topology of the cell.

The aim of macromodeling is to develop a set of equations to express the dependent variables in terms of the independent variables. First, a general form must be developed for these equations. Then simulations are performed on each cell to obtain timing data that spans the complete problem space. These data are used to compute the coefficient values in the model equations.

1.2 Objective

The objective of this thesis is two-fold. The first objective is to provide an insight into the problems associated with the development of a timing model. Various factors that can affect the timing performance of a circuit are examined. These factors include the gate topology, the loading of the circuit, the sizes of the transistors, the input combination, waveform representation and the definition of propagation delay. The second objective is to develop a new timing model suitable for optimization and timing verification. This means that the model should have a short computation time and yet the accuracy should remain reasonably high.

1.3 Outline of the Thesis

This thesis is organized as follows. Chapter 2 reviews existing approaches used in the development of timing models, and addresses advantages and disadvantages of each approach. Chapter 3 presents a definition of the problem space associated with macromodeling. Chapter 4 provides an overview of a new proposed timing macromodel. It describes the various theoretical aspects of the new model, the options chosen and the assumptions that were made. Chapter 5 discusses the practical issues concerning the collection of experimental timing data: methods to carry out the simulation and related software. An examination of the behavior of CMOS logic gates is also presented in this

chapter. Chapter 6 describes prototype implementation and experimental results, followed by conclusions and recommendations for future work in Chapter 7.

Chapter 2

Related Work

This chapter presents a subjective review of selected timing models from the four categories: nodal analysis, RC model, analytical approach and macromodeling approach.

2.1 Nodal Analysis Approach

Nodal analysis methods use the Kirchhoff's current law to solve for the voltage at each node in the circuit. The basic equation to be solved in matrix form is

$$Y\hat{V} = \hat{J} \quad (2.1)$$

where Y is the conductance matrix, $\hat{V}$ is the vector of the node voltage and $\hat{J}$ is the vector of current. SPICE is by far the most popular circuit level simulator which uses this approach. SPICE solves a set of differential equations, as in equation (2.2), using modified nodal analysis, a numerical integration method and other matrix computation techniques to obtain the time domain solution of a circuit. If the circuit contains nonlinear circuit elements, like MOS transistors, the Newton-Raphson iteration algorithm has to be employed to compute the solution.

$$C\frac{d}{dt}\hat{V} + Y\hat{V} = \hat{J} \quad (2.2)$$

The time step of the analysis can be varied to keep a balance between efficiency and accuracy. An algorithm for controlling the step size is used to improve the computation efficiency. The accuracy of the solution at each time point is tested by estimating the truncation error. If the solution is not accurate enough, this solution is

discarded and the calculation is repeated with a reduced step size. SPICE is well known for its accuracy and also its huge consumption of computation time.

Another timing model proposed by Lin[15] also uses nodal analysis. This timing model is very similar to SPICE, except that it makes some simplifications. It approximates the non-linear functions describing the conductance of a MOS transistor by a stepwise linear function. So the conductance of a MOS transistor is a constant within a time step. At the beginning of a time step, the model uses the following equations to calculate the conductance of the MOS transistor.

$$\begin{aligned}
 G(t) &= \beta \cdot (2 \cdot (V_{GS} - V_{th}) - V_{DS}) \cdot \frac{V_{GS} - V_{th}}{V_{DS}}, & \text{if } V_{GS} - V_{th} \geq V_{DS} \geq 0 \\
 G(t) &= \beta \frac{(V_{GS} - V_{th})^2}{V_{DS}}, & \text{if } V_{DS} \geq V_{GS} - V_{th} \geq 0 \\
 G(t) &= 0 & \text{if } V_{GS} < V_{th}
 \end{aligned} \tag{2.3}$$

where $\beta = (k/2) (w/l)$ and k is the transconductance parameter, w and l are the width and effective length of the MOS transistor, respectively. V_{th} is the threshold voltage. Because of this simplification, many parasitic effects of the MOS transistors are ignored.

Because the conductance is assumed to remain constant over each time step, equation (2.2) is a set of linear differential equations. Hence, the Newton iteration method can be avoided. The model makes another simplification by neglecting the 'OFF' MOS transistors. For example, the model neglects the pull-up network and lumps all the ground capacitors in the pull-up network to the load capacitor when determining the falling transition of the output. In this way, the short circuit current for the short period of time in which both the pull-up and pull-down networks are 'ON' is also neglected. To account for this effect, the model magnifies the output load capacitance by 1.5 times when the output

voltage is still above 4.5V for a falling transition and when the output voltage is still below 0.5V for a rising transition. These simplifications decrease the size of the matrices in equation (2.2) and speed up the computation, at the expense of reduced accuracy.

2.2 The RC model

Whereas nodal analysis methods dominate for timing verification applications, the RC model is the most popular one in the circuit optimization context, due to its computational simplicity. Numerous efforts have been devoted to this switch-level model. In the RC model, each transistor is modeled as a perfect switch in series with a linear resistor plus linear capacitors to ground. The circuit operates by opening and closing the switches to establish electrical path(s) from the output node to the power rails (V_{DD} or ground). The time constant of the RC circuit is taken as the delay of the MOS circuit.

Figure 2.1 illustrates the RC model for an inverter. Since only one of the switch is 'ON' at a time, the inverter is reduced to an RC circuit. Hence, the delay for the inverter is

$$\begin{aligned} d_{HL} &= R_n C_{total} \\ d_{LH} &= R_p C_{total} \end{aligned} \quad (2.4)$$

where d_{HL} and d_{LH} are the delay for the falling and the rising output transients respectively, R_n and R_p are the resistance of the NMOS and the PMOS respectively, C_{total} is the total capacitance at the output node - the sum of the load capacitance C_L and the parasitic capacitances of the two transistors at the output node.

As gates become more complex, the number of transistors inside the gates increases, and hence the resulting RC circuits become more complicated. Furthermore, the equivalent circuit for a gate depends on the value of the inputs. It may become very

Figure 2.1 The RC model for an Inverter.

difficult, sometimes even impossible, to calculate the values of equivalent resistance and capacitance (R_{eq} and C_{eq}) for a complex CMOS circuit.

There are two ways to obtain the resistance and the capacitance for a gate. The first way is to compute R and C from the processing parameters. TILOS[21] uses this method. In this model, the channel resistance is taken as the resistance of the transistor. The channel resistance is equal to the unit sheet resistance divided by the width of the transistor. There are three types of capacitances - drain to gate, gate to source and gate capacitances. The drain to gate capacitance and the source to gate capacitance are considered to be the same. The equations for the capacitances are

$$c_{gd,gs} = CJA \cdot DIFF \ HANG \cdot x + 2 \cdot CJP \cdot (DIFF \ HANG + x) \quad (2.5.a)$$

$$c_g = CGTA \cdot MASKCHANNEL \cdot x \quad (2.5.b)$$

where CJA is the diffusion area capacitance, $DIFF \ HANG$ is the length of diffusion extends beyond the active area, CJP is the diffusion perimeter capacitance, $CGTA$ is the gate capacitance, $MASKCHANNEL$ is the mask channel length and x is the width. All these variables are either processing related or can be extracted from the layout.

Other methods, such as Crystal [19] and Epoxy [18], make use of SPICE simulation to determine the values of R and C . Crystal[19] considers a transistor as an RC circuit. It computes the resistance by dividing the SPICE-simulated delay value by the load capacitance. It uses different resistance values depending on whether the transistor is 'ON' or 'OFF'. Epoxy, on the other hand, uses the following set of equations to model a transistor:

$$\begin{aligned} R_{typefet} &= K_{rtype} \left[\frac{l_{fet}}{w_{fet}} \right] \\ C_{gatefet} &= K_{gtype} w_{fet} l_{fet} \\ C_{sdfet} &= K_{sdtype} w_{fet} \end{aligned} \quad (2.6)$$

The three constants, K_{rtype} , K_{gtype} and K_{sdtype} are determined by a least square fit to SPICE simulations.

Three ways are proposed by Crystal[19] to estimate the delay from the complicated RC circuit. The first one is the lumped RC model. In this model, the equivalent resistance and capacitance are computed by lumping all the resistances and the capacitances. Hence, the delay is:

$$d = [\sum R] [\sum C]. \quad (2.7)$$

Only those capacitances between the triggering transistor and the output node are included in the summation. Other capacitances are assumed to be completely charged or discharged. This model is the crudest one among the three models, since it assumes that all the capacitances must be discharged through all the resistances. Furthermore, the input rise/fall time is not taken into account.

The second model is the lumped slope model. This model takes the input rise/fall time into account by introducing a new parameter - the rise time ratio. This model defines the intrinsic rise time of a gate to be the rise time of the output when the input is a step function. The rise time ratio for a gate is equal to the rise/fall time of the input divided by its intrinsic rise time. Each rise time ratio has its effective resistance for a transistor determined from SPICE simulations. Therefore, the table of effective resistance becomes function of the types of the transistors, the value the transistor is transmitting(0 or 1), and the rise time ratio. During the analysis, the effective resistance of the triggering transistor is computed by interpolating the appropriate values in the table. All the other transistors are assumed to be either fully 'ON' or fully 'OFF'. Then, the delay is calculated using equation (2.7).

The major problem with this model is that the effect of the rise/fall time of an input cannot propagate through different stages of gates to the output. This is because the rise/fall time of the output of a gate cannot be evaluated by the RC model. Crystal[19] uses the intrinsic rise time of a gate as the input rise/fall time of the following gate for all cases. This could create a significant error for a long chain of gates.

The last model is the distributed slope model. This model is similar to the lumped slope model but computes the delay in a slightly different way. Instead of using equation (2.7), the following equation is used.

$$d = \sum_i R_i C_i \quad (2.8)$$

where R_i is the sum of the resistances from the power source (V_{DD} or ground) to node i in the RC circuit and C_i is the total capacitance at node i . Hence, each capacitance is only weighted by the resistance between it and the signal source.

Of the three models found in Crystal, the last one is the most accurate. Equation (2.8) is used by many other researchers, such as TILOS[21], except for the use of the rise time ratio to calculate the effective resistance of the triggering transistor.

2.3 The Analytical Approach

A third delay modeling approach consists of developing equations for the delay and the rise/fall time of a logic gate from the basic I_D versus V_{DS} equations for the MOS transistors. Hedenstierna and Jeppson[12] derive the output equations for a long channel inverter driven by a ramp input by using the device equations of the NMOS and the PMOS transistors. They also specify a criterion to distinguish between fast and slow inputs. An input is said to be fast if it reaches its final value before the OFF transistor leaves the saturation region of the transient. Otherwise, the input is slow. The OFF transistor is the one which is OFF at the beginning, travels through the saturation region and then settles in the linear region at the end. Figure 2.2 illustrates this definition. V_1 is the value of the output voltage at which the OFF transistor leaves the saturation region. V_2 is the value of the output voltage at which the input reaches its final value. In this case, the OFF transistor is the NMOS. Hedenstierna and Jeppson use the following equation for the input:

$$V_{in}(t) = \begin{cases} 0 & t < 0 \\ t/\tau & 0 \leq t < \tau V_{DD} \\ V_{DD} & \tau V_{DD} \leq t \end{cases} \quad (2.9)$$

Hence the criterion for the fast input can be stated as in equation (2.10):

$$\frac{1}{\tau} > \frac{(V_{DD} - V_{Tn})^3}{6V_{Tn}} \cdot \frac{k_N}{C_L} \quad (2.10)$$

where C_L is the load capacitance, $k_N = \beta \frac{W}{L}$ is the gain factor for the NMOS, and τ is the input rise time.

Figure 2.2 Input and Output Waveform for (a) a fast input; (b) a slow input.

For the falling output with fast input, they divide the output into three regions: $V_o \geq V_2$, $V_1 \leq V_o \leq V_2$, and $V_o \leq V_1$. They then derive expressions for V_o as a function of time regions. These equations are found to be

$$V_o = V_{DD} - \frac{\tau k_N}{6C_L} \left(\frac{t}{\tau} - V_{Tn} \right)^3 \quad \text{for } V_o \geq V_2 \quad (2.11.a)$$

$$V_o = V_2 - \frac{k_N}{2C_L} (V_{DD} - V_{Tn})^2 (t - \tau V_{DD}) \quad \text{for } V_2 \geq V_o \geq V_1 \quad (2.11.b)$$

$$t - t_{sat} = \frac{C_L}{k_N (V_{DD} - V_{Tn})} \ln \left(\frac{2(V_{DD} - V_{Tn}) - V_o}{V_o} \right) \quad \text{for } V_1 \geq V_o \quad (2.11.c)$$

Hedenstierna and Jeppson use equation (2.11.c) to solve for $t_{1/2}$ which is in the third region ($V_o \leq V_1$). Equation (2.11.b) is used to solve for t_{sat} , which is the time at which the 'OFF' transistor leaves the saturation region. Equation (2.11.a) is used to solve for V_2 .

They find that the propagation delay for an inverter with a fast ramp input is:

$$\begin{aligned}
 t_{dHL} &= t_{1/2} - \frac{\tau}{2} V_{DD} \\
 &= t_{dHL,step} + \frac{\tau}{6} (V_{DD} + 2V_{Tn})
 \end{aligned} \quad (2.12)$$

where τ is the rise time of the input and $t_{dHL,step}$ is the delay for the step input, which is equal to:

$$t_{dHL,step} = \frac{C_L/k_N}{(V_{DD} - V_{Tn})} \left[\frac{2V_{Tn}}{V_{DD} - V_{Tn}} + \ln\left(\frac{2(V_{DD} - V_{Tn}) - V_{DD}/2}{V_{DD}/2}\right) \right] \quad (2.13)$$

Hence, the delay for fast ramp input is the sum of the step response delay and a value dependant only on the input rise time. Equations (2.10) to (2.13) are for the falling output. A similar set of equations can be derived for the rising output.

Hedenstierna and Jeppson repeat the analysis for the "slow" input. Again the output is divided into three regions: $V_o \geq V_1$, $V_2 \leq V_o \leq V_1$, and $V_o \leq V_2$. For the first region ($V_o \geq V_1$), equation (2.11.a) is still valid and is used to compute V_1 . For the second region ($V_2 \leq V_o \leq V_1$), the equation of V_o as a function of input rise/fall time is found to be:

$$\frac{V_{DD}}{V_o} = \left[\exp\left(\frac{\tau k_N}{2C_L} \left(\frac{t}{\tau} - V_{Tn}\right)^2\right) \right] \left[\left(\frac{V_1}{V_{DD}} \exp\left(\frac{\tau k_N V_1^2}{2C_L}\right) \right)^{-1} + \sqrt{\frac{\pi \tau k_N V_{DD}^2}{8C_L}} \left(\operatorname{erf}\left(\sqrt{\frac{\tau k_N V_1^2}{2C_L}}\right) - \operatorname{erf}\left(\frac{\tau k_N (t/\tau - V_{Tn})^2}{2C_L}\right) \right) \right] \quad (2.14)$$

where $\exp(x)$ is the exponential function and $\operatorname{erf}(x)$ is the error function. The equation for V_o for the third region is found to be:

$$t - \tau V_{DD} = \frac{C_L}{k_N(V_{DD} - V_{Tn})} \left[\ln\left(\frac{2(V_{DD} - V_{Tn}) - V_o}{V_o}\right) - \ln\left(\frac{2(V_{DD} - V_{Tn}) - V_2}{V_2}\right) \right] \quad (2.15)$$

Since $t_{1/2}$ can be either in the second region or in the third region, V_2 must be computed first by setting $V_{in} = t/\tau = V_{DD}$ in equation (2.14). If V_2 is smaller than $V_{DD}/2$, equation (2.14) is used to calculate $t_{1/2}$. Otherwise, equation (2.15) is used, instead. In this case, an iterative approach must be used since a direct solution does not exist.

This exercise on the part of Hedenstierna and Jeppson demonstrates the enormous complexity of an analytical approach for the simplest logic gate: the inverter. It leaves very little hope of ever finding a feasible solution for general gates.

The second limitation is accuracy. Only the 'OFF' transistor is considered in the analysis. Effects, like the short circuit current and parasitic capacitances, are ignored.

2.4 The Macromodeling Approach

This section presents two existing macromodeling approaches found in the literature.

2.4.1 MOGLO

MOGLO[14], developed by Hoppe *et al.*, is a timing macromodel for inverter-like CMOS logic gates. In this model, the ratio of the width of the PMOS to the width of the NMOS (W_p/W_n) is fixed and is adjusted to a value that makes the delay of the rising and the falling output the same. This reduces the number of model parameters representing the transistor sizes to one.

One of the observations by Hedenstierna and Jeppson[12] is used by Hoppe *et al.* For a fast input, the delay is the sum of the delay of a step input and a value which depends only on the input rise/fall time. Equation (2.12) is re-stated as:

$$t_d = t_{d,step} + t_d^{in} \quad (2.16)$$

where $t_{d,step}$ is the step response delay, which is equal to equation (2.13) for the falling output, and t_d^{in} is the input dependent delay. According to equation (2.13), the step response delay is proportional to C_L/k_N ($t_{d,step} \cong C_L/k_N$). Hence

$$t_{d,step} \approx C_L \frac{W_p}{L} \approx C_L / W_{n(p)} \quad (2.17)$$

Since W_p/W_n is fixed, equation (2.17) can be rewritten as

$$t_{d,step} - C_L / W_{n(p)} - C_L / W = \frac{1}{G^*} (C_L / W) \quad (2.18)$$

where $W = W_p + W_n = W_n (W_p/W_n + 1)$.

t_d^* in equation (2.16) is given by

$$t_d^* = \frac{t_{r/f}}{6} (1 + 2 \frac{V_m}{V_{DD}}) \quad (2.19)$$

which is the same as $t_{dHL,step}$ in equation (2.12) with τV_{DD} replaced by $t_{r/f}$.

The proportionality constant, G^* , in equation (2.18) is considered as a specific transconductance, which can be used as a free parameter. To account for the parasitic capacitance, another free parameter, $C_{s/d}^*$, is introduced for the specific source/drain capacitance. To increase the accuracy of the model, the constant term, $1/6$, in equation (2.19) is replaced by another free parameter, Γ . The final equation for the delay of an inverter for both rising and falling is

$$t_d = (C_L + W C_{s/d}^*) / (G^* W) + \Gamma t_{r/f} (1 + 2 V_T / V_{DD}) \quad (2.20)$$

where V_T is the absolute value of the threshold voltage of the transistors.

In this model, the rise/fall time of the output is defined as the slope of the output voltage, $V_o(t)$, at the mid-level voltage. Figure 2.3 illustrates the definition of the switching time for a rising output graphically. In Figure 2.3, line AB touches $V_o(t)$ at $V_{DD}/2$ and t_{sw} is the rise time. Mathematically, t_{sw} can be expressed as:

$$\begin{aligned}
 t_{rw} &= V_{DD} \cdot \left\{ \left(\frac{dV_o}{dt} \right)^{-1} \right\} \Big|_{V_{DD}/2} \\
 &= V_{DD} \cdot \left(\frac{dt}{dV_o} \right) \Big|_{V_{DD}/2}
 \end{aligned} \tag{2.21}$$

Figure 2.3 The Switching Time of a Rising Output.

The derivative $\frac{dt}{dV_o}$ is obtained from equation (2.11.c):

$$\frac{d}{dV_o} t \Big|_{V_{DD}/2} = \left[\frac{C_L/k_N}{(V_{DD} - V_T)} \right] \left[\frac{8}{3} \right] \left[1 + \frac{1}{3} \frac{V_T}{V_{DD}} \right] \tag{2.22}$$

To relate t_{sw} with $t_{d,step}$, equation (2.13) is rewritten as:

$$t_{d,step} = \left[\frac{C_L/k_N}{(V_{DD} - V_T)} \right] \left[\ln(3) + \frac{2}{3} \frac{V_T}{V_{DD}} \right] \tag{2.23}$$

In the process of deriving the above two equations, approximations are made to simplify the expression. (V_T/V_{DD}) is considered to be small. Hence, terms with $(V_T/V_{DD})^2$ and higher powers of (V_T/V_{DD}) are neglected. By eliminating k_N in the above two equations, the final equation for t_{sw} is obtained as

$$t_{sw} = \frac{8}{3 \ln(3)} \left(1 - 0.27 \frac{V_T}{V_{DD}}\right) t_{d,step} \quad (2.24)$$

According to Hoppe, the maximum delay for any gate occurs when all series transistors are switching simultaneously and when only one of the parallel transistors is switching. Furthermore, a few 'ON' transistors in series can be regarded as one 'ON' transistor with reduced width because an MOS transistor can be modeled as a switched resistor in the lowest level. Hence, the multi-input NAND or NOR gates are considered as an inverter with reduced width for the worst case input. Consequently, equation (2.20) can be applied to m-input NAND gates and m-input NOR gates with different sets of parameters. As for the inverter, the delay for the rising and falling output must be the same. Therefore, all PMOS transistors and all NMOS transistors must have their own uniform width. Moreover, the ratio W_p/W_n must be fixed to make the rising and the falling transients symmetrical. Hence the delay equation for the NAND gates and the NOR gates with the worst case input is given by

$$t_d = (C_L + WC^* \gamma_{ps}^p) / (G^* \gamma_{ps}^p W) + \Gamma \gamma_{ps}^p t_{uf} (1 + 2V_T/V_{DD}) \quad (2.25)$$

Similarly, equation (2.24), which describes the rise/fall times of an inverter, can be applied to the NAND gates and the NOR gates with the worst case input. The type of gate affects t_{sw} through the value of $t_{d,step}$, which is

$$t_{d,step} = (C_L + WC^* \gamma_{ps}^p) / (G^* \gamma_{ps}^p W) \quad (2.26)$$

C_L represents the sum of two sources of load capacitance. They are the wiring capacitance and the input capacitance of fanout gates. The constant capacitance, C^* , is used to denote the wiring capacitance which is estimated from the layout. Another parameter $\cdot C^* \gamma_{ps}^p$, the specific input capacitance, is developed to compute the input capacitance of a fanout gate. $C^* \gamma_{ps}^p$ for a specific gate is obtained by matching the SPICE

output waveforms of a symmetric inverter driving that gate, and variable capacities. Therefore, the Miller capacitance is taken into account. Hence C_L for a gate is equal to

$$C_L = C^W + cC_{i,driven}^{type} W_{driven} \quad (2.27)$$

where c is the fanout factor of the current gate, $C_{i,driven}^{type}$ is the specific input capacitance of the gates driven by the current gate, and W_{driven} is the total width ($W_n + W_p$) of the driven gates.

One of the disadvantages of this model is the fact that it is for circuits with fast input only because equation (2.16) is only valid for fast input. As the input slows down, the error in using this model increases.

2.4.2 Matson's Model

Matson's model is another timing model using macromodeling. This is a complex model developed for NMOS circuits. The waveshape of a signal in Matson's model is represented by two parts - the portion of time shift (T_{BE}) and the portion that the signal switches (T_{SW}). Figure 2.4 shows an example. The switch portion is assumed to be a ramp with exponential tail and T_{SW} is the time constant.

A detailed analysis on the inverter is done by Matson. The behavior of the pulldown transistor during the transition period is examined. The pulldown transistor behaves like an amplifier when it is saturated and like a resistor when it is in its linear operation region. So, for fast input, it is essentially a resistor during the input switch. For slow input, it is more like an amplifier.

The behaviour of the parasitic capacitors - c_{gs} and c_{gd} - are also examined. The Miller's effect, which have a great effect on both c_{gs} and c_{gd} are also considered in the analysis. It turns out that the input capacitance during the input switches (C_{SWin}) is more

Figure 2.4 A Sample Waveform in the Matson's Model.

than the input capacitance before the input switches (C_{BEin}) if the input rise/fall time is large. This difference will increase and eventually reaches a plateau as the input rise/fall time increases. The difference is caused by the Miller effect.

For more general gates, the model becomes more complicated because of the additional transistors. A closed-form expression cannot be found. Elmore delay[9] is used to approximate the delay of the output waveforms. This effectively approximates the true response as a single time constant exponential. Elmore delay is defined as the moment of the impulse response of a gate:

$$T_D = \int_0^{\infty} t h(t) dt \quad (2.28)$$

where $h(t)$ is the impulse response of a gate.

Moreover, more transistors mean more parasitics capacitance in the circuit. Some of these capacitances must be discharged or charged during the transition period. Hence, they have to be added to the load capacitance. Figure 2.5 is used to model a general gate.

Figure 2.5 Matson's Model of a General Gate.

After the examination of SPICE results, the following set of equations are developed for the model:

1) For the falling output:

i) Delay until output begins to fall:

$$T_{BEFout} = T_{BEF0} + mT_{SWRin} \quad (2.29.a)$$

$$T_{BEF0} = a_1 + b_1 \frac{1}{w_{pd}} C_{total} \quad (2.29.b)$$

$$m = d_1 + d_2 \frac{S_{pu}}{w_{pdtotal}} + d_3 \frac{C_{total}}{w_{pd}} \quad (2.29.c)$$

ii) Switching (fall) time of falling output transition:

$$T_{SWFout} = T_{SWF0} + mT_{SWRin} \quad (2.29.d)$$

$$T_{SWR0} = a_1 + \text{Elmore delay approximation with}$$

$$r_{pd} = \frac{b_1}{w_{pd}} \text{ for each conducting pulldown} \quad (2.29.e)$$

$$m = d_1 + d_2 \frac{S_{pu}}{w_{pdtotal}} + C_{total} \left(\frac{d_3}{w_{pd}} + \frac{d_4}{w_{pdbefore}} \right) \quad (2.29.f)$$

2) For the rising output:

iii) Delay until output begins to rise:

$$T_{BERout} = T_{BER0} + mT_{SWFin} \quad (2.29.g)$$

$$T_{BER0} = a_1 + b_1 \frac{C_{total}}{S_{pu}} \quad (2.29.h)$$

$$m = d_1 + d_2 \frac{S_{pu}}{w_{pdtotal}} + d_3 \frac{C_{total}}{w_{pd}} \quad (2.29.i)$$

iv) Switching (rise) time of rising output transition:

$$T_{SWRout} = \frac{(T_{SWR0} + T_{SWR0}^{asym})^2}{(T_{SWR0} - T_{SWR0}^{asym}) + mT_{SWFin}} + (T_{SWR0}^{asym} + mT_{SWFin}) \quad (2.29.j)$$

$$T_{SWR0} = a_1 + \text{Elmore delay approximation with } r_{pu} = \frac{b_1}{S_{up}} \quad (2.29.k)$$

$$m = d_1 + d_2 \frac{S_{pu}}{w_{pdtotal}} + C_{total} \left(\frac{d_3}{w_{pd}} + \frac{d_4}{w_{pdbefore}} \right) \quad (2.29.l)$$

$$T_{SWR0}^{asym} = e_1 + e_2 T_{SWR0} \quad (2.29.m)$$

v) Input capacitance during the input switches

$$C_{SWRin} = C_{SWR0} + (\Delta C_{SWRin}) \left(\frac{mT_{SWin}}{\Delta C_{SWRin} + mT_{SWin}} \right) \quad (2.29.n)$$

$$\Delta C_{SWRin} = C_{SWR-in} - C_{SWR0in} \quad (2.29.o)$$

For the inverter,

$$C_{SWR0in} = c_{gs} + c_{gd} \cong a_1 w_{pd} \quad (2.29.p)$$

$$C_{SWR0-in} = c_{gs} + 2c_{gd} \cong b_1 w_{pd} \quad (2.29.q)$$

$$m = w_{pd} \left(c_1 + c_2 \frac{g_m}{C_L} \right) \cong w_{pd} \left(c_1 + c_2 \frac{w_{pd}}{C_L} \right) \quad (2.29.r)$$

For the general gate,

$$C_{SWR0in} = \frac{a_1 w_{pd}}{1 + (a_2 w_{pd}) / w_{pdbefore}} \quad (2.29.s)$$

$$C_{SWR-in} = b_1 w_{pd} \quad (2.29.t)$$

$$m = w_{pd} \left[c_1 + \frac{w_{pd}}{C_{total}} \left(c_2 + c_3 \frac{w_{pd}}{w_{pdbefore}} \right) \right] \quad (2.29.u)$$

In the above equations:

$a_1, b_1, c_1, c_2, d_1, d_2, d_3, d_4, e_1$ and e_2 are free parameters which have to be determined;

w_{pd} is the width of the triggering transistor (M_{trig});

$w_{pdtotal}$ is the effective total w_{pd} of the devices in the conducting path (w_{pd} 's are treated as conductances) and the following equations are used to define C_{total} :

$$C_{total} = c_{pdriver} + c_{pdafter} + C_L; \quad (2.29.v)$$

$$c_{pdriver} = c_1 w_{pd}; \quad (2.29.w)$$

$$c_{pdafter} = \sum_{after} c_2 w_{pdr} \quad (2.29.x)$$

Different equations are used for rising and falling output. Hence, there is no need to have a fixed ratio between different type of transistors. Moreover, there is no need for the transistors of the same type to have the same size since the parasitic capacitance are calculated individually. While MOGLO can only calculate the worst case delay, Matson's model can calculate the delay corresponding to each input. This makes the model more powerful but also more complex. The model must either store the electrical path for each

input or compute the path dynamically. Because of the complexity of this model, more SPICE simulations and more complicated procedure for curve fitting are required. The overhead CPU cost for this model increases.

2.5 Summary

This chapter has presented a review of selected timing models from the literature. Four types of models have been discussed: nodal analysis, RC, analytical and macromodeling approaches. Macromodels were emphasized to serve a basis for comparison for the model developed in this thesis.

Chapter 3

Macromodeling Problem Space Definition

There are many different ways of capturing the analog behaviour of logic gates into a set of mathematical equations. Choices exist, in particular, at four different levels. These are: (i) the abstraction of gate topology, i.e. the cell parameters that affect the timing behaviour, (ii) the abstract representation of analog waveforms and hence, the definition of dependent variables, (iii) the representation of output loading conditions, and (iv) the selection of input vectors for which timing information should be sought. This chapter examines these factors and issues as they relate to macromodeling.

In any event the model must yield two important quantities for both rising and falling transients. The first is the propagation delay through the gates and the second is the output waveform's rise/fall times in order to estimate the delay through concatenated gates.

3.1 Abstraction Of Gate Topology

Some timing models impose some restrictions on the topology of the gates. The aim of these restrictions is to limit the number of the variables and the number of equations in the model. There are two commonly used representations for the sizes for the transistors. The first one is individual sizing where each transistor is associated with a unique size variable. Matson's model and most of the RC models use this representation. This representation allows more flexibility for the optimization algorithm at the expense of increased model complexity. The second approach is uniform gate sizing where the transistors in a gate are somehow constrained so that a single size variable can uniquely

define the sizes of all the gate's transistors. For example, some timing models fix the ratio between the width of the NMOS transistor (W_n) of an inverter and the PMOS transistor (W_p). With this fixed ratio, one set of equations can be used for both the rising and falling transients.

NAND and NOR gate can be accommodated in a fashion similar to the inverter with a single gate size parameter. All transistors of the same type have the same size - W_n for all the NMOS transistors and W_p for all the PMOS transistors. The W_p to W_n ratio can either be the same for all logic gates of a given type or it can be adjusted according to the number of transistors in series. Equation (3.1) states the relationship between W_p and W_n if the W_p to W_n ratio are the same for all the gates.

$$W_p = \alpha_{inverter} \cdot W_n \quad \text{for NAND and NOR gates} \quad (3.1)$$

With this option, the rising output will not be symmetrical to the falling output. To keep using one set of equations, only the worst case output is considered.

The W_p to W_n ratio can be used to make the rising output symmetrical to the falling output. Hence, a second option is to set the ratio according to the type and the number of input of the gates. The following equations define the ratios:

$$W_p = \frac{\alpha_{inverter}}{N} \cdot W_n \quad \text{for NAND gate} \quad (3.2)$$

$$W_p = N \cdot \alpha_{inverter} \cdot W_n \quad \text{for NOR gate} \quad (3.3)$$

where $\alpha_{inverter}$ is the ratio for the inverter and N is the number of input to the gate. The drawback of this option is that the layout of the gates have heights that can vary significantly.

A third option is to have the two sizes independent so they can be adjusted individually. Hence, the switching performance can also be optimized, but the model will require two variables for representing the sizes of the transistors.

3.2 Waveform Representation

The selection of a waveform representation is very important to a timing model because it can affect both the number of equations in the model, as well as the number of variables. There are four common waveform representation methods used in various timing models. They are described in this section.

3.2.1 Step Function Representation

The simplest waveform representation is the step function (Figure 3.1). The advantage of this representation is its simplicity. There is no need to have extra equations for evaluating the waveform representation of the output. The number of variables in the model will not be increased either. This is because there is no information on the rise/fall time that can be passed on to the next stage. It is always equal to zero. This is also the major drawback of this representation.

3.2.2 Ramp Function Representation

Another way to represent a waveform is to use a ramp function to approximate the waveform. This is the most common waveform representation of which there are a number of variations.

MOGLO by Hoppe *et al* uses the slope of the waveform at $V_{DD}/2$ as the slope of the ramp. Figure 2.3 shows an example using this method to represent a waveform. With some approximation, the slope is defined as a constant fraction of the step response delay

Figure 3.1 Step Function as Waveform Representation.

in this model. Since the step response delay is also used in the delay calculation, there is no extra parameter produced. Hence, there is no need to have extra effort in simulation and in curve fitting. On the other hand, the slope of a single point in the waveform is usually not a good approximation of the waveform. Moreover, the most basic equations are used and some approximations are used in the derivation of equation (2.24). Under certain circumstances, using this method may result in a significant error. Sakurai[20] suggests the use of 70% of the slope of the output at $V_{DD}/2$ as a better approximation for the input rise/fall time of the next stage.

Another variation for the ramp function representation uses a line formed by joining two points on the actual waveform. For example, Hallam[11] proposes the use of two points at 90% of V_{DD} and 10% of V_{DD} respectively as illustrated in Figure 3.2. Line A is the actual output. Line B is the line used to represent line A. Line B intersects line A at $0.9V_{DD}$ and $0.1V_{DD}$. Once t_1 and t_2 are found, t_L and t_b , hence the fall time of the signal, can be computed. The fall time, t_f , can be stated as:

$$t_f = t_b - t_a = 1.25 (t_2 - t_1) \quad (3.4)$$

Figure 3.2 Ramp function representation using two points from the waveform.

Another approach is to use threshold voltages of the MOS transistors as break points instead of the using $0.9V_{DD}$ and $0.1V_{DD}$. When the input of an inverter is above $(V_{DD} + V_{tp})$, the PMOS transistors will be 'OFF'. On the other hand, when the input is below V_{tn} , the NMOS transistors will be 'OFF'. This is also true for M-input NAND gate and M-input NOR gate if the triggering input is connected to the NMOS transistor closest to ground and the PMOS transistor closest to V_{DD} respectively. Hence, using these two points to define the transition period of an output gives a more meaningful waveform representation for the input of the next stage. Using Figure 3.2 with $(V_{DD} + V_{tp})$ replacing $0.9V_{DD}$ and V_{tn} replacing $0.1V_{DD}$, the fall time becomes:

$$t_f = t_b - t_a = (t_2 - t_1) \cdot \frac{V_{DD}}{(V_{DD} - V_{tp}) - V_{tp}} \quad (3.5)$$

Using two points to define the ramp function is a more accurate representation than using the slope at $V_{DD}/2$ because more information from the waveform is used. This

increased accuracy, however, comes at the cost of increased computation and simulation overhead.

3.2.3 Piecewise Linear Waveform Representation

The third type of waveform representation is the piecewise linear method. Straight lines with different slopes are joined together to approximate the waveform. Theoretically, there is no limit in the number of lines used. But the complexity of the model increases with the number of lines.

Deng[6] uses two straight lines in his model. To define the waveform representation, the slopes of the two segments are needed. Three points from the actual waveform are required to compute the slopes. Figure 3.3 shows an example of the two-segment piecewise linear model used by Deng. The first segment is a line with endpoints at t_{si} and t_{so} - the points at which the input and output cross the $V_{DD}/2$ level respectively. The second segment extends from the endpoint of the first segment at t_{so} to V_{DD} , passes 90% of V_{DD} at t'_{ss} and cuts V_{DD} at t_{ss} .

Figure 3.3 Two-segment piecewise linear representation of a waveform purposed by Deng[6].

Alternatively, threshold voltages can be used as break points, as illustrated in Figure 3.4. The time interval of S_1 , $(t_{s0}-t_{s1})$, can be calculated from t_{s0} and t'_{s1} using equation (3.6).

Figure 3.4 Two-segment piecewise linear waveform representation using threshold voltages as break points.

$$t_{r,s_1} = t_{so} - t_{s1} = (t_{so} - t'_{s1}) \left(\frac{V_{DD}/2}{V_{DD}/2 - V_{in}} \right) \quad (3.6)$$

Similarly, the time interval of S_2 , $(t_{s2}-t_{so})$, can be calculated from t_{so} and t'_{s2} using equation (3.7).

$$t_{r,s_2} = t_{s2} - t_{so} = (t'_{s2} - t_{so}) \left(\frac{V_{DD}/2}{V_{DD}/2 - V_{tp}} \right) \quad (3.7)$$

The reasons for using V_{in} and V_{tp} are the same as for the ramp function representation. The $V_{DD}/2$ point is used as the meeting point of the two segments to simplify the simulation process since the $V_{DD}/2$ point is also needed in calculating the delay. Unlike the ramp function representation, which only has one option in passing the rise/fall time to the next stage, there are a few options possible for this representation. First, only the rise time of either one of the segments is passed on. This would result in one

extra parameter in the model equations and one more equation in the model. Or the rise times of both segments can be passed to the next stage. This would result in two extra parameters in the model equations and two more equations in the model.

3.2.4 Piecewise Non-linear Waveform Representation

The fourth method for waveform representation uses a straight line with an exponential tail to approximate the waveform. Matson's model uses this method. An example of this method is shown in Figure 3.5. This representation consists of a time shift (T_{BE}), a ramp (line AB) and an exponential tail with a time constant of T_{SW} . As stated by Matson, conceptually T_{BE} is the time until the output begins to move in response to an input transition and T_{SW} is a measure of how fast the output switches. Only T_{BE} and T_{SW} are used in Matson's model. Although the ramp exists in the representation, it is not used in his model.

Figure 3.5 Waveform representation using a ramp and an exponential tail

This method is a more accurate representation but at increased cost. More effort in both simulation and curve fitting are required. The problem of curve fitting is affected

more since the representation consists of two independent parts. It is more complicated to curve fit through both of them. The problem of locating the boundary of the ramp and the exponential part has to be addressed also. Two extra variables and two more equations have to be included in the model.

Another piecewise non-linear waveform representation consists of a time shift and a single time constant exponential waveform. It is assumed that the initial portion of the exponential waveform is linear enough so that the waveform 'looks' like a ramp function with an exponential tail. Figure 3.6 shows an example for rising output. The exponential

Figure 3.6 Piecewise non-linear waveform representation for a rising output.

waveform intercepts the original waveform at two points - when the actual waveform reaches V_{in} and when the actual waveform reaches $V_{4\tau}$ - the voltage of an exponential waveform at four times of the time constant. In Figure 3.6, t_1 and t_2 are the times at which the output reaches V_{in} and $V_{4\tau}$ respectively. Hence, t_2 can be expressed by equation (3.8).

$$t_2 = t_{st} + 4\tau \quad (3.8)$$

With these two points, the time constant of the exponential waveform (τ) and the time at which the exponential starts (t_{st}) can be computed using equation (3.9) and equation (3.10) respectively.

$$\tau = (t_2 - t_1) / \ln\left(\frac{V_{DD} - V_{in}}{V_{DD} - V_{4\tau}}\right) \quad (3.9)$$

$$t_{st} = t_2 - \tau \cdot \ln\left(1 - \frac{V_{4\tau}}{V_{DD}}\right) \quad (3.10)$$

Only the time constant is passed on to the next stage as the input rise/fall time. So there is no need to calculate t_{st} in the model. This results in one extra parameter and one extra equation in the model. WASIM[17] by Nassif uses a similar representation with a slightly different definition of the exponential function which uses three variables.

3.3 Loading

The loading of a cell is an important factor of the performance of the cell. The loading is caused by the fanout gates and the interconnection wire as shown in Figure 3.7.

Figure 3.7 Components of the Loading.

The interconnection can be modeled as an RC circuit. Since the interconnection is made by material of high conductivity and the current is small, the effect of the resistor is usually ignored. This will greatly simplify the timing model.

For the fanout gates, loading comes from the gate resistance and gate capacitances of the transistors connected to the switching input. Since the gate resistance is very large, it is usually ignored. The gate capacitance consists of the gate-to-drain capacitance, the gate-to-source capacitance and gate-to-bulk capacitance which are all non-linear. Hence, the value of the gate capacitance changes as the input switches. So there is a need to model the gate capacitance. Some models, like the Matson's model, develop a set of equations to model this effect (equation (2.29.n) - (2.29.u)). Some models, like the Hoppe's model[14], throw in another independent variable to account for this effect. Other models choose to ignore this effect.

With these simplification, the loading is the summation of the interconnect capacitance and the gate capacitance of the fanout gates.

3.4 Input Combination

Both the propagation delay through a gate and the output transit time depend on which inputs are switching, as well as how many are switching. Since there exists many different combinations of input states, it is not feasible to model all the combinations. The following three input vectors represents the boundary cases which are the more suitable candidates:

- 1) only the input connected to the series transistor nearest to the output switches, with other inputs stable(Figure 3.8).
- 2) only the input connected to the series transistor nearest to the power rail (V_{DD} or ground) switches, with other inputs stable(Figure 3.9).

3) all the inputs switch at the same time and at the same rate.

Figure 3.8 Input condition for case 1 for (a) 3-input NAND gate and (b) 3-input NOR gate.

Figure 3.9 Input condition for case 2 for (a) 3-input NAND gate (b) and 3-input NOR gate.

3.5 Summary

Various problems facing the development of a timing macromodel are discussed. These problems include the waveform representation, the gate topology, the loading and the input combination. Possible solutions to these problems are presented. The pro and con of these solutions are also examined.

Chapter 4

Overview of the New Proposed Model

This chapter presents the theoretical aspects of a new approach to macromodeling. It also describes the choices that were made from among the available options presented in Chapter 3 and provides justification for these choices.

4.1 Definition of Model Parameters

This section describes the various options chosen as the solutions of the problems facing the development of a timing model as discussed in Chapter 3.

4.1.1 Gate Topology

Model equations are developed for the inverter, NAND and NOR gates in the proposed model. The number of inputs to the gates is limited to four.

For the inverter, the ratio between the width of the transistors is fixed to make the rising and falling outputs symmetrical. For the other gates, each type of transistor in the gate has a uniform width. The two options stated in equations (3.1)-(3.3) are supported.

4.1.2 Waveform Representation

Three types of waveform representations mentioned in Chapter 3 are implemented for the purpose of comparison. The three types of representations are:

- 1) ramp function representation - the $(V_{DD} - V_{tp})$ and V_{tn} levels are used to define the ramp function as stated in equation (3.5);
- 2) piecewise linear waveform representation - the two segment representation shown in Figure 3.4 are chosen for the proposed model;

- 3) piecewise non-linear waveform representation - the single time constant exponential function with a time shift as shown in Figure 3.6 is chosen for implementation.

4.1.3 Loading

Only capacitive load is considered in the proposed model. This load is the summation of input capacitance of the next stage and the parasitic capacitance from the interconnection. The input capacitance of the next stage is the sum of the gate capacitances of the transistors connected to the output of the gate. The gate capacitance for each transistor is estimated from the area of the gate using the following equation:

$$C_{gate} = C_{ox} \cdot W \cdot L \quad (4.1)$$

where C_{ox} is oxide capacitance per unit area, W and L are the width and length of the channel of the transistor.

4.1.4 Input Vector

An input vector is chosen to represent all the input combinations. The three input vectors stated in section 3.4 of page 34 are implemented and compared to determine the input vector to be used. The three input vectors are:

- 1) only the input connected to the series transistor nearest to the output switches;
- 2) only the input connected to the series transistor nearest to the power rail (V_{DD} or ground) switches;
- 3) all the inputs switch simultaneously.

4.2 Development of Model Equations

4.2.1 General Form

The general forms of the equations for the propagation delay and the output waveform are taken as

$$f(\hat{x}) = c_1\phi_1(\hat{x}) + c_2\phi_2(\hat{x}) + c_3\phi_3(\hat{x}) + \dots + c_p\phi_p(\hat{x}) \quad (4.2)$$

where $\hat{x}$ is the vector of the model parameters, $\phi_i(\hat{x})$ are non-linear functions of x and c_i are the linear coefficients for the equation. Each product, $c_i\phi_i(\hat{x})$, in equation (4.2) is called a term. The general form is a summation of terms. More terms can be added if higher accuracy is required. Terms can be replaced if the replacement yields a better approximation to the simulation result. Alternatively, terms can be removed to achieve faster speed. The strategy of this model is to start off with a long polynomial with very high accuracy. Then terms are removed to obtain a simpler polynomial until the accuracy is dropped to the user specified level.

4.2.2 Computation of Coefficient Values

Curve fitting is used to determine the values for the coefficients in equation (4.2) based on the data points obtained from SPICE. By inspection of equation (4.2), we see that although the $\phi_i(\hat{x})$'s are non-linear functions, $f(\hat{x}_r)$ is linear in terms of the coefficients, c_i . Hence, the general linear curve fitting techniques can be applied. The residual of a data point is defined as

$$e_r = b_r - f(\hat{x}_r) \quad (4.3)$$

where b_r is the y-value of a data point and $f(\hat{x})$ is the value of the fitting function evaluated at $\hat{x}_r$. Residuals are used to measure the 'closeness' of a function $f(\hat{x})$ to a set of data points. Three common ways exist to combine residuals into a single closeness measurement. These are:

$$\begin{aligned}
 l_1 &= \sum_{i=1}^m |e_i| && \text{sum of absolute residual} \\
 l_2 &= \sqrt{\left(\sum_{i=1}^m e_i^2 \right)} && \text{square root of sum of square of residual} \\
 l_\infty &= \text{Max } |e_i| \text{ for all } i && \text{maximum of absolute residual}
 \end{aligned} \tag{4.4}$$

The l_2 norm is the well-known least square fit, which is used here. Solving a linear fitting function is basically solving a set of simultaneous linear equations. Matrix computation techniques are used to systematically compute these coefficients. These techniques include the Householder's Transformation and the QR Factorization, which are described in detail in Appendix A. There exist a number of commercially available software packages to solve the linear curve fitting problem using these techniques. The IMSL library of numerical routines is one of them and is used here.

4.2.3 Model Accuracy

For the transistor sizing problem, the timing model should be simple enough to keep the computation time small. But the accuracy of the model must be maintained at a reasonably high level. So it is proposed to start with an very accurate polynomial and remove those 'less' important terms. This removal of terms decreases the accuracy of the model. This process is stopped when the accuracy drops to a user specified level. It is also very important to have a good original equation.

Before a scheme can be developed to simplify the polynomial, one needs to define the 'importance' of a term and the 'accuracy' of the model. For each data point, the 'weight' of each term can be measured by the magnitude of the term over the sum of all the terms. As an example, the 'weight' of the $c_2\phi_2(t)$ term in equation (4.2) is

$$I_{\phi, \phi} = |(c_2 \phi_2(z)) / f(z)| \quad (4.5)$$

A boundary can be set to distinguish the 'important' terms. If the weight (I) is above the boundary, the corresponding term is 'important'. I is different for different data points since the variables change from one data point to others. Therefore, a term may not be 'important' all the time. The 'importance' of a term is then defined as the number of points for which that term is 'important' over the total number of the data points.

The 'accuracy' of the model is defined as the number of 'perfect' data points from the model over the total number of data points. A data point from the model is 'perfect' if its difference with SPICE is within the tolerance of the model. Obviously, the higher the accuracy is the better the model is.

4.2.4 State-Space Graph Formulation

This problem of removing terms can be visualized by using the state-space graph. An example is shown in Figure 4.1. Each node in the graph represents a unique combination of the terms from the original polynomial. Each node will have different accuracy. Node V_s represents the original polynomial, which has five terms in this example. Each directed arc represents the removal of a term. All the nodes in level 1 have four terms. All the nodes in level 2 have three terms and so on. Node V_f represents the equation with no terms in it. Therefore the task is to locate the node with an accuracy greater than or equal to the specified accuracy and at the lowest possible level.

Two separate algorithms are developed to implement this process. Some parameters, like the accuracy of the final model, have to be specified before applying these algorithms. The following sub-sections provide a detailed description of these two algorithms.

Figure 4.1 State-space graph to illustrate the problem of removing terms.

4.2.5 Algorithm 1

To use this algorithm, two parameters must be specified. They are the accuracy of the final model and the maximum drop in accuracy per term. This algorithm is basically a greedy algorithm with some modification. The procedure of the algorithm is as follows:

Algorithm 1

Input: P = a polynomial; A = a vector of simulation data; D = maximum drop in accuracy when one term is removed; A = minimum accuracy of the model
Output: $P1$ = a new polynomial

Begin

```

ACC := run curve fitting to determine accuracy of P1;
I := importance of each term in P1;
continue := TRUE;
while ( continue = TRUE )
do
    i := the term in P1 with the least importance and had not been tried;
    if ( i cannot be found ) then
        continue := FALSE;
    else
        set i to be tried;
        P2 := P1 with i removed;
        ACC2 := run curve fitting to determine accuracy of P2;
        I2 := importance of each term in P2;
    end if
until ( (ACC-ACC2) <= D or continue = FALSE )
if (ACC2 > A ) then
    P1 := P2;
    ACC := ACC2;
    I := I2;
else
    continue := FALSE;
end if
end while
return P1;
end

```

This algorithm uses the 'importance' of the terms as an indicator in determining which term is to be deleted. It then checks the accuracy of the polynomial after the term is deleted. If the drop in accuracy is not acceptable, another term is picked and deleted and the accuracy of the new polynomial is checked again. Otherwise, it will continue to delete another term from the new polynomial.

This algorithm exits either when deleting any term from the polynomial will decrease the accuracy below the specified one, or when deleting any term will cause a

drop in accuracy greater than the maximum drop specified. In the worst case, the total number of nodes visited by the algorithm is $\frac{n(n+1)}{2}$, where n is the number of terms in the original polynomial and there will be only one term in the output polynomial. But on the average, the number of nodes visited will be much smaller than this.

The bold line in Figure 4.2 shows a typical path of nodes which could be visited by this algorithm. Node V_s always is the starting point. The dotted path shows the bound back situation because of the drop in accuracy is too high. V_f is the end point. The algorithm starts with the original five term polynomial (node V_s), removes the least important term and moves to node V_2 . It recalculates the accuracy of the new four-term polynomial, the drop in accuracy, and the importance of the remaining four terms. The algorithm has to check if this new accuracy is still above the specified accuracy and if the drop in accuracy is below the maximum drop allowed. If either one of these conditions fails, the algorithm has to bound back to node V_s . If both conditions are passed, the algorithm continues to remove the least importance term among the four remaining terms and moves to node V_{10} . It then redoes the calculation and checks the two conditions again. Then it removes another term and moves on to node V_{22} . At this node, it finds that one of the conditions failed. So it bounds back to node V_{10} and moves to node V_{16} by removing the second least importance term. Finally, the algorithm is at node V_f and finds that it has to bound back to node V_{26} . At node V_{26} , the algorithm finds that there is no other term that can be removed. So it exits and the polynomial corresponding to node V_{26} is the model equation to be used.

4.2.6 Algorithm 2

Algorithm 1 is fast but it is likely that the result is a local minimum - the returned polynomial is not the simplest polynomial with the specified accuracy.

Figure 4.2 A typical path of nodes that may be travelled by Algorithm 1.

This algorithm is slower than algorithm 1 but the result is usually better than that of algorithm 1. This algorithm is not an exhaustive search for the simplest polynomial. So it is still possible that it is trapped in a local minimum. But it is less likely to happen. The number of nodes visited by this algorithm can be given by the following equation.

$$\text{No of Nodes visited} = \frac{(n+r-1)(n-r+2)}{2} \quad (4.6)$$

where n is the number of terms in the original equation and r is the number of terms in the final equation.

To use this algorithm, only the final accuracy of the model needs to be specified.

The procedure of the algorithm is as follows:

Algorithm 2;

Input: P = a polynomial; B = a vector of simulation data; A = minimum accuracy of the model.

Output: $P1$ = a new polynomial

Begin

$P2 := P;$

$maxACC :=$ run curve fitting to determine accuracy of $P2;$

while $maxACC \geq A.$

$P1 := P2;$

$maxAcc := -1;$

for $i := 1$ to no. of terms in $P1$

$P3 := P1$ with i -th term deleted;

$ACC =$ run curve fitting to determine accuracy of $P3;$

if ($ACC \geq maxAcc$) **then**

$maxAcc := ACC;$

$P2 := P3;$

end if

end for

end while

return $P1;$

end

In this algorithm, every term in the polynomial is deleted, one at a time, and the accuracy is calculated for each polynomial. So, if there are n terms in the polynomial originally, after this process, there will be n different $(n-1)$ -term polynomials and n accuracies, one for each polynomial. The polynomial with the highest accuracy among the n polynomial is chosen to repeat the process. The algorithm exits when all n accuracy are smaller than the specified accuracy.

The bold line in Figure 4.3 shows a typical path of nodes which could be visited by this algorithm. The algorithm starts at node V_5 and visits node V_1 , V_2 , V_3 , V_4 and V_5 . During each visit, the algorithm calculates the accuracy of the polynomial corresponding to that node. It finds that the polynomial corresponding to node V_3 has the highest accuracy among the five nodes and the accuracy is still above the specified accuracy. So it

settles on node V_3 . It then visits V_7 , V_{10} , V_{13} and V_{14} . It then settles on node V_{10} which has the highest accuracy and this accuracy is still above specified accuracy. Finally, the algorithm settles on node V_{16} . After visiting node V_{26} and V_{27} , the algorithm finds that both nodes have a polynomial with an accuracy below the specified one. So the algorithm exits and the polynomial corresponding to node V_{16} is used. Note that in this algorithm, there is no bound back and more nodes are visited.

Figure 4.3 A typical path of nodes that may be travelled by Algorithm 2.

4.3 Summary

The proposed timing model are presented. The variables in the models and the restrictions imposed are defined. The transistors in the gate have uniform sizes depending

on the types of the transistors. The two sizes are also related. The waveform representation used is chosen from the three implemented waveform representations. The loading of a gate is taken as the summation of the gate capacitance of the input transistors in the next stage and the wiring capacitance. The gate capacitance is estimated from the area of the channel. The input vector is also chosen from three input combinations based on experimental result. The general format of the model equations are presented. Two algorithms are developed to simplify the model equations.

Chapter 5

Simulation Result

This chapter presents extensive simulation results for a sample technology as well as the experimental framework used for the collection of this data. The program is written in C and uses HSPICE to simulate the gate. This chapter explains the duty of the C program. It also covers various practical aspects of the SPICE simulation. The timing performance of the logic gate under various conditions is also examined.

5.1 Experimental Framework

Figure 5.1 shows the block diagram of the experimental framework.

Figure 5.1 Block Diagram showing the configuration of the system.

The program sweeps through a range of values for various parameters, such as the width of the MOS transistors, the load capacitor,.... etc. It sets up the input file for HSPICE with the provided netlist of the circuit. It calls HSPICE to perform the transient analysis

and to evaluate the delay and the output rise/fall time. After HSPICE finishes the analysis, the program extracts these data from the HSPICE output file and stores it in a file. This process is repeated until a pre-determined amount of data is collected which covers the practical range of the parameters.

For the simulation, the HSPICE level 3 MOSFET model is used. This level of model considers most of the important parasitic effects, like the channel length modulation, the subthreshold current and the carrier mobility reduction. The drain and source capacitances are also included in the simulation. The area and the perimeter (excluding the channel side) of the drain and source for each transistor need to be included in the netlist of the circuit. Figure 5.2 summarizes the assumptions made about the layout of each transistor.

Figure 5.2 Assumed Layout for the transistors.

From Figure 5.2, the area of the drain of the NMOS transistor, AD_n , is

$$AD_n = AS_n = W_n (d_1 + d_2 + d_3) \quad (5.1)$$

where AS_n is the area of the source. Similarly, the perimeter of the drain of the NMOS, PD_n , is

$$PD_n = PS_n = W_n + 2(d_1 + d_2 + d_3) \quad (5.2)$$

where PS_n is the perimeter of the source.

Simulations cover the practical range of each parameter: from the minimum value of the parameter to 25 times the minimum value. For the width of the transistors, the minimum values are obtained from physical design rules. For the load capacitance, it is the input capacitance of the smallest possible inverter. The input rise time is more complicated since both the minimum and maximum input rise/fall time have to be found. Figure 5.3 is used to illustrate how the two rise/fall times are computed. The minimum rise/fall time is the output rise/fall time of an inverter with maximum width connected to an inverter with minimum width (Figure 5.3a). On the other hand, the maximum rise/fall time is the output rise/fall time of an inverter with minimum width connected to an inverter with maximum width (Figure 5.3b). Using these maximum and minimum, the step size for the input rise/fall time can be computed as

$$t_{step\ size} = (t_{max} - t_{min}) / (25 - 1) \quad (5.3)$$

Figure 5.3 Circuits used to compute (a) the min. rise/fall time and (b) the max. rise/fall time.

In order to speed up the simulation, the 'SWEEP' statement is used so that each call to HSPICE can sweep through the range of one parameter. Then a few sets of data can be collected per HSPICE call. Since the model only needs a few points from the input and output waveforms, the '.MEASURE' statement from HSPICE is used to obtain these data points.

5.2 Summary of Simulation Results

This section looks at the behavior of the logic gates under various conditions. These conditions include variations of the independent variables, different input vectors and different output loading conditions. The ramp function representation is used to represent the input and the output.

5.2.1 Effect of Independent Variables

The effect of three independent variables - the width of the transistors (W_n), the load capacitance (C_L) and the input rise time (T_r) - on the logic gates are examined. For the following three dimensional mesh surface, the x and y axes represent the independent variables which are expressed as the multiple of their step size. For example, the relationship between T_r and the actual input rise/fall is as follow:

$$T_{r, actual} = T_{r, min} + T_r \cdot T_{r, stepsize}$$

Figure 5.4 shows the effect of the independent variables on the propagation delay of an inverter. It shows that the propagation delay is inversely proportional to W_n while directly proportional to C_L and T_r . W_n has the most significant effect on the delay. The effect of C_L is quite linear while the effect of T_r shows some non-linearity.

(a)

(b)

(c)

Figure 5.4 The effect of W_n , C_L and T_r on the delay of an inverter:

- (a) delay as functions of C_L and T_r with $W_n=10$;
- (b) delay as functions of W_n and T_r with $C_L=10$;
- (c) delay as functions of W_n and C_L with $T_r=10$.

Figure 5.5 shows the effect of the independent variables on the fall time of an inverter. The general behavior of the fall time is similar to that of the propagation delay. However, the effect of T_r is much more pronounced.

Figure 5.6 shows the propagation delay for a 4-input NAND gate and a 4-input NOR gate. The transistor size ratio of the two gates are specified by equations (3.2) and (3.3) respectively. For the NAND gate, the effect of W_n and C_L is similar to that of the inverter. But the propagation delay becomes inversely proportional to T_r . Figure 5.6d shows that the propagation delay is increased if the size of the transistors is large. Because the transistor size ratio of the NOR gate is specified by equation (3.3), a small increase in W_n means a large increase in W_p and hence the parasitic capacitance. The increases in the parasitic capacitance offset the increase in the driving capability of the transistor. Hence, the delay increases even though the size of the transistors is increased. Figure 5.7 shows that the output of a 4-input NAND gates with a fast input and a slow input. It shows that the effect of the input rise time on the output is diminishing. The delay is increased at a rate smaller than that of the input rise time. Because of the definition of the propagation delay, the propagation delay becomes inversely proportional to the input rise/fall time as in Figure 5.6a and Figure 5.6c.

5.2.2 Effect of Input Vectors

The effect of different input vectors are examined using a 4-input NAND gate. The three input vectors examined are:

- 1) only the input connected to the series transistor nearest to the output switches;
- 2) only the input connected to the series transistor nearest to the power rail (V_{DD} or ground) switches;
- 3) all the inputs switch simultaneously.

(a)

(b)

(c)

Figure 5.5 The effect of W_n , C_L and T_r on the output fall time of an inverter:

- (a) output fall time as functions of C_L and T_r with $W_n=10$;
- (b) output fall time as functions of W_n and T_r with $C_L=10$;
- (c) output fall time as functions of W_n and C_L with $T_r=10$.

(a)

(b)

(c)

(d)

Figure 5.6 The behavior of delay of NAND and NOR gates:

- (a) delay of 4-input NAND gate as functions of C_L and T_r with $W_n=10$;
- (b) delay of 4-input NAND gate as functions of W_n and C_L with $T_r=10$;
- (c) delay of 4-input NOR gate as functions of C_L and T_r with $W_n=10$;
- (d) delay of 4-input NOR gate as functions of W_n and C_L with $T_r=10$.

Figure 5.7 The output of a 4-input NAND gate with a fast and a slow input.

Figure 5.8 shows the output of a 4-input NAND gate with the above input vectors. The voltage waveform of the intermediate nodes are also shown. For input vector (1), the intermediate nodes are initially set to zero. For input vector (2) and input vector (3), the intermediate nodes are initially set to V_{DD} . The environment and parameters of the NAND gate are the same for the three input vectors. The results show that the propagation delay varies significantly with the different input vectors.

SPICE simulations were performed on the NAND gate over the pre-set range of the three independent variables for all three input vectors. Figure 5.9 shows the comparisons between the propagation delay of the input vectors. It shows that the gate with input vector (3) has the largest delay among the three vectors. The reason can be found in Figure 5.8c. Figure 5.8c shows that the NMOS transistors of the 4-input NAND are turned on one by one - M4 first, then M3 after a while, then M2 and finally M1. The

Figure 5.8 Voltage waveforms for the 4-input NAND gate using: (a) input vector (1), (b) input vector (2), (c) input vector (3).

* The four PMOS transistors are represented by one PMOS transistor since they are in parallel.

Figure 5.9 Comparison between the propagation delay with the three input vectors: (a) input vector (1) and input vector (2); (b) input vector (2) and input vector (3); (c) input vector (1) and input vector (3).

short circuit is also the largest since all four PMOS transistors are needed to be turned off. These factors make this input vector have the largest delay.

Figure 5.9a shows that neither input vector (1) nor input vector (2) produces result that are always greater than the other. After a closer examination of the data, it is found that input vector (1) produces larger result when T_r is large. W_n and C_L only have minimal effect. Figure 5.8a shows that the intermediate nodes of the NAND gate are charged up during the input switches. This limits the current flowing through M1 and prolonging the delay. The slower the input is, the longer the current is limited.

The effect of the input vectors on the output fall time is relatively small. The maximum magnitude of the differences is about 13% and the majority are below 5%. Therefore, the main concern in choosing an input vector is the propagation delay. The propagation delay based on input vector (3) is on average 30% larger than the other. Experimental comparisons are based on a single critical timing path through a logic network. Design using input vector (3) would be too pessimistic. Therefore input vector (2) is used as the input vector for the proposed timing model. It is chosen over input vector (1) because it gives a larger delay for most of the cases.

5.3 Summary

The chapter presented an overview of the software system used to collect SPICE simulation data as well as the assumptions made about the layout of the logic gates to include the parasitics capacitances into the simulations. The effect of the independent variables and input vectors on the delay and output rise/fall time were also examined for selected logic gates.

Chapter 6

Macromodeling Results

This chapter presents the program that produces the final model equations for logic gates. The model equations for the three waveform representations are generated with the two proposed algorithms. Because of the difference in the original model equations and the input waveform, each representation has its own program and simulation data. The proposed timing model is compared with SPICE for accuracy and efficiency. A comparison between the two algorithms for polynomial simplification is also presented.

6.1 System Overview

Figure 6.1 shows the block diagram of the program to generate the model equations. The program is written in C. The two proposed algorithms in Chapter 4 are implemented.

Figure 6.1 Overview of the system for generating the model equations.

At start up, the program reads in the necessary information from a file. This file specifies which algorithm should be used and contains the filename that stores the SPICE data, the output filenames, the required accuracy of the final model and some other information. The program reads in the data points, and calls the chosen algorithm to start the process. After the algorithm is finished, the final model equation and the coefficients are stored in a file. The distribution of errors and the 'importance' of each term in the final equation are also stored in different files.

IMSL is a numeral library used for the curve fitting problem. It is basically used to solve the problem:

$$\text{minimize } \|\Phi\hat{c} - \hat{b}\|_2 \quad (6.1)$$

where $\hat{c}$ is the vector of coefficients, $\hat{b}$ is the vector of SPICE data point and Φ is the $m \times n$ matrix of $\phi_i(\hat{x})$. (Refer to Appendix A for detailed explanation.) Matrix Φ is normalized with $\hat{b}$ to increase the accuracy of curve fitting using equation (6.2). Then $\hat{b}$ becomes a vector of 1s. Φ and $\hat{b}$ are computed by the algorithm before the subroutines in IMSL are called.

$$\Phi[i, j] = \Phi_{orig}[i, j] / b_i \quad \text{for all possible } j \quad (6.2)$$

Certain parameters have to be set before either simplification algorithm can be started. These parameters include:

- i) the tolerance of the model which has a default value of 5%;
- ii) the boundary to distinguish the 'import' terms which has a default value of 2%;
- iii) the maximum decrease in accuracy when a term is removed, which has a default value of 4%;
- iv) the minimum accuracy of the model which has a default value of 90%.

6.2 Comparison of the Two Simplification Algorithms

Table 6.1 compares the results of the two algorithms described in Chapter 4. The number of terms, the accuracy of the final equations and the time used are listed in the table for each type of gate. The equations are for the propagation delay using the ramp function representation. The gates listed have asymmetrical outputs. The original equation is the same for each gate type and for both algorithms.

Type of Gate	Algorithm 1			Algorithm 2		
	no. of terms	accuracy (%)	time used (sec)	no. of terms	accuracy (%)	time used (sec)
Inverter	8	92.72	328.2	7	92.90	4873.5
2-input NAND*	5	94.24	359.4	5	94.24	4762.3
3-input NAND	5	92.90	384.9	5	93.98	4817.3
4-input NAND*	4	92.99	385.9	4	92.99	4879.0
2-input NOR	6	95.78	400.6	6	96.09	4841.9
3-input NOR	6	93.86	408.0	6	94.44	4881.1
4-input NOR	6	95.26	377.9	5	91.14	4986.6

Table 6.1 Comparison between Algorithm 1 and Algorithm 2 for the ramp function waveform representation.

* The results of Algorithm 1 and Algorithm 2 are the same for this gate.

Equation (6.3) shows the original polynomial for the propagation delay of the 2-input NAND gate in Table 6.1. It has 22 terms and an accuracy of 99.96%.

$$\begin{aligned}
 D = & a_1 + a_2 C_i^2 + a_3 T_r + a_4 T_r C_i + a_5 T_r C_i^2 + a_6 \sqrt{T_r} + a_7 C_i \sqrt{T_r} + a_8 / W_n + a_9 C_i^2 / W_n \\
 & + a_{10} T_r / W_n + a_{11} T_r C_i^2 / W_n^2 + a_{12} / W_n^2 + a_{13} C_i / W_n^2 + a_{14} C_i^2 / W_n^2 + a_{15} T_r / W_n^2 \\
 & + a_{16} C_i T_r / W_n^2 + a_{17} \sqrt{T_r} / W_n^2 + a_{18} C_i \sqrt{T_r} / W_n^2 + a_{19} C_i^2 \sqrt{T_r} / W_n^2 \\
 & + a_{20} C_i \sqrt{T_r} / W_n + a_{21} C_i^2 \sqrt{T_r} / W_n + a_{22} W_n C_i \sqrt{T_r}
 \end{aligned} \tag{6.3}$$

The final equation for the 2-input NAND gate is shown in equation (6.4) which has 5 terms and an accuracy of 94.24%.

$$D = a_1 + a_2 T_r + a_3 C_i^2 / W_n + a_4 C_i \sqrt{T_r} / W_n + a_5 C_i^2 \sqrt{T_r} / W_n \quad (6.4)$$

The final equation given by algorithm 2 is usually better than or the same as the equation given by algorithm 1 in terms of the number of terms and the accuracy. Algorithm 1 gives a better result only in a few rare occasions. But the time required by algorithm 2 is an order of magnitude greater than that required by algorithm 1. Moreover, the result from algorithm 2 is usually only marginally better.

6.3 Comparison with SPICE

This section describes how the models are compared with SPICE and the result of the comparison.

6.3.1 Configuration of Testing System

After the model equations for all the gates are obtained, the model is compared with SPICE using circuits with chains of gates. Figure 6.2 shows the block diagrams of the program that performs the test. The three waveform representations implemented are all compared with SPICE.

The program first reads in three data files. The first file describes the connection of the circuit in BLIF format[24]. The second file specifies the environment of the circuit - the loading, the input transit time, the type of input and the sizes of the gates. The third file contains the SPICE macro definition of the logic gates and the model parameters for the process technology. The program then checks if the circuit can form a chain from the input to the output. If it can, it will set up the SPICE input file, call HSPICE to perform the

Figure 6.2 Configuration of the program for testing the models.

simulation, extract the result from the SPICE output. It will also compute the same result using the model equations and compare the two sets of results.

6.3.2 Results

Table 6.2 lists the number of terms in the final equations and the accuracy of the model equations for the three waveform representations. The multi-input gates are the types with asymmetrical output. The equations, generated by algorithm 2, are for the propagation delay of the logic gates. Table 6.2 shows that the model using the exponential waveform representation has the simplest model equations and yet the accuracy is high. The worst one is the piecewise linear representation.

Table 6.3 lists the delay and the output transit time from SPICE and the three timing modeling for three different circuits. Each of the three circuits has about ten different logic gates in a chain. The inputs for the three circuits are rising ramps with different rise times. In this comparison, the input and output transit times are defined as

the time interval between $0.1V_{DD}$ and $0.9V_{DD}$. Therefore, conversion between transit time definitions has to be performed before and after the computation for the three waveform representations. The description of the circuits can be found in Appendix B.

Types of Gate	Ramp		Exponential		Piecewise Linear	
	No. of terms	Accuracy (%)	No. of terms	Accuracy (%)	No. of terms	Accuracy (%)
Inverter	7	92.90	7	95.01	18	90.75
2-input NAND	5	94.24	6	98.36	10	92.60
3-input NAND	5	93.98	3	95.81	9	92.30
4-input NAND	4	92.99	3	96.38	4	91.48
2-input NOR	6	96.09	4	95.07	9	90.58
3-input NOR	6	94.44	4	95.96	6	92.50
4-input NOR	5	91.14	7	96.47	5	93.00

Table 6.2 Comparison of the complexity of the three waveform representations.

Circuit No.	SPICE		Ramp		Exponential		Piecewise Linear	
	Delay (ns)	Transit time (ns)	Delay (ns)	Transit time (ns)	Delay (ns)	Transit time (ns)	Delay (ns)	Transit time (ns)
Ex. 1	2.3276	0.2615	2.5594	0.2290	2.7255	0.2473	2.9121	0.3262
Ex. 2	6.0148	0.3908	6.0483	0.3239	6.3606	0.5743	6.5845	0.4879
Ex. 3	5.0124	0.5077	4.6901	0.5608	4.9451	0.5279	5.3437	0.6048

Table 6.3 Experimental Results

Table 6.4 presents a comparison of the results of Table 6.3. It shows the best case, the worst case and the average for all the circuits and timing models. A negative value means that the result produced by the model is smaller than the one produced by SPICE. The average in Table 6.4 is the absolute value of the percentages. The transit time has a relatively high error for all the cases. One of the main reasons is the different transit time definition. Figure 6.3 shows the SPICE output waveform of the test circuit in Ex. 3. It also shows the ramp function waveform representation of the output and the representation used in this comparison. The two lines already have more than 10% difference in the rise time.

Figure 6.3 Output waveform and two of its representation for the circuit in Ex3.

Waveform Representation	Delay			Transit Time		
	Best Case (%)	Worst Case (%)	Average (%)	Best Case (%)	Worst Case (%)	Average (%)
Ramp	0.5573	9.9578	5.6482	10.464	-17.12	13.342
Exponential	-1.3425	17.093	8.061	-5.6487	-46.966	18.866
Piecewise Linear	6.6105	25.111	13.731	19.126	24.86	22.903

Table 6.4 Comparison of the accuracy of the three waveform representations with SPICE.

6.4 Summary

The model equations for the three waveform representations were generated using both simplification algorithms. Algorithm 1 proved to be a better compromise between the simplicity of the model and its accuracy. The three timing models are compared with SPICE for a few circuits. The result showed that the ramp functions waveform representation is the best among the three representation in term of accuracy.

Chapter 7

Conclusions

7.1 Summary

This thesis examined the problem space for developing a timing model. The proposed timing model is limited to inverter, NAND and NOR gates. Moreover, each type of transistor in a gate has a uniform width and the two widths are related. This enables the model to use one parameter to uniquely represent transistor sizes in a gate. Two different layout options exist allowing the gates to have symmetrical and asymmetrical outputs. The loading is the input capacitance of the next stage plus the parasitic capacitance of the interconnect. Three input vectors were examined to determine which one will be used in the model. Input vector (2) on page 36 was chosen for experimentation. Two algorithms were developed to generate the final equations from an original long polynomial. Whereas algorithm 2 always produces a more accurate result, algorithm 1 presents a better compromise between CPU time and accuracy.

SPICE simulations were performed on all the logic gates. The results showed that the transistor sizes have the dominant effect on both the propagation delay and the output transit time while the input transit time has the least effect. It was also found that the input transit time has a diminishing effect on the delay as it increases for the NAND and NOR gates. Another observation is that the propagation delay can be increased even if the sizes of the transistors in a NOR gate are increased.

The model equations for the three waveform representations were generated using the two algorithms. Results showed that the model equations for the exponential

waveform representation have the least number of terms and yet the highest accuracy among the three representations. The three sets of equations were compared with SPICE using chains of gates. The ramp functions waveform representation was found to be the best in term of accuracy among the representations. Theoretically, the piecewise linear waveform representation should be better. But it has the drawback that the extra independent variable in the model makes it much harder to develop the original polynomial, and the polynomial is longer than the polynomials for the other two representations. This directly affects the accuracy of the models. This also increases the CPU time for generating the final equations. Moreover, more CPU time is required by the SPICE simulations since there is one more variable to be swept.

Examining the output at each stage of the chain of gates, it was found that the output transit time of the chain is independent from the input transit time. Only the output transit time of the first few stages are affected by the input transit time. This complements the observation by Wu[23].

7.2 Contributions

The contributions of this thesis to the field of timing modeling are two fold. Firstly, it examines the behaviour of CMOS logic gates under various conditions and looks at how the independent variables and the input vectors affect the performance of the logic gates. Secondly, it provides a detailed examination of the problem space for developing a timing model. Areas, such as gate topology, input vectors, loading and waveform representation and curve fitting have been examined. Two new algorithms were developed to systematically remove terms from a polynomial to generate the model equations. A timing model was developed using these two algorithms. From the experimental result, the timing model is very accurate when compared with SPICE - less than 10% error for the

propagation delay with chains of 10 gates. The model is very fast - five orders of magnitude smaller than the time required by SPICE. From the experimental result, the model took about 1ms to complete the computation while SPICE took 1 or 2 min.

7.3 Future Work

With the present status of the model, there is still some room for improvement. At present form, the model uses the same set of equations for both rising and falling output. The worst case is used to develop the equations if the output is not symmetrical. This is one of the factors which contributes to the high error for the output transit time in Table 6.4. For more accurate results, each type of output should have its own set of model equations.

The two algorithms developed for polynomial simplification are primitive. They can be further investigated to improve their performance. For example, the 'importance' of a term is not the best indicator for removing a term. If a better indicator can be found, the number of 'bound back' can be reduced or eliminated.

Furthermore, the model could be extended to include two new classes of gates. The first class is clocked circuits such as transmission gates and clocked logic gates, which are widely used in VLSI circuits. The problem with these gates is the clock signal. For these gates, three signals - the input, the clock and the complement of the clock - control the output which makes the task of developing the definition of propagation delay and the output rise/fall time for these gates more difficult. The second class of circuits is the CMOS complex gate which are complementary series-parallel combinations of P-type and N-type transistors.

Appendix A Curve Fitting

This appendix shows how the subroutines in IMSL[25] solve the linear least square problem which is based on the method described in [8]. Theory from [10] is also used.

A.1 The Householder's Transformation

A n by n matrix P of the form

$$P = I - 2 \frac{\hat{v}\hat{v}^T}{\hat{v}^T\hat{v}} \quad (\text{A.1})$$

is called a Householder's reflection. $\hat{v}$ is a n -vector (an vector consisted of n elements) and is called a Householder's vector.

The product, $P\hat{x}$, is the reflection of $\hat{x}$ in the hyperplane defined by all the linear combination of $\hat{v}$. $\hat{v}^T\hat{v}$ is the dot product of $\hat{v}$ itself, i.e. $\hat{v} \bullet \hat{v}$. $\hat{v}\hat{v}^T$ is the outer product of $\hat{v}$ and $\hat{v}^T$. For example, if $\hat{v} = \begin{bmatrix} a & b \end{bmatrix}^T$, then

$$\hat{v}\hat{v}^T = \begin{bmatrix} a \\ b \end{bmatrix} \begin{bmatrix} a & b \end{bmatrix} = \begin{bmatrix} a^2 & ab \\ ab & b^2 \end{bmatrix} \quad (\text{A.2})$$

Since both I and $\hat{v}\hat{v}^T$ are symmetric, P is also symmetric. Moreover, it can be proved that P is also orthogonal.

In the problem of the least square fit, the transformation of interest is the one that will zero all the components of $\hat{x}$, except the first one, x_1 . Mathematically, for a non-zero n -vector, $\hat{x} = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \end{bmatrix}^T$;

$$P\hat{x} = \begin{bmatrix} c & 0 & \cdots & 0 \end{bmatrix}^T \quad (\text{A.3})$$

It turns out that if

$$\hat{v} = \begin{bmatrix} x_1 \pm \|\hat{x}\|_2 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{then} \quad P\hat{x} = \begin{bmatrix} \mp\|\hat{x}\|_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (\text{A.4})$$

where $\|\hat{x}\|_2$ is the l_2 norm of the vector $\hat{x}$. The sign of $\|\hat{x}\|_2$ in equation (A.4) is chosen to be the same sign of x_1 . This will ensure that $\|\hat{v}\|_2$ will be bigger than $\|\hat{x}\|_2$. Hence, it will be less likely to run into numerical problems. Therefore, $\hat{v}$ can be written as

$$\hat{v} = \begin{bmatrix} x_1 + \text{sgn}(x_1) \|\hat{x}\|_2 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (\text{A.5})$$

A.2 The QR Factorization

The QR factorization of an m by n matrix A is given by

$$A = QR \quad (\text{A.6})$$

where Q is a m by n orthogonal matrix and R is a m by n upper triangular matrix.

Repeatedly applying Householder's transformation to A , one can evaluate Q and R . For example, let A be a 4 by 4 matrix. After the first application of the Householder's transformation, it becomes

$$P_1 A = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ 0 & b_{22} & b_{23} & b_{24} \\ 0 & b_{32} & b_{33} & b_{34} \\ 0 & b_{42} & b_{43} & b_{44} \end{bmatrix} \quad \text{where} \quad P_1 \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{bmatrix} = \begin{bmatrix} b_{11} \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (\text{A.7})$$

The transformation affects every element in matrix A . Therefore, b_{ij} is different from a_{ij} . After the second application, it becomes

$$P_2 P_1 A = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ 0 & c_{22} & c_{23} & c_{24} \\ 0 & 0 & c_{33} & c_{34} \\ 0 & 0 & c_{43} & c_{44} \end{bmatrix} \text{ where } P_2 = \begin{bmatrix} I_1 & 0 \\ 0 & \tilde{P}_2 \end{bmatrix} \text{ and } \tilde{P}_2 \begin{bmatrix} b_{22} \\ b_{32} \\ b_{42} \end{bmatrix} = \begin{bmatrix} c_{22} \\ 0 \\ 0 \end{bmatrix} \quad (\text{A.8})$$

After the third and the last application of the Householder's transformation, the upper triangular matrix, R , is obtained.

$$P_3 P_2 P_1 A = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ 0 & d_{22} & d_{23} & d_{24} \\ 0 & 0 & d_{33} & d_{34} \\ 0 & 0 & 0 & d_{44} \end{bmatrix} = R \quad (\text{A.9})$$

$$\text{where } P_3 = \begin{bmatrix} I_2 & 0 \\ 0 & \tilde{P}_3 \end{bmatrix} \text{ and } \tilde{P}_3 \begin{bmatrix} c_{33} \\ c_{43} \end{bmatrix} = \begin{bmatrix} d_{33} \\ 0 \end{bmatrix}.$$

From equations (A.6) and (A.9), Q can be evaluated from

$$\begin{aligned} Q^T &= P_3 P_2 P_1 \\ Q &= (P_3 P_2 P_1)^T \end{aligned} \quad (\text{A.10})$$

A.3 The Least Square Fit

Equation (4.2) shows the linear fitting function for one data point. To evaluate all those c_i , more data points are required. So equation (4.2) can be expressed in matrix form as

$$F(\hat{x}) = \Phi(\hat{x}) \hat{c} \quad (\text{A.11})$$

where $\Phi(\hat{x})$ is a m by n matrix representing $\phi_i(\hat{x})$ at different values of $\hat{x}$. $\hat{c}$ is the n -vector representing the coefficients, c_i , in (4.2).

In order to have a valid solution, the number of data points, m , must be greater than or equal to the number of the coefficient, n . The objective of the least square fit is to find $\hat{c}$ to minimize

$$\|\Phi\hat{c} - \hat{b}\|_2 \quad (\text{A.12})$$

where $\hat{b}$ is a m -vector representing the data points.

There is an important characteristic of the l_2 norm that transforms the fitting problem from an optimization problem into a linear algebraic problem. It is the fact that the l_2 norm is persevered under the orthogonal transformation. Hence,

$$\|\Phi\hat{c} - \hat{b}\|_2 = \|(Q^T\Phi)\hat{c} - (Q^T\hat{b})\|_2 \quad (\text{A.13})$$

where Q is a m by m orthogonal matrix.

So, QR factorization can be applied to Φ and it becomes

$$Q^T\Phi = R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \quad (\text{A.14})$$

where R is a m by n upper triangular matrix and R_1 is a n by n upper triangular matrix.

Let $Q^T\hat{b} = \begin{bmatrix} \hat{d} \\ \hat{e} \end{bmatrix}$ where $\hat{d}$ is a n -vector and $\hat{e}$ is a $(m-n)$ -vector. Then from (A.13),

$$\|\Phi\hat{c} - \hat{b}\|_2^2 = \|(Q^T\Phi)\hat{c} - (Q^T\hat{b})\|_2^2 = \|R_1\hat{c} + \hat{d}\|_2^2 + \|\hat{e}\|_2^2 \quad (\text{A.15})$$

Since $\|\hat{e}\|_2^2$ will always be greater than or equal to zero, the minimum of (A.15) occurs when

$$\|R_1\hat{c} + \hat{d}\|_2^2 = 0 \quad \text{or} \quad R_1\hat{c} = -\hat{d} \quad (\text{A.16})$$

Therefore, the least square fit can be obtained by applying QR factorization to Φ and then solving a set of linear equations.

Appendix B Test Circuits

This appendix describes the test circuits in Table 6.3. It shows the connections of the gates in blif format, the size of the transistors, the output loading and the input rising time. All input are rising.

B.1 Test circuit in Ex. 1

The blif file and the size of NMOS transistors in the gate are as follow:

```
.model ex1
.inputs a4
.outputs a2a4
.gate (a)' a=a2a4_not O=a2a4          12000    2
.gate (a+b)' a=0 b=mt28 O=a2a4_not    24000    1
.gate (ab)' b=1 a=mt26 O=mt28         10000    2
.gate (a+b)' a=0 b=mt23 O=mt26         5000    1
.gate (a+b+c)' a=mt19 b=0 c=0 O=mt23   20000    2
.gate (abc)' a=1 c=1 b=a4 O=mt19       24500    2
.end
```

The sizes of the NMOS transistors are in nanometer. The integer following the size specifies the type of the gate: 1 means that the gate is specified by equation (3.1) and 2 means that the gate is specified by equations (3.2) and (3.3), depending on the types of the gates.

The output loading is 54.14fF and the input rise time is 1ns.

B.2 Test circuit in Ex. 2

The blif file and the size of NMOS transistors in the gates are as follow:

```
.model ex2
.inputs a4
.outputs a2a5
.gate (a)' a=a2a5_not O=a2a5           12500      1
.gate (a+b)' a=mt142 b=0 O=a2a5_not    24000      2
.gate (a+b)' a=mt140 b=0 O=mt142       11000      2
.gate (a)' a=mt137 O=mt137_not         25000      1
.gate (a+b)' a=0 b=mt137_not O=mt140    2000       2
.gate (a)' a=mt126 O=mt126_not         24000      2
.gate (abcd)' a=1 d=1 b=mt126_not c=1 O=mt137 30000      1
.gate (a+b+c+d)' a=mt118 b=0 c=0 d=0 O=mt126 4000       1
.gate (ab)' a=1 b=mt107 O=mt118        19000      2
.gate (a+b)' a=mt93 b=0 O=mt107        13300      1
.gate (a+b)' a=a4 b=0 O=mt93           9000       2
.end
```

The notation are the same as in Ex. 1. The output loading is 124.14fF and the input rise time is 300ps.

B.3 Test circuit in Ex. 3

The blif file and the size of NMOS transistors in the gates are as follow:

```
.model ex. 3
.inputs a4
.outputs a2a5
.gate (abc)' c=1 b=1 a=a43 O=a2a5          9000      1
.gate (a+b)' a=a99 b=0 O=a43              24000     1
.gate (abc)' c=1 a=1 b=a46 O=a99          10900     2
.gate (a+b+d)' a=a88 b=0 d=0 O=a46        5000      1
.gate (abcd)' a=1 b=1 c=1 d=a4b8 O=a88    23000     2
.gate (a+b)' a=a6h8 b=0 O=a4b8           24000     1
.gate (a)' a=a4b44 O=a6h8                30000     1
.gate (a+b+c)' a=a4c5 b=0 c=0 O=a4b44    12000     2
.gate (abc)' b=1 c=1 a=a4dd O=a4c5       19000     2
.gate (a+b)' a=a4 b=0 O=a4dd             20300     1
.end
```

The notation are the same as in Ex. 1. The output loading is 14.14fF and the input rise time is 2.4ns.

Reference

- [1] L. Brocco, S. McCormick and J. Allen, "Macromodeling CMOS Circuits for Timing Simulation," *IEEE Transactions on Computer-Aided Design*, vol. 7, No. 12, pp. 1237-1249, Dec 1988.
- [2] P. Chan and K. Karplus, "Computing Signal Delay in General RC Networks by Tree/Link Partitioning," *IEEE Transactions on Computer-Aided Design*, vol. 9, No. 8, pp. 898-902, Aug 1990.
- [3] P. Chan and M. Schlag, "Bounds on signal delay in RC mesh networks," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 581-589, June 1990.
- [4] H. Chen and S. Dutta, "A timing model for static CMOS gates," *IEEE Trans ICCAD*, pp. 72-75, 1990.
- [5] S. Chung, "Simulation Techniques for VLSI Circuits," *Proceedings of IEEE ASIC Conference and Exhibit 1990*, pp. T8.1-T8.6, 1990
- [6] A. Deng, "Piecewise-linear timing delay modeling for digital CMOS circuits," *IEEE Trans. Circuit and Systems*, vol. CAS-35, pp. 1330-1334, Oct. 1988.
- [7] A. Deng and Y. Shiau, "Generic linear RC delay modeling for digital CMOS circuits," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 367-376, April 1990.
- [8] J. Dongarra, J. Bunch, C. Moler and G. Stewart, *LINPACK Users' Guide*, SIAM, Philadelphia, 1979.
- [9] W. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Applied Physics*, vol. 19, no. 1, pp. 55-63, Jan. 1948.
- [10] G. Golub and C. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, 1989.
- [11] P. Hallam, T. Pritchard and G. Townsend, "Performance Macromodelling and Optimization of Regular VLSI Structures," *Proc. EDAC 91*.
- [12] N. Hedenstierna and K. O. Jeppson, "CMOS circuit speed and buffer optimization," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 270-281, Mar. 1987.
- [13] K. Hedlund, "Aesop: A tool for automated transistor sizing," *Proc. 24th ACM/IEEE Design Automation Conference*, pp. 114-120, 1987.
- [14] B. Hoppe, G. Neuendorf, D. Schmitt-Landsiedel and W. Specks, "Optimization of high-speed CMOS logic circuits with analytical models for signal delay, chip area, and dynamic power dissipation," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 236-247, Mar. 1990.

- [15] S. Lin and M. Marek-Sadowska, "An Accurate and Efficient Delay Model for CMOS Gates in Switch-Level Timing Analysis," *Proceedings of International Symposium on Circuits and Systems 1990*, pp. 856-860, 1990.
- [16] M. D. Maston and L. A. Glasser, "Macromodeling and optimization of digital MOS VLSI circuit," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 659-678, Oct. 1986.
- [17] S. Nassif and S. Director, "WASIM: A Waveform Based Simulator for VLSICs," *IEEE International Conference on Computer-Aided Design 1985*, pp. 29-31, 1985.
- [18] F. Obermeier and R. Katz, "An electrical optimizer that consider physical layout," *Proc. 25th ACM/IEEE Design Automation Conference*, pp. 453-459, June 1988.
- [19] J. K. Ousterhout, "A switch level timing verifier for digital MOS VLSI," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 336-349, July 1985.
- [20] T. Sakurai and A. Newton, "Delay Analysis of Series-Connected MOSFET Circuits," *IEEE Journal of Solid-State Circuits*, vol. 26, No. 2, pp. 122-131, Feb 1991.
- [21] J. Shyu, J. Fishburn, A. Dunlop and A. Sangiovanni-Vincentelli, "Optimization-Based Transistor Sizing," *IEEE Custom Integrated Circuits Conference*, pp. 417-420, 1987.
- [22] S. Vermuru and A. Thorbjornsen, "A Model for Delay Evaluation of a CMOS Inverter," *Proceedings of International Symposium on Circuits and Systems 1990*, pp. 89-92, 1990.
- [23] C. Wu and M. Shiau, "Efficient Physical Timing Models for CMOS AND-OR-Inverter and OR-AND-Inverter Gates and Their Applications," *IEEE Transactions on Computer-Aided Design*, vol. 9, No. 9, pp. 1002-1009, Sep 1990.
- [24] *Berkeley Logic Interchange Format (BLIF)*, University of California Berkeley, 1990
- [25] *IMSL Math/Library User's Manual*, IMSL Inc., 1991.

END

2|3·1 2·92

FIN