1

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.

Bounded Timing Models for Digital Gates

by

W. Walter Fergusson, B.A.Sc.

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

Master of Engineering

Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Electronics
Carleton University

© copyright

W. Walter Fergusson

1989

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 0-315-54369-8

Canada

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

“Bounded Timing Models for Digital Gates”

submitted by W. Walter Fergusson, B.A.Sc.

in partial fulfilment of the requirements

for the degree of

Master of Engineering

Thesis Supervisor

Chairman, Department of Electronics

Carleton University

May 26, 1989

Abstract

To increase the accuracy and reliability of gate level simulators and timing analyzers a new gate level timing model is introduced.

It takes into account

- process variations as well as variations in supply voltage and device temperature,
- correlation between gate delays within a chip,
- uncertainty in capacitive loading and delays, and
- effects of unideal switching thresholds and signal slew rates.

The gate level timing model was extended to provide an absolute bound on timing values.

Contents

1	Desired Properties in a Gate Level Model	1
1.1	Thesis Objectives	2
1.2	The Broader Problem	2
1.3	Desired Functional Properties of a Digital Gate	3
1.4	Desired Digital Gate Timing Properties	3
1.4.1	Physical Parameters	4
1.4.2	Electrical Properties	5
1.5	Uncertainty	6
1.5.1	Process Parameters	6
1.5.2	Environmental Parameters	6
1.5.3	Circuit Parameters	7
1.5.4	Transition Time and Thresholds	7
1.5.5	Bounds of Behavior	7
1.6	Summary of Desired Gate Level Properties	8
1.7	Summary of Thesis	8
2	Problems with Current Models and Simulators	10
2.1	Modeling Inaccuracies	11
2.1.1	Delay Multipliers	11
2.1.2	Correlation	14

2.1.3	Effects of Input Slew Rate	16
2.1.4	Effects of Switching Thresholds	16
2.1.5	Uncertainty in Delay	18
2.1.6	Summary	18
2.2	Gate Level Simulator Problems	20
2.2.1	Three Gate Level Simulators	20
2.2.2	FUNSIM	21
2.2.3	Best/Worst Case Simulation	21
2.3	Gate Level Simulators with Spread	22
2.3.1	The Concept	22
2.3.2	Spread Simulators Without Common Ambiguity Resolution	23
2.3.3	Spread Simulators with Common Ambiguity Resolution	25
2.4	Desired Properties of Gate Models and Simulators	28
3	The Digital Gate Timing Model	30
3.1	Die Parameter	30
3.2	Timing Range	31
3.3	Timing Parameters	31
3.3.1	Incorporation of Output Load Capacitance	35
3.3.2	Incorporation of Input Transition Times	35
3.4	The Specific Model	38
3.4.1	The Raw Die Parameter Function	38
3.4.2	The Trapezium	40
3.4.3	The Parameterized Trapezium	41
4	Characterization of the Gate Level Timing Model	42
4.1	Characterization of the Die Parameter	42
4.1.1	A Large Problem Made Smaller	42

4.1.2	Example Die Parameter Function	43
4.1.3	How the Characterization was Performed	44
4.1.4	Results of Die Parameter Characterization	50
4.2	Characterizing the Individual Gates	51
4.2.1	Delay Path	51
4.2.2	Fitting the Parameterized Trapezium	53
4.2.3	Computational Effort	54
4.3	Results	54
4.3.1	Typical Results	54
4.3.2	Other Technologies	57
4.3.3	Summary	57
5	The Bounded Timing Model	60
5.1	Effects of Process and Environment	60
5.1.1	Bounding With One Parameter	61
5.1.2	Bounding With Many Parameters	64
5.1.3	Complexity of the Bounding Parameters	66
5.2	Effects of Circuit Parameters	66
5.2.1	Complexity of Bounding Circuit Parameters	68
5.3	Increasing Bounding Accuracy	68
5.3.1	Subdividing Parameter Ranges	68
5.3.2	Subdivisions Effect on Characterization Effort	71
5.4	Summary of Bounded Timing Model	73
6	Model Usage and Future Research	74
6.1	The Simulator	74
6.1.1	Set-up	74
6.1.2	Timing Simulation	75
6.1.3	Post Processing	76

6.1.4	Expected Results	77
6.2	The Timing Analyzer	78
6.2.1	The Use of Simulators	78
6.2.2	Timing Analysis versus Simulation	79
6.2.3	The Timing Analyzer	80
6.3	Hierarchical Timing Analysis	84
6.3.1	Macro Models	84
6.4	Functional Analysis	85
6.4.1	Simulators	85
6.4.2	Hierarchy	85
6.5	Timing Analysis in Design Synthesis	86
6.6	Unsolved problems	86
6.6.1	Multiple clocks	86
6.6.2	Asynchronous clocks	87
6.7	Timing information in data	87
7	Conclusion	88
7.1	Contributions in this Thesis	89
7.1.1	Die Parameter	89
7.1.2	Provable Bound With Correlation	89
7.1.3	Uncertainty in Circuit Parameters	90
7.1.4	Timing Verification	90
7.2	Summary	90
A	Solving Constraint Equations	91
A.0.1	Optimization Theorem	92
A.0.2	Results	93
B	Glossary	94

List of Tables

2.1	Gate Delays and Delay Multipliers	13
4.1	Minimum, Nominal and Maximum Values for Environmental, Circuit and Process Parameters	44
4.2	Derived Constants of the Raw Die Parameter Function	51
4.3	NOR3: Intrinsic Delay, Rising Input	56
4.4	NOR3: Transition Time, Rising Input	56
4.5	NOR3: Transition Time, Falling Input	56

List of Figures

2.1	Optimistic Results Produced by a Compromise Delay Multiplier	12
2.2	Optimistic Results Produced by a “Worst” Case Delay Multiplier	15
2.3	Effects of Input Slew Rate on Gate Delay	17
2.4	Effects of Switching Thresholds on Gate Delay	19
2.5	Ambiguous Timing in a Clock Tree	24
2.6	Ambiguity Resolution of a Clock Tree	26
2.7	Ambiguous Timing Verses Timing Correlation	27
3.1	Mapping Process and Environmental Parameters into the Die Parameter	32
3.2	Gate Level Timing Parameters as a Trapezium	33
3.3	Timing Parameters of a Digital Gate	34
3.4	The Trapezium as a Function of Output Loading	36
3.5	A Four Dimensional View of a Gate’s Timing Parameter . . .	37
3.6	The Relationship Between Transition Time and Intrinsic Delay	39
4.1	Improving the Die Parameter	45
4.2	Relative Trapezium Area as a Measure of Uncertainty	47
4.3	Finding the Bounding Trapezium	49
4.4	Examples of Delay Paths Through Digital Gates	52

- 4.5 Typical Characterization Results 55
- 5.1 Bounding the Effects of One Parameter on Gate Timing . . . 62
- 5.2 Refining the Bound on Gate Timing Caused by Variations in
One Parameter 63
- 5.3 Bounding the Effects of Two Parameters on a Gate's Timing 65
- 5.4 Bounding the Effects of the Load Parameter on a Gate's Timing 67
- 5.5 Bounding Gate's Timing Without Subdividing Parameter
Ranges 69
- 5.6 Bounding Gate's Timing Using Subdivided Parameter Ranges 70
- 5.7 Bounding Gate's Timing Using Global Bounds on Parameters 72

Chapter 1

Desired Properties in a Gate Level Model

Most of present day digital design is concentrated at the gate level. A design, done with logic gates, can be laid out automatically using computer aided design tools. The characteristics of each type of gate can be determined by analog simulations, independently of the circuit in which they will be placed. The interaction between interconnected gates can often be treated as the delay of charging a single equivalent capacitor. This capacitor can be estimated once the layout is known.

To insure their design works correctly, designers use digital simulators to emulate the behavior of their design. The accuracy of the simulator is a function of how well the gate level models used by the simulator emulate the physical devices, how accurately they treat the interaction between gates, and particularly how well they can handle variations which occur within gates of the same type due to processing variations. Also, the gate level model needs to handle environmental variations such as temperature and supply voltage.

1.1 Thesis Objectives

This thesis aims to develop a digital gate level timing model which will allow for exhaustive analysis of a circuits timing. To to this the model must accurately emulate affects of process and environmental variations, as well as the interaction between gates. Also, the model will allow for the uncertainty in measuring or estimating most physical parameters that affect the timing behavior of the digital gate. In particular, the gate level timing model will overcome weaknesses in existing simulators in modeling affects of slew rates and timing correlation within a chip. To remain usable, the time taken to simulate a design with this model must be within the same order of magnitude as the time taken to simulate the design with current gate level simulators.

1.2 The Broader Problem

This work grew out of a problem involved in the simulation of large systems comprised of VLSI chips. The idea is to break down the design into functional blocks which can be analyzed for timing faults. If a block has no timing faults the block's function can be abstracted to a behavioral level model which simulates much faster than a circuit composed of gates. To do an exhaustive analysis of a block's timing a new gate level timing model was needed. This thesis aim is to develop this model.

In chapter 6 the use of this model in solving this broad problem will be addressed. The integration of this model into timing analyzers and simulators will be explored as well as other uses for the bounded timing model.

1.3 Desired Functional Properties of a Digital Gate

When the inputs of a gate are presented with valid logic levels, the gate should produce the correct logic level at the output. Thus the logic model is sufficient to describe the functional properties of a digital gate.

1.4 Desired Digital Gate Timing Properties

In a combinatorial gate, the important timing properties are associated with the time it takes the output to change logic levels in response to an input change. These are often expressed in terms of *propagation delay*, *rise time*, and *fall time*.

For clocked gates such as flip-flops, data has to remain stable within a timing window around the clock edge to permit reliable operation. This leads to other timing properties such as *set-up* and *hold times*.

While the functional properties of a gate are easy to describe accurately the timing properties of a gate are not. This is because the timing properties of a gate depend on

- environmental factors such as temperature and supply voltage,
- process variations such as gate length and oxide thickness, and
- circuit factors such as capacitive output loading.

1.4.1 Physical Parameters

The following is an account of a few *parameters*¹ (a parameter is a value of a property) that have an affect on a gate's timing.

Process Parameters

When a silicon chip is made, it goes through many process steps. These steps can affect the physical properties of the digital gates. These physical properties are called the *process parameters*. Examples of these are gate length, oxide thickness, gate-drain overlap, etc.

These *process parameters* in turn affect the electrical properties of the transistors that make up the gate. These in turn affect the timing properties of the gate.

Environmental Parameters

There are some properties of the environment that can affect the behavior of the chip or gate. When these properties are quantified they are referred to as *environmental parameters*. The two most common *environmental parameters* that could affect the timing of a digital gate are the temperature of the chip and the supply voltage.

Circuit Parameters

The circuit in which the gate is a component, can affect the timing behavior of the gate. *Circuit parameters* affect specific instances of gates. One *circuit parameter* that has a large affect on the gate's timing is the capacitive load

¹Words in italics can be found in the glossary starting on page 94. Because some of these words have non-standard definitions they will remain in italics throughout this thesis.

on the output of the gate. Another *circuit parameter* is the *signal transition time* at the gate's input.

1.4.2 Electrical Properties

Transition Times

Signal transition time is the time it takes the signal to go from one voltage level to another. The affects of *transition time* are most noticeable in technologies with low gain.

The delay of a CMOS logic gate is due mainly to the time it take the gate to charge the output node. Thus the output does not switch suddenly at some time after the gate input changes, but starts to react almost instantly to the input changes. However, it may take a long time to charge the output signal to the correct voltage. Thus the *transition time* is a major component of a CMOS gate's delay.

The actual propagation delay caused by *transition time* is influenced by another property, the *switching threshold*, which will now be described.

Switching Threshold

The *switching threshold* is the value of the input voltage at which the output voltage will begin to change from one logic state to another. For CMOS gates which have a low logic level of 0V and a high logic level of V_{DD} , the ideal *switching threshold* is $V_{DD}/2$. However very few CMOS gates are designed to have a *switching threshold* of $V_{DD}/2$. This is because the CMOS gates can be made smaller with an off-center switching threshold and a change in the *switching threshold* does not affect the logic function of the gate.

Many gate level models assume that a gate's *switching threshold* is $V_{DD}/2$. For a gate with a *switching threshold* higher than $V_{DD}/2$, a rising input does

not cause the gate to switch until the input voltage reaches this threshold. If the *transition time* is long, then the signal will take longer to reach the elevated *switching threshold* and the gate will take longer to switch. If the input is falling, it reaches the elevated *switching threshold* before it reaches $V_{DD}/2$. Thus the gate will switch sooner than expected.

To model the timing of a gate properly, the electrical properties of the *switching threshold* level and *transition time* must be taken into account.

1.5 Uncertainty

If all the electrical properties and physical parameters could be measured or controlled accurately a highly accurate gate model could be constructed. A method of calculating the timing parameters of a gate could be developed and the accuracy of the model would only depend on how well one modeled each affect and how well each parameter could be measured. Unfortunately most physical parameters and electrical properties cannot be measured accurately. These will now be described.

1.5.1 Process Parameters

When one is designing a chip, one does not know the exact *process parameters* of the chip that will be manufactured. Hopefully, the process is well characterized and monitored so the bounds of *process parameters* are limited. However, this bounded uncertainty in the *process parameters* will result in a bounded uncertainty in the timing parameters of a gate.

1.5.2 Environmental Parameters

The *environmental parameters* (like voltage and temperature) of the chip are not known until the device is in use. The ranges for the *environmental*

parameters are usually large and result in a large uncertainty in the gate timing parameters.

Fortunately, the *environmental parameters* are given minimum and maximum values that are set by system requirements. The chip designer does not have to concern himself with parameters that fall out of this range.

1.5.3 Circuit Parameters

During early parts of the design stage, the physical layout of the design is not available. Since this has a major affect on the load capacitance which has a major affect on the timing of the gate only approximations to the load capacitance can be used.

Fortunately, in the later stages of the design, when the chip is laid out, the *circuit parameters* can be gauged more accurately. However this does not mean that the timing effects can be accurately determined since load capacitance can be heavily affected by *process parameters*. Thus the uncertain output load causes an uncertainty in gate timing values.

1.5.4 Transition Time and Thresholds

Since these particular properties are affected by *process* and *circuit parameters* they too are uncertain.

1.5.5 Bounds of Behavior

Fortunately, uncertainty caused by variations in *process parameters*, *environmental parameters*, *circuit parameters*, and *switching thresholds* can be bounded. The silicon foundry guarantees that chips delivered to the designer will fall within specific *process parameter* boundaries. Likewise the system designs will include bounds on the *environmental parameters*.

The *circuit parameters* can be measured or calculated when the masks are generated but before the chip is produced. Thus they can be bounded by photolithography and process variations.

With all these bounds a gate level timing model should be able to give bounds on the timing of a gate.

1.6 Summary of Desired Gate Level Properties

Apart from the exact representation of the functional properties of a gate level model provided by Boolean equations, the following properties of a gate level model are desired.

- Gate timing must take into account process variations.
- Gate timing must take into account environmental variations.
- Gate timing must take into account circuit variations. Specifically, capacitive loading of signals and input signal *transition times*.
- Gate timing is uncertain and can be bounded.

A few other properties will be added to this list once some problems with current simulators and gate level models are discussed.

1.7 Summary of Thesis

In this chapter some properties of digital gates were discussed, and a list of desired properties of digital gate level timing models was presented. In the following chapter current gate level models will be examined. The problems with these models will be discussed along with some problems encountered

by simulators using these models. The list of desired properties of digital gates will be expanded based on the examination of these problems.

The new gate level timing model will then be introduced. Affects of process and environmental variations will be modeled by the composite *die parameter*. The modeling of uncertainty with this *die parameter* follows. The model will then be extended to take care of circuit parameter variations.

A particular set of gates will then be characterized. The characterization process will complete the new gate level model. It will then be evaluated against the desired properties of a gate level model.

The model will then be extended to permit a provable bound on any timing parameter of the gate model.

Finally the usage of the model will be discussed along with future research directions using this model.

Chapter 2

Problems with Current Models and Simulators

One of the major sources of inaccuracy in modern gate level simulators is their simplistic model of a gate delay. The two major areas where model flaws are introduced are in the use of a *delay multiplier* and not allowing for gate delay *correlation*. Other minor flaws are the affects of unideal input *switching thresholds*, finite input signal *transition time*, and uncertainty of output loading. These will be discussed in this chapter.

Although there are problems with all the gate level models referenced in this section, this does not mean that they are without value. A simple model with flaws is more beneficial than no model at all. In fact the simplest gate level timing model cited in this thesis[LSI85] has been used to design thousands of working VLSI chips.

2.1 Modeling Inaccuracies

2.1.1 Delay Multipliers

An example of a standard gate level model is one used by LSI Logic[LSI85]. It only incorporates nominal delay values. These are obtained from a circuit simulation (for example SPICE) of the gate at the nominal electrical parameters resulting from nominal process and *environmental parameters*.

Since a process is not always nominal, one has to take into account the affects of the expected process variations on the delay of the gate. Also one has to take into account the environmental factors such as supply voltage and temperature. To take all of these variations into account, the gates being characterized are simulated at nominal values to get a nominal delay and at the extreme ranges of these values to get a minimum and a maximum delay. A minimum *delay multiplier* is a number that multiplies the nominal delay to get the minimum delay. A maximum *delay multiplier* is a number that multiplies the nominal delay to get the maximum delay. These delays are then examined and usually two *delay multipliers* are chosen to be assigned to the entire set of gates.

A problem arises when a single pair of multipliers are chosen to be representative of all of the individual gate *delay multipliers*. If the largest maximum *delay multiplier* over all types of gates is chosen, the timing will be unrealistically pessimistic. If these extreme values are replaced by more optimistic ones, a designer relying on the gate delay models might be misled into using a gate configuration which would simulate satisfactorily but would not function in some environments.

Consider figure 2.1 on page 12. Because the *dclay multiplier* has to apply to all gates, a pessimistic value of the *delay multipliers* is usually chosen. For example, consider three gates with delays described in table 2.1 on page 13.

Figure 2.1: If a *delay multiplier* underestimates the delay through a single type of gate, then it can underestimate the delay of a critical path that is comprised of these gates.

Example Gate Delays and Multipliers						
Gate	Delay in ns			Multiplier		
	max	nom	min	max	min	max*min
NAND2	2.89	1.70	1.11	1.7	.65	1.11
MUX2	2.89	1.52	0.84	1.9	.55	1.05
DFF	8.11	4.61	2.72	1.8	.59	1.06
NOR2	5.56	2.06	0.82	2.7	.40	1.08

Table 2.1: These figures show how gate delays cannot be represented by a single *delay multiplier*. Any one *delay multiplier* will underestimate or overestimate at least one gate delay.

To be pessimistic one would choose a minimum *delay multiplier* of 0.35 and a maximum *delay multiplier* of 2.7. This would clearly overestimate the delay of the NAND2 and MUX2 gates. So as a compromise a minimum *delay multiplier* of 0.5 and a maximum *delay multiplier* of 2.0 could be chosen. Choosing these values could be justified by the fact that only a few of the gates characterized exceed these values. (In this case only the NOR2 gate's extreme values would exceed these chosen *delay multipliers*).

However by reaching this compromise, potential problems may arise. A conservative designer who uses a lot of NOR2 gates could find that the manufactured chips do not always work at the speed the chip was designed for. This is because the designer simulated his device at nominal values and then doubled the delays to simulate at what was thought to be the maximum delay values. The actual values for the maximum delay could range to 2.7 times the nominal delay.

This example shows that the common use of *delay multipliers* in model-

ing the gate delays of a circuit or chip could lead to circuits or chips that will not meet the original design specification even though the simulated design meet all the design specifications.

Even if the conservative *delay multiplier* of 2.7 was used, the chip simulation may not catch a *timing fault*. (A *timing fault* or *timing violation* occurs when a timing specification of a gate is violated.) In figure 2.2 on page 15 there is a circuit that has no *timing violations* for nominal delays. There are no predicted *timing violations* for maximum delays if the maximum *delay multiplier* of 2.7 is used, however there is a *timing violation* if the actual maximum delays are used.

Thus neither the compromise maximum *delay multiplier* of 2.0 nor the “worst” case *delay multiplier* of 2.7 can catch all *timing violations*.

The gate level timing model used by Texas Instruments[TI86] for their 2- μ m CMOS standard cells is an example of a gate model that has individual delay multipliers. Typical, maximum and sometimes minimum timing values are specified for all gates.

2.1.2 Correlation

Two delays are said to be *correlated* when there is an explicit or implicit relationship between the delay of one gate and the delay of a second gate. This is implicit in the gate level models that use delay multipliers. The ratio between delays of a NAND2 and a NOR2 gate remain constant. Since their structure is different it is hard to believe that this ratio could remain constant. The figures in table 2.1 on page 13 show it does not.

However if one examines the product of the two *delay multipliers* for each gate it is almost constant. This means that while the ratios of two gate delays are not constant they are very close. Thus one can assume that the gate delays are correlated. It would also be realistic to assume that the

Figure 2.2: If the maximum *delay multiplier* was chosen to be the largest of all gate's individual *delay multipliers*, then there are circuits that would not work even though the simulator catches no *timing violations*. In the nominal case, the data arrives 2.19ns after the clock. With the maximum *delay multiplier* the data arrives 5.92ns after the clock. With actual minimum delays, the data arrives 0.12ns before the clock and will cause the circuit to malfunction.

correlation is subject to some uncertainty.

The three gate level models[LSI85][TI86][NCR85] referenced in this thesis do not address the *correlation* problem.

2.1.3 Effects of Input Slew Rate

Very few gate models take into account the affect of input *slew rate* on a gate's delay. This is normally only done by switch or circuit level models and simulators.

To understand why it is important to include input *slew rates* of gates in a model, consider a gate that is lightly loaded and driven by a signal that has a very long rise time. This is shown in figure 2.3 on page 17. In this case the gate responds to the input as if the input were a D.C. signal. The time the output signal uses to swing from one end of its range to the other depends on the time it takes for the input to go from V_{ILmax} to V_{IHmin} . If the gain of the gate is small then $V_{IHmin} - V_{ILmax}$ is large and the gate may take a long time to switch.

One can see the time for the output to swing from one logic level to another is then strongly dependent on the input signal *slew rate*. To be accurate a gate level model must take into account varying input *slew rates*. Most existing models do not take into account the affects of input *slew rate*[TI86][NCR85][LSI85].

2.1.4 Effects of Switching Thresholds

Inaccuracy also enters most current gate level delay models because they ignore non-ideal *switching thresholds*. An ideal *switching threshold* is 50% of V_{DD} for CMOS. However, since gates are digital in nature there is very little incentive to make sure that the *switching threshold* is at $V_{DD}/2$ especially

Figure 2.3: An excessive input *transition time* can produce a longer output *transition time* than an input step. Thus one should take into account the affects of input *transition time* on output *transition time*.

since this would result in gates taking up more area on a chip.

If the *switching threshold* is not at 50% then finite rise and fall times can cause inaccuracy in the predicted delay. For example if a gate has an input *switching threshold* at 40% and an input signal rises in 10ns then the input will cause the gate to start switching after 4ns. If the signal with the same *slew rate* falls the input will cause the gate to start switching after 6ns. This is shown in figure 2.4 on page 19. The assumption of 50% input *switching threshold* will cause a gate to start switching after 5ns for each case.

Note that the error introduced here is roughly 20%. This is a large error to accept.

Note that this effect could only be included into the model if signal *slew rates* were included in the model. The three comparison gate level models[TI86][NCR85][LSI85] do not support varying *slew rates*.

2.1.5 Uncertainty in Delay

Uncertainty of Track Capacitance

Uncertainty in gate delay often comes from the the unknown capacitance of metal or polysilicon wire. This arises from the uncertainty in the thickness of gate oxide or field oxide in MOS technologies. Since the thickness of the field oxide is not critical, die to die variations in interconnect capacitance is often $\pm 15\%$ of the nominal capacitance[Gre86].

2.1.6 Summary

- *Correlation* between gate delays exists, but is seldom modeled.
- Input signal *slew rate* does have an affect on gate delays.

Figure 2.4: A CMOS gate does not switch until the input signal is near the gates switching threshold. Since the *switching threshold* is not always at $V_{DD}/2$ gates can take longer to react to inputs falling than inputs rising. This longer reaction time is also a function of input *slew rate*.

- The assumption of a 50% *switching threshold* leads to model inaccuracy.
- Large uncertainties caused by unknown capacitive load are seldom modeled.

2.2 Gate Level Simulator Problems

The simulator, *timing verifier* or analyzer have a great effect on the usefulness of a gate level timing model. This is because the simulator can use, ignore or enhance the accuracy of a gate level model to predict the behavior of a circuit.

Thus along with examining the problems of gate level models one must examine the problems of gate level simulators.

2.2.1 Three Gate Level Simulators

For examples three gate level simulators were chosen. The two commercial simulators are “VERILOG”[VER89] and “P/C SILOS”[PCS88]. The “FUNSIM”[FUN89] gate level simulator is the third simulator. It is a proprietary simulator used by Bell-Northern Research.

VERILOG

The VERILOG simulator is a simulator that allows one to simulate at varying degrees of model abstraction. The gate level simulator section of VERILOG supports independent rise and fall delays but does not directly support models with minimum and maximum delays. It also does not support delay multipliers.

This is an example of a very simple gate level simulator. This does not mean that the VERILOG simulator is not useful. It is designed to model circuits at levels other than the gate level.

P/C SILOS

The P/C SILOS simulator is a gate level simulator. It supports independent rise and fall time delays. Delay multipliers can be assigned to individual gate types. It also supports a linear increase in delay time with output load. It does not support uncertainty or ambiguity resolution and it can not support input transition times.

2.2.2 FUNSIM

The FUNSIM simulator is used mainly as a gate level simulator. It supports independent rise and fall times. It supports gate models with minimum, typical and maximum values along with *delay multipliers*. It supports non-linear output loading affects on timing delays. It also supports uncertainty as well as common ambiguity resolution. It does not support correlation as mentioned in section 2.1.2, nor does it support input transition times.

2.2.3 Best/Worst Case Simulation

As mentioned before, an ASIC designer is concerned about all operating cases for his chip. To allow for all cases a designer usually simulates a *slow* and a *fast* chip. (A *slow* chip corresponds to gates with maximum delays while a *fast* chip corresponds to gates with minimum delays.) These are often called “best” and “worst” case simulation even though the “best” (read *fast*) chip may not work while the “worst” (read *slow*) chip will work. To avoid this confusion the terms *slow* and *fast* will be used when referring

to the overall speed of a chip.

Although *fast* and *slow* chip simulations can catch many *timing faults*, they cannot catch all timing errors. This is because there are an infinite number of cases between the *fast* and *slow* chip cases. Currently one can also simulate at nominal delay values and hope that there are no *timing faults* for intermediate cases.

A good designer can be reasonably assured of this on a small design. However less experienced designers are being put in charge of designing chips that could not fit on a board 10 years ago. The mistakes that the designer makes adds to his experience but may cause the chip to fail.

2.3 Gate Level Simulators with Spread

2.3.1 The Concept

Some gate level simulators can deal with inaccuracy by allowing uncertainty in delay times[FUN89]. These are called *spread simulators*. Time in a *spread simulator* is usually described by a minimum and a maximum time or a mean and deviation time.

The *spread simulators* introduce the uncertainty into timing values which can vary from chip to chip. All possible variations of parameters are then modeled by this uncertainty. They do this by keeping track of more than one time value. (For example: minimum and maximum time or nominal and deviation time.) With an experienced designer, guesses can be made of the affects of differing *delay multipliers* as well as the affects of varying input *slew rates* and output loads. Although most of these affects are not quantified, they could be. This would allow a simulator that addresses all of the complaints about model inaccuracies mentioned before. However this simulator would not be that useful to do exhaustive case analysis.

To ensure proper operation of a device, one needs to simulate the affects of all parameter variations. Unfortunately, the resulting large uncertainty soon overwhelms the simulator's ability to predict delays.

2.3.2 Spread Simulators Without Common Ambiguity Resolution

To see how the uncertainty can overwhelm the ability of a simulator to predict delays consider the clock tree in figure 2.5 on page 24. The inverters have a nominal delay of 2.0ns, a *slow* chip delay of 4.0ns and a *fast* chip delay time of 1.0ns. If a clock distribution tree is 5 levels deep the delay from input to output is 5-20ns. If two flip-flops were being clocked by the same clock in a shift register and a flip-flop delay ranges from 2.0-8.0ns then the data to the second flip-flop could arrive anywhere from 7ns (= 5ns clock + 2ns flip-flop) to 28ns (= 20ns clock + 8ns flip-flop) after the input clock. The clock to the second input arrives from 5-20ns after the input clock. For a significant portion of time (from 7ns to 20ns) the data can arrive at the same time as the clock and produce a *set-up* or *hold* violation. In reality this does not happen because both flip-flops are triggered by their common clock while the uncertainty was built up from tracing the paths back to the input clock.

Thus without considering the common *delay path* and the common ambiguity of the delay uncertainty this simulator produces extremely pessimistic results.

Fortunately many new gate level simulators trace delays back towards the source and resolve this ambiguity. This is called *common ambiguity resolution*. This solves some, but not all problems dealing with uncertainty.

Figure 2.5: Uncertainty adds up as delay elements are encountered. Yet a signal transition can only occur at an instance in time. If two separate elements refer to the same signal and do not take this fact into account, the uncertainty is added and not resolved.

2.3.3 Spread Simulators with Common Ambiguity Resolution

A *spread simulator* would not be as pessimistic if it used *common ambiguity resolution*. However each timing check required would involve checking for common ambiguity. This is because a common ambiguity check does not eliminate all ambiguous paths, but references all delays to a specific signal under consideration. See figure 2.6 on page 26 for an example. If signal *c* was used as a reference, the ambiguity between signals *a* and *b* is not resolved. In other words an ambiguity resolution can only be applied to one timing check at a time. A different timing check would require a new ambiguity resolution. So this would increase the time needed to simulate the chip, but would provide a more realistic simulation of the chip. The FUNSIM simulator[FUN89] can perform *common ambiguity resolution*.

However since chips are getting bigger, larger clock trees are needed, and flip-flops are not always clocked by the same signal. Thus ambiguity problems still arise.

Ambiguity Resolution Without Timing Correlation

One may ask if a particular timing fault is caused by uncertainty in real delay and not caused by uncertainty in simulator spread. To demonstrate this point consider two flip-flops, the output of one feeding the data input of the other. This time their clock inputs are not the same signal but each is driven by an inverter and these inverters are being driven by one signal. The data path delay from *c'* is through 1 inverter and 1 flip-flop. The delay is 3.0-12.0ns. The clock to the second flip-flop goes through an inverter with a delay of 1.0-4.0ns. There is still an overlap from 3-4ns and therefore a *timing violation* is flagged. This is shown in figure 2.7 on page 27.

Figure 2.6: To resolve ambiguity a signal transition is assumed to take place at a specific time. All signal times are then referenced to this signal by tracing delay paths backwards and forwards from this reference signal.

Figure 2.7: Ambiguity resolution works by pinning down a signal and referring all other signals to this one. Correlation on the other hand uses the fact that if one element has a specific delay then all other delays can be approximated because all elements are on the same wafer and have nearly the same physical properties. Thus ambiguity resolution involved picking a reference signal while *correlation* involves knowing that the things that affect one gates delay will also affect other gate delays on the same die.

To be realistic, this *timing violation* will never occur on an ASIC. This is because the delay of the inverter and flip-flop in the data path are set to 1.0ns and 2.0ns, while the inverter's delay in the clock path is 4.0ns. Because these gates are on the same chip, they have nearly identical process variations as well as applied temperature and voltage. It would be unrealistic to expect one inverter to have a delay of 1.0ns and a second inverter to have a delay of 4.0ns. This is shown for the *fast* and *slow* chips in figure 2.7 on page 27.

What is needed is to take into account the fact that gates within a chip are in the same environment and the delays are correlated by the process and *environmental parameters*. However, gates simulated with correlated timing values still need *common ambiguity resolution* to achieve good results.

2.4 Desired Properties of Gate Models and Simulators

The desired timing properties of digital gate models and digital gate simulators are as follows.

- Gate timing model must take into account process, environmental and circuit variations. (ie. affects of gate oxide thickness, voltage, inter-track capacitance and the like.)
- Gate timing must take into account circuit variations. Specifically: capacitive loading of signals and input signal *transition times*.
- Gate timing models must incorporate timing *correlation* between gates on a single chip.
- Gate timing model should be able to represent and bound uncertainty of gate delays.

- Also, from the examination of simulators, the simulator, *timing verifier* or *timing analyzer* must be able to resolve common ambiguity.

Chapter 3

The Digital Gate Timing Model

In this chapter a new gate level model is proposed. It can handle the *correlation*, common ambiguity, differing *slew rates* and other factors that are needed to perform a robust simulation of a VLSI design.

3.1 Die Parameter

The *correlation* between gates in the same chip will be done using a *die parameter*. This *die parameter* will represent all process, and environment parameters. It will do this by mapping the process and environment parameter spaces into the *die parameter* space by the *die parameter* function. Further the extremes of the *die parameter* will be 0 and 1. The 0 will correspond to a fast chip and a 1 will correspond to a slow chip.

All timing parameters of the gate model will be linear functions of this *die parameter*. Therefore to examine the design for all possible process and environmental parameters all one has to do is to examine the results for the

die parameter ranging from 0 to 1. See figure 3.1 on page 32.

3.2 Timing Range

To take into account uncertainty, all values used by the simulator will be extended into ranges. Thus the relationship between a timing parameter and the *die parameter* will be able to bound the timing parameter. Timing parameters will come in pairs. An upper bound line will correspond to a maximum timing value and a lower bound line will corresponds to a minimum timing value. The area bounded by these lines is called a *trapezium*. (A *trapezium* is a four sided figure with two parallel sides. These sides correspond to *die parameter* equal to 0 or 1.) A picture of this is in figure 3.2 on page 33.

3.3 Timing Parameters

The actual timing parameters measured are *intrinsic gate delay* for a rising and a falling input and the *transition time* for the output signal. The *intrinsic gate delay* is measured from the 10% mark of the input signal swing to the 10% mark of the output signal swing. The *transition time* for the output signal is measured from the 10% mark of the output signal swing to the 90% mark of the output signal swing. These values are shown in figure 3.3 on page 34. These timing measurements were used in the STAT! simulator[Ben87a].

At first glance the definition of *intrinsic delay* does not make sense. This is because the 10% trigger for measuring does not cause the output to change. This is corrected by the fact that the *trapeziums* are not fixed for a gate. The *trapeziums* are functions of *circuit parameters*. In this case the

Figure 3.1: The *die parameter* represents the collective affect of all process and *environmental parameters* on the timing of a gate. The *die parameter* function is a function which maps the process and *environmental parameters* into the *die parameter*. The *die parameter* ranges from 0 (a fast chip) to 1 (a slow chip).

Figure 3.2: Gate timing parameters are functions of the *die parameter* and are bounded by lines. The shape composed of four lines, two of which are parallel, is called a *trapezium*.

Figure 3.3: There are two basic timing parameters used in the gate level model. The first is the *intrinsic delay*. It is measured from 10% of the input range to 10% of the output range. The *transition time* is the time the output takes to go from 10% of its range to 90% of its range.

corners of the *trapeziums* are a function of the output load capacitance and the input signal *transition time*.

3.3.1 Incorporation of Output Load Capacitance

As mentioned, the position and shape of the *trapezium* is a function of output capacitive loading. A picture of this is in figure 3.4 on page 36.

As in most gate models the timing parameters are a linear function of capacitive load. In this particular case the upper and lower bounds on the timing parameter are planes extending into the 3rd dimension as shown in figure 3.4.

To handle the uncertainty in capacitive load values, two sets of planes are generated for each particular instance of gate. One set of planes corresponds to the lowest expected capacitive load, and the other set of planes corresponds to the highest expected capacitive load. Since the model has a linear relationship between timing values and capacitive load the lower bounding plane for the timing parameter is the lower bounding plane for the minimum capacitance loading while the upper bounding plane is the one that corresponds to the maximum capacitive loading. Thus a bound on the timing value is given for a range of capacitive loading.

3.3.2 Incorporation of Input Transition Times

Input *transition times* (or *slew rates*) are incorporated into the model in the same way capacitive loading was. The figures would look much like the ones in figure 3.4 on page 36. Figure 3.5 on page 37 is an attempt to show this relationship. When these two *circuit parameters* are included together the resulting *trapezium* would be expanded to four dimensions.

It is the incorporation of input *transition times* that allow one to make

Figure 3.4: The *trapezium* in figure 3.2 on page 33 is only for one value of output loading. To include the affects of output loading with the *trapezium* it is extended in another dimension. This dimension is output loading. The extended *trapezium* is called the *parameterized trapezium*.

Figure 3.5: The timing value of a gate is a function of the *die parameter* which represents all process and *environmental parameters*. The timing value is bounded by a *trapezium*, which is a function of the *circuit parameters* (output capacitance and input transition time).

sense of the unusual definition of *intrinsic delay*. Refer to figure 3.6[Ben87a, p. 1046] on page 39 in the following explanation.

Consider the affects of the input *transition time* on the gate's *intrinsic delay*. The nonstandard definition of the *intrinsic delay* is time between the input getting to 10% of its input swing to the output getting to 10% its swing. This time can be broken up into two sections. The first is the time the input takes to get to the *switching threshold* of the gate and the second is the time it takes the gate to respond to the input at the *switching threshold*. The second part of the *intrinsic delay* is called the *inertial delay*.

Since the *intrinsic delay* is a function of the *transition time* the *intrinsic delay* is essentially calculated as the sum of the *inertial delay* and the time taken to get to the switching threshold. Also since each gate is characterized individually and since both rising and falling inputs are considered, the model's *intrinsic delay* includes information about the *switching threshold* of the gate.

3.4 The Specific Model

In this section the specific mathematical formulation of the model is discussed.

To allow for examples and specific details of a gate's timing, a $3\mu m$ CMOS cell library used at Carleton University for Canadian Microelectronics Corporation chips will be characterized.

3.4.1 The Raw Die Parameter Function

The *raw die parameter* is an empirical function of process and *environmental parameters*. An example of the *raw die parameter* function is of the form

$$\text{raw die parameter} = (1 + \alpha_G * (G - G_n) + \beta_G * (G - G_n)^2)$$

Figure 3.6: The time taken for a gate to react to an input (its *intrinsic delay*) is composed of two parts. The first part is the time it takes for the input signal to reach the gates *switching threshold*. This is nearly a linear function of the input *transition time*. The second part is the time taken to charge up internal and output nodes to produce a noticeable reaction to the input signal reaching the *switching threshold*. This is the *inertial delay*.

$$\begin{aligned} & \ast(1 + \alpha_T \ast (T - T_n) + \beta_T \ast (T - T_n)^2) \\ & \ast(1 + \alpha_V \ast (V - V_n) + \beta_V \ast (V - V_n)^2). \end{aligned} \quad (3.1)$$

- T is the temperature in $^{\circ}C$. It ranges from a minimum of $0^{\circ}C$ to a maximum of $80^{\circ}C$, with a nominal value of $T_n = 25^{\circ}C$.
- V is the supply voltage in Volts. It ranges from a minimum value of $4.5V$ to a maximum value of $5.5V$, with a nominal value of $V_n = 5.0V$.
- G is the gate length in μm . The gate length ranges from $2.0\mu m$ to $4.0\mu m$, with a nominal value of $G_n = 3.0\mu m$.

This function is an empirical approximation to data from a CMOS gate array handbook[LSI85, pp. 2-2 to 2-3] and a CMOS standard cell handbook[TI86, p1.7]. A linear approximation was considered, but quadratic terms were added to increase the accuracy of the approximation.

This raw *dic parameter* is then scaled to the range $[0,1]$. This scaled version is called the *dic parameter*. It exists for all gates subject to the same process and environmental factors.

3.4.2 The Trapezium

The *trapezium* is characterized by its corners. Since the x coordinate of the corners is known the y coordinates only have to be given.

The corners are:

- Lower Left: $(0, ll)$.
- Lower Right: $(1, lr)$.
- Upper Left: $(0, ul)$.
- Upper Right: $(1, ur)$.

3.4.3 The Parameterized Trapezium

As mentioned earlier, the *trapezium* is actually a function of output load and input *transition time*. The formulation of the parameterized *trapezium* is as follows.

- Lower Left: $\alpha_{ll} + \beta_{ll} * L + \gamma_{ll} * S + \delta_{ll} * L * S$
- Lower Right: $\alpha_{lr} + \beta_{lr} * L + \gamma_{lr} * S + \delta_{lr} * L * S$
- Upper Left: $\alpha_{ul} + \beta_{ul} * L + \gamma_{ul} * S + \delta_{ul} * L * S$
- Upper Right: $\alpha_{ur} + \beta_{ur} * L + \gamma_{ur} * S + \delta_{ur} * L * S$

where

- L stands for the capacitive output load in Farads and
- S is the input *transition time* (slew) in seconds.

The coefficients are determined during characterization.

The linear load term approximates affects of capacitive loading which is incorporated into most gate level timing models[LSI85][TI86][NCR85]. The linear *transition time* term takes into account affects of *transition times* and *switching thresholds* on gate delays. The quadratic term in load and slew is to catch any unexpected effects in timing caused by the interaction between output loading and input *transition time*.

Chapter 4

Characterization of the Gate Level Timing Model

The characterization of a set of gates require two steps. The first step is the characterization of the *die parameter* function. The second step is the characterization of the individual gates.

4.1 Characterization of the Die Parameter

4.1.1 A Large Problem Made Smaller

Ideally the *die parameter* would be characterized by simulating each gate under many combinations of environmental and process variations. Then all this information would be used to pick a form for the *die parameter* function that would minimize the uncertainty or spread in the timing.

In practice this is too large a problem and only a representative logic gate is simulated. If there were 2 environmental factors and 3 process factors and 2 circuit factors, 10 gates and they were simulated at minimum, nominal, and maximum values for their parameters then $10 * 3^{(2+3+2)}$ or 21870 simulation

runs would be required to extract timing parameters. With each simulation taking approximately one minute on a SUN 3/60 workstation this might take half a month. With each gate having 4 to 24 timing parameters this would correspond to approximately 250000 points. Then a non-linear curve fit for all constants in the trial *die parameter* function would be done. This might take months or years.

To reduce the size of the problem only one gate is simulated with all the different process and environmental conditions.

4.1.2 Example Die Parameter Function

To characterize the *raw die parameter* function NAND2 gate was chosen to be simulated. The only *process parameter* chosen was gate length. Both voltage and temperature were used as *environmental parameters*. A 3 μ m CMOS process was characterized.

The empirical *raw die parameter* function was formed by observing that the parameters chosen are independent and affect the total delay of the chip as a multiplicative factor. A trial function (with a thought of scaling it to the range of 0 to 1 after the characterization) was based on nominal values returning a value of 1.0[LSI85, pp. 2-2 to 2-3]. Each parameter's deviation from the nominal value was approximated by a quadratic function. To repeat equation 3.1 on page 40, the trial *raw die parameter* function is:

$$\begin{aligned} \text{raw die parameter} = & (1 + \alpha_G * (G - G_n) + \beta_G * (G - G_n)^2) \\ & * (1 + \alpha_T * (T - T_n) + \beta_T * (T - T_n)^2) \\ & * (1 + \alpha_V * (V - V_n) + \beta_V * (V - V_n)^2). \quad (4.1) \end{aligned}$$

Parameters used for Characterization			
Parameter	minimum	nominal	maximum
Gate Length	$2.0\mu m$	$3.0\mu m$	$4.0\mu m$
Supply Voltage	4.5V	5.0V	5.5V
Temperature	$0^{\circ}C$	$25^{\circ}C$	$80^{\circ}C$
Capacitive Load	1 unit	3 unit	10 unit
Transition Time	1ns	3ns	10ns

Table 4.1: The gates are simulated with minimum, nominal and maximum values for *environmental*, *circuit*, and *process parameters*. The simulator used is HSPICE. One unit load ($0.16pF$) is defined to be the nominal input capacitance of a two input NAND gate.

4.1.3 How the Characterization was Performed

The NAND2 gate was simulated with minimum, nominal and maximum values of gate length, supply voltage, chip temperature, output loading and input *transition time* rate. The output capacitive load is measured in unit loads. A unit load is the approximate input capacitance of a NAND2 gate input. The value used here is $0.16pF$. All possible combinations were simulated. This resulted in $3^5 = 243$ simulation runs. The parameters values making up the 243 simulation runs are shown in table 4.1 on page 44.

To optimize the values for $\alpha_G, \beta_G, \alpha_T, \beta_T, \alpha_V$, and β_V the sum of the *relative errors* for the *trapezium* was chosen as the function to minimize. Figure 4.1 on page 45 gives an idea on how the process proceeds. Four ideas are important here.

- The relative area is a measure of relative uncertainty.
- The timing values from the simulation are fixed. The characterization

Figure 4.1: A nonlinear optimization package is used to optimize the constants in the *raw die parameter* function. A visual representation of this process is to slide the simulation points horizontally. This is because the timing value of a simulation point is fixed, but the *raw die parameter* changes when the constants in the *raw die parameter* function are changed by the optimization package.

process involves warping the X axis. (ie. sliding the points horizontally).

- The *trapeziums* can be found by linear programming.
- The X axis will be mapped into the range 0 to 1.

The Sum of the Relative Areas

The relative area can be thought of as a measure of relative uncertainty. It can be taken as the area of the *trapezium* ($AREA(C)$ in figure 4.2) divided by the area under the average value of the timing parameter being measured ($AREA(A) = AREA(B) = (a + b + c + d)/4$). Figure 4.2 on page 47 is a diagram of this.

Note that the X axis has been mapped into its final 0 to 1 line segment. For characterization of the *die parameter* one never needs to find this mapping. Only when the final version of the *die parameter* is required does this mapping need to be used.

Thus taking the corners a, b, c, d as numerical values then the relative area is:

$$\frac{1}{2}((a - b) + (c - d))\frac{1}{e} \quad (4.2)$$

or

$$2 \frac{(a - b) + (c - d)}{a + b + c + d} \quad (4.3)$$

where

$$e = \frac{(a + b)/2 + (c + d)/2}{2}. \quad (4.4)$$

The relative area is a measure of relative uncertainty. It makes much more sense to use relative uncertainty rather than absolute uncertainty to measure timing values. For example a 1ns absolute uncertainty is tolerable if the timing value was 10ns but it would be intolerable if the timing value

Figure 4.2: How relative uncertainty can be approximated by the ratio of areas. The $AREA(A)$ is the absolute area, while $AREA(C)/AREA(A)$ is the average area.

was 0.5ns. Thus a relative uncertainty is a better cost function to use when optimizing.

Characterizing the Die Parameter

Because the *die parameter* is graphed on the X axis, the progression of the optimization has some unusual features. Changing the *raw die parameter* constants slides the simulation points horizontally. When this is done, a *trapezium* of minimum area is formed around the points and the relative area was calculated. The steps are repeated using a number of different values for the coefficients until a suitable fit is found. These steps are internal to the “amoeba” non-linear optimization routine[Pre88, pp. 305-309].

Finding the Trapeziums

The desired result is to minimize the relative area bounded by these *trapezium*. However to implement the optimization efficiently the area of each of the *trapezium* was minimized. This can be done by minimizing the sum $a + c$ subject to the upper bound constraint:

$$a + \frac{x - x_{min}}{x_{max} - x_{min}}(c - a) > y, \quad (4.5)$$

and to maximize the sum of $b + d$ subject to the lower bound constraint:

$$b + \frac{x - x_{min}}{x_{max} - x_{min}}(d - b) < y \quad (4.6)$$

for all points (x, y) . Here x represents the *raw die parameter* and y represents the timing value measured. The values of x_{min} and x_{max} represent the absolute minimum and maximum values that the *raw die parameter* can assume for the current *raw die parameter* function.

Figure 4.3: The simulation points are to be bounded by lines. To bound the minimum area, it is easy to see that all one has to do is to minimize the area under the line $\overline{ac}$ and maximize the area under line $\overline{bd}$.

In these equations x, y, x_{min} , and x_{max} are known. Thus a and c occur linearly in the constraints and the cost function. Therefore one can use linear programming (Simplex method [Pre88, pp. 329-343]).

Mapping the Die Parameter Function

After a *raw die parameter* function was determined, a simple linear mapping from the minimum and maximum values of the *raw die parameter* function to the range 0 to 1 was performed. This created the *die parameter* function used to correlate delays.

The *die parameter* function was also checked to ensure it was a monotonic function of gate length, supply voltage and temperature. This is done by making sure that $\partial(\text{die parameter})/\partial(\text{parameter}) > 0$ for all combinations of parameter values. The monotonic property of the *die parameter* function is needed to get a bound on the model uncertainty as explained in section 5.1.

4.1.4 Results of Die Parameter Characterization

Using to the *raw die parameter* function (equation 3.1) the results of the characterization are in table 4.2 on page 51.

The influence is a measure of the dominance of each term in the raw parameter function. The influence is the maximum absolute error that is caused by ignoring the term in the *raw die parameter* function. The value of 0.54 for α_G represents a 54% error would be caused by ignoring this term. One can see that the β_T term could be ignored since the relative error introduced would be less than 0.05%.

Raw Die Parameter Constants		
Constant	Value	Influence
α_G	0.54	0.54
β_G	0.013	0.013
α_T	$4.4 * 10^{-3}$	0.24
β_T	$1.4 * 10^{-7}$	0.0004
α_V	-0.26	0.13
β_V	0.070	0.02

Table 4.2: These are the constants found by minimizing the area of the *trapeziums*. The influence is the maximum absolute change to the nominal value of the *raw die parameter* function caused by the term containing the constant.

4.2 Characterizing the Individual Gates

There are two major timing values that are needed for each gate. They are the *intrinsic delay* and the output *transition time*. Each of these values has to be found for each *delay path* through the gate.

4.2.1 Delay Path

A *delay path* is the path from an input to an output than can cause a logic change at the output. Figure 4.4 on page 52 gives an example of the *delay paths* for a 2 input NAND gate and a 2:1 multiplexor.

Each of these paths are simulated under the same conditions that were used in simulating the NAND2 gate for finding the *die parameter* function. These conditions are summarized in table 4.1 on page 44.

Figure 4.4: A logic path is a path from an input to an output whose timing can be represented by a single timing parameter. The signal transition direction as well as the state of other inputs do define different *delay paths*.

4.2.2 Fitting the Parameterized Trapezium

Fitting the parameterized *trapezium* is a linear programming problem just as fitting the *trapezium* was when the *die parameter* was being characterized.

To repeat, the following equations are used as the upper and lower bounds.

The constraints to find the upper bounding plane are:

$$a + \frac{x - x_{\min}}{x_{\max} - x_{\min}}(c - a) > y. \quad (4.7)$$

where

$$a = \alpha_a + \beta_a L + \gamma_a S + \delta_a SL, \quad (4.8)$$

and

$$c = \alpha_c + \beta_c L + \gamma_c S + \delta_c SL \quad (4.9)$$

for each point (x, y) . As before, x represents the *die parameter*, while y is the timing value.

The function to be minimized is

$$\sum_{(x,y)} (y_u(x, L, S) - y)/y \quad (4.10)$$

where

- y is the timing parameter,
- x is the *die parameter*,
- L is the capacitive load on the output,
- S is the *transition time* of the input, and
- $y_u(x, L, S)$ is the value of the upper bound plane for the values of x, L , and S . ($y_u = a(1 - x) + cx$.)

This function is essentially the sum of the relative distance of the bounding plane from the actual simulated point.

In an analogous fashion, the lower bounding plane involves maximizing

$$\sum_{(x,y)} (y_l(x, L, S) - y)/y \quad (4.11)$$

subject to

$$b + \frac{(x - x_{min})}{x_{max} - x_{min}}(d - b) < y \quad (4.12)$$

$$b = \alpha_b + \beta_b L + \gamma_b S + \delta_b SL \quad (4.13)$$

$$d = \alpha_d + \beta_d L + \gamma_d S + \delta_d SL, \quad (4.14)$$

where $y_l(x, L, S)$ represents the lower bound plane.

The parameterized *trapezium* is a four dimensional figure. An example of the parameterized *trapezium* without input *transition time* dimension is given in figure 3.4 on page 36.

4.2.3 Computational Effort

The major effort in computing the parameterized *trapeziums* used to characterize the gates are simulating each gate for all combinations of parameters. Fortunately this effort is only needed once, it can be automated, and it is suitable for overnight runs. For example, the 2 input NAND gate was simulated $3^5 = 243$ times using HSPICE on a SUN 3/60 workstation. This took 6 hours of elapsed time.

The rest of the characterization of the NAND2 gate took less than 5 minutes of CPU time on a SUN 3/60.

4.3 Results

4.3.1 Typical Results

Figure 4.5: Here is a sample of the results of characterization of a NOR3 gate. (Note that this gate was not used to characterize the *die parameter* function, yet there is a strong *correlation* with the *die parameter* function.) The plots are (from the upper left counting clockwise):

- Intrinsic time for a falling output; one unit load, 1ns input transition time.
- Transition time for a rising output; one unit load, 1ns input transition time.
- Transition time for a rising output; ten unit loads, 3ns input transition time.
- Transition time for a falling output; one unit load, 1ns input transition time.

NOR3: Intrinsic Delay, Rising Input				
corner	α $\times 10^{-9}$	β $\times 10^3$	γ $\times 1$	δ $\times 10^{12}$
ll	.230	.119	.153	.075
lr	.498	.348	.364	.103
ul	.230	.046	.295	.088
ur	.440	.304	.568	.132

Table 4.3: NOR3: Intrinsic Delay, Rising Input

NOR3: Transition Time, Rising Input				
corner	α $\times 10^{-9}$	β $\times 10^3$	γ $\times 1$	δ $\times 10^{12}$
ll	.218	.857	.232	.009
lr	.956	3.896	.197	
ul	.231	.906	.274	
ur	1.054	4.717	.224	

Table 4.4: NOR3: Transition Time, Rising Input

NOR3: Transition Time, Falling Input				
corner	α $\times 10^{-9}$	β $\times 10^3$	γ $\times 1$	δ $\times 10^{12}$
ll	1.487	6.661		
lr	6.688	27.133		
ul	1.558	7.078	0	
ur	6.990	27.893		

Table 4.5: NOR3: Transition Time, Falling Input

Typical results are shown in figure 4.5 on page 55 and in table 4.3, table 4.4, and table 4.5 starting on page 56. This NOR3 gate was simulated for all conditions specified in table 4.1 on page 44. It is obvious that the *die parameter* is very strongly correlated to timing values due to the small uncertainty associated with each timing value. The NOR3 gate was the least *correlated* of all gates characterized. In the tabular results, some values were zero. This is because some timing values are not affected by *circuit parameters*. In particular the output *transition time* is often not affected by the input *transition time* and the *intrinsic delay* is sometimes not affected by the output loading.

4.3.2 Other Technologies

This gate level timing model was tailored to the CMOS technology, however there are no obvious problems associated with using this model for TTL or ECL gates.

4.3.3 Summary

To evaluate the performance of the model, one has to keep in mind the goals of the model. They are restated here.

1. Gate timing model must take into account process, environmental and circuit variations. (ie. the affects of gate oxide thickness, voltage, intertrack capacitance and the like.)
2. Gate timing model must be able to represent uncertainty.
3. The gate model should not prevent the use of *common ambiguity resolution*.

4. Gate timing models must incorporate timing *correlation* between gates on a single chip.
5. Gate timing model should be able to bound the uncertainty.

To evaluate results each one of these goals will be examined.

1. The model does take into account process, environmental and circuit variations. In this particular example the process variation in gate length and the environmental variation in supply voltage and temperature were taken into account by the design of the *die parameter* function. The circuit variations (output capacitive load and input signal *transition time*) were taken into account by the parameterized *trapeziums*.
2. The uncertainty in timing values is represented as a range since all values have a lower and upper bound. These bounds are the hyperplane's of the parameterized *trapezium*.
3. Since the delays in the gate model are freely available to the outside simulator, the model does not interfere with *common ambiguity resolution*.
4. The model delays are all correlated through the *die parameter* function. For any simulation of the device, only one value of the *die parameter* is chosen. This value will correspond to a particular silicon chip. This means that all devices will be simulated at this particular value of the *die parameter* and thus will be correlated.
5. The bounds provided by the parameterized *trapezium* do provide a bound on uncertainty. Since the timing values are very well correlated with the *die parameter*, this provides a reasonable confidence that the

bounds provided by the parameterized *trapeziums* are good. Also first order approximations to timing values do not predict any discontinuities or wild swings when a process or electrical parameter is varied [Wes85, pp32-51][Wes85, pp137-141]. This gives added confidence to the bounds provided by the gate model.

However this is not an *absolute bound* but an arbitrary bound. In the next chapter, the model is extended to provide an absolute bound on uncertainty to be used to prove correct timing.

Another goal was that the model's accuracy would be of the same order as the uncertainty of measured values. For the NAND2 gate in particular the relative uncertainty of timing values is on the order of 10%. Since the capacitive load is only known to $\pm 15\%$ the model's uncertainty does not add greatly to the existing uncertainty.

Chapter 5

The Bounded Timing Model

As mentioned in the previous chapter's summary the model is very useful for design work. It is superior to other existing gate models[LSI85][TI86][NCR85], and will permit robust designs. This simple model only lacked an *absolute bound* for timing values. Although it is seldom used in design work, this chapter will describe the extension to the gate model that would permit a circuits timing to be proven correct.

5.1 Effects of Process and Environment

The bounding of the exact behavior of the device can be done by worst case tolerance analysis [Spe88]. This analysis requires that timing values are a monotonic function of the process and *environmental parameters*. In most cases, this is true. Even if it is not, there are extensions that will allow the relaxation of this restriction [Nas86].

If the *die parameter* is a monotonic function of these parameters then bounding can be done. In the specific examples that were done this was always the case. If it were not the case, the function mapping the parameters

into the *die parameter* could be modified to ensure that this is the case.

5.1.1 Bounding With One Parameter

The bounds can be established as follows. Consider two points

$$(die_0(a, b, c, d, e_0), tt_0(a, b, c, d, e_0)) \quad (5.1)$$

and

$$(die_1(a, b, c, d, e_1), tt_1(a, b, c, d, e_1)). \quad (5.2)$$

Here a , b , c , d , and e represent *process, environmental, or circuit parameters* (Circuit parameters do not actually effect the *die parameter*).

Since the *transition time* (tt) is a monotonic function of the parameter e , then for any value of e between e_0 and e_1 , tt is between tt_0 and tt_1 . The same goes for the *die parameter*. If e is between e_0 and e_1 then the *die parameter* die is between die_0 and die_1 . This describes a rectangle on the familiar timing value versus *die parameter* graph. Figure 5.1 on page 62 shows an example of this.

Note that the rectangle extends beyond the bounding planes. The first interpretation is that the bounding planes are not bounding the timing value absolutely. This is indeed the case. However, the bounding rectangle was formed by the parameter e over the range $[e_0, e_1]$ (assuming $e_0 < e_1$). If this range were more finely divided, then a set of smaller rectangles would be formed. Figure 5.2 on page 63 is an example of this. Thus for more work, one could get a smaller area of uncertainty.

To find the parameterized *trapezium* for this new bounded case all one has to do is include the corners of the rectangles not represented by simulation points with the simulated points in the formulation described in section 4.2.2.

Figure 5.1: The bounding of a timing parameter is done by introducing extra points to be bounded. These extra points (denoted by o's) when paired with the corresponding simulation points (denoted by x's) form a rectangle. This rectangle encompasses the timing behavior of the gate if the mapping is monotonic. This introduces a timing bound that is too large to use.

Figure 5.2: Here the simulation points (marked by x's) and the introduced bounding points (marked by o's) are plotted. The bounding rectangles are formed and one can see that the bound is still provable if the mapping is monotonic. Also the bounding is much tighter than if only two simulation points were used.

5.1.2 Bounding With Many Parameters

The actual model used has one process and two *environmental parameters*. Thus one has to extend the work of the previous section to more than one parameter. To simplify the explanation the model will be extended to two parameters first. A simple extension will then cover any number of parameters.

Bounding With Two Parameters

To examine how this can be extended to two parameters, the mapping from parameter space to the timing vs. *die parameter* space must be examined. This is given in figure 5.3 on page 65.

First consider the parameter space. One can see that this region is a rectangle. Each edge of the rectangle in the parameter space $\overline{AB}$, $\overline{BD}$, $\overline{CD}$, $\overline{AC}$ is mapped into a rectangle ab , bd , cd , ac in the timing vs. *die parameter* space. Because the mapping is monotonic, any point on the perimeter of the rectangle $ABDC$ must lie within the rectangles ab , bd , cd , ac . Furthermore any point X within the rectangle $ABDC$ must be mapped into a point x within the rectangles ab , bd , cd , ac or inside any region bounded by these rectangles.

Like the previous example with one parameter, the bounding *trapezium* may be formed by adding extra points to be bounded. In figure 5.3 on page 65 these points are marked by "o".

Bounding With Many Parameters

It is easy to see that the method can be extended beyond two dimensions. The rectangle in the parameter space becomes a hyper-rectangle (a rectangle of higher dimensions). Likewise in the timing vs. *die parameter* graph the rectangles become hyper-rectangles.

Figure 5.3: Extending the bounding to two parameters causes a series of bounding rectangles. New bounding points are introduced, and the characterization can proceed as before. The bound is still absolute if the mapping is monotonic.

5.1.3 Complexity of the Bounding Parameters

As noted before the parameterized bounding *trapezium* is calculated using linear programming. The only affect of adding new points is to add to the number of linear bounds that the linear programming method must check. However each new parameter added causes an exponential increase in the number of points needed to be simulated.

With one parameter, only two points were simulated and 4 points were needed to be bounded. With two parameters four points were simulated, four rectangles were produced and 16 points were needed to be bounded. Since the rectangles share the simulated points the four simulated points were counted twice. This means that fewer points have to be considered, but it will not cut down on the order of complexity of the algorithm. For three parameters, eight hyper-rectangles were produced, requiring 64 points to be bounded. The number of points to be simulated is $Order(2^N)$, and the number of points to bound is $Order(4^N)$.

5.2 Effects of Circuit Parameters

The *circuit parameters* do not have any effect on the *die parameter*. However, bounding the behavior of the gate's timing with respect to variations in the *circuit parameters* is much the same as bounding the gate's timing behavior with respect to the environmental and *process parameters*.

One can see this when referring back to figure 3.4 on page 36. The diagram shows that the timing is a function of load (and input *transition time*) just as it is a function of *die parameter*. As before one has to ensure that the timing value is a monotonic function of output capacitive load and input *transition time*.

Figure 5.4 on page 67 is an example of the bounding of the timing with

Figure 5.4: The affects of the load parameter on the absolute bounding of a gate's timing are nearly the same as for the environmental and *process parameters*. The simplification occurs since there is no intermediate mapping of the load parameter to the *die parameter*. The load parameter shows up as the X-axis. The bounding rectangle and points are introduced as before. The bounding is still absolute if the mapping is monotonic.

respect to the load parameter. It is simpler than the affects on timing of the environmental and *process parameters*. This is because there is no intermediate *die parameter* function in the case of *circuit parameters*.

5.2.1 Complexity of Bounding Circuit Parameters

Since there are only two *circuit parameters* there is no exponential explosion in the number of points to simulate or consider when bounding the affects of circuit variations on a gate's timing. However, since circuit variations in load are rather large, the uncertainty introduced by bounding the gate's behavior from one unit load to ten unit loads is so large as to make the model impractical to use. Thus a more accurate bound is needed.

5.3 Increasing Bounding Accuracy

5.3.1 Subdividing Parameter Ranges

A more accurate bound can be achieved by subdividing the parameter range that causes the pessimistic bound. In characterizing the $3\mu\text{m}$ CMOS gates, it was noticed that the gate length parameter variations affected the timing values greatly while the supply voltage parameter did not. So in this case the gate length parameter range should be subdivided into smaller regions while the supply voltage need not be. Figure 5.5 on page 69 gives an example of the accuracy that was achieved before the subdivision while figure 5.6 on page 70. shows the improvement caused by subdividing one interval. In both these examples the X-axis represents a *circuit parameters*, but it could just as easily have represented a environmental or process parameter.

Figure 5.5: Bounding circuit parameters without subdividing the circuit parameter ranges produces *absolute bounds* that are analogous to the crude bounds from minimum values to maximum values in timing. Therefore if a parameter has a major affect on a gate's timing parameter it must be broken up into subranges to ensure a reasonable bound that can be used.

Figure 5.6: By breaking parameter ranges up into subranges, a considerable improvement in the bound can be obtained while proving that the timing parameters still are bounded for any parameter variation.

5.3.2 Subdivisions Effect on Characterization Effort

Subdividing a parameter range causes an increase in the number of simulations required for characterization. If there are N parameters each requiring $S_i, i = 1 \dots N$ points to subdivide the range then the number of simulations required is $\prod_{i=1}^N S_i$. If there are a large number of parameters, or a parameter range is subdivided finely, there are too many points required for characterization to be done in a reasonable time.

To solve this problem one has to consider two trade-offs. The first is to live with a bound that is not the best *absolute bound*. Remember that the accuracy of the bound only needs to be of the same order as the uncertainty in the bounds. Also one still has a smaller range of uncertainty than normal worst case (from minimum timing to maximum timing) spread simulation and that *common ambiguity resolution* can be used to reduce the effects of this uncertainty.

The second trade off is to bound the affects on the gate's timing by one or more parameters. If one could bound the parameter's affect on timing (say to $\pm 4\%$ from nominal) then one could simulate with this parameter at nominal value and add the uncertainty on to the simulated value. This will not increase the number of points if any parameters are introduced this way. Thus for parameters with very little (but noticeable) affect no extra effort beyond the *global bounding* needs to be done. Many parameter's affects can be added on this way. (Here one makes the assumption that these parameter's affects are not greater than the sum of their parts.)

Figure 5.7 on page 72 shows what the new timing vs. *die parameter* plot would look like if some parameters were bounded in this second way. The simulation points have been converted into a range caused by simulating some parameters at nominal value. No extra points are needed to calculate

Figure 5.7: Because division of all parameter ranges is both computationally expensive and not necessary, a *global bound* on parameters with minor affects on timing is performed. All simulation points (marked by x's) have a predetermined spread introduced by simulating a minor parameter at only its nominal value. These stretched simulation points are then used as corners in the bounding rectangles. This is done while maintaining a *provable absolute bound*.

the bounding parameterized *trapezium* because all of it is the original bounding *trapezium* increased by $\pm 4\%$ depending on whether this is a lower or upper bound.

5.4 Summary of Bounded Timing Model

1. The model does take into account process, environmental and circuit variations.
2. Uncertainty is taken into account by having timing values represented as a range. This range forms an upper and lower bound on the possible timing values.
3. *Common ambiguity resolution* can be applied to this gate level model.
4. The model delays are all correlated through the *die parameter*.
5. The bounds provided by the bounding parameterized *trapezium* provide an *absolute bound* on the model's timing behavior provided
 - the *die parameter* is a monotonic function of all *process* and *environmental parameters*.
 - the effects of simulating parameters at nominal values are bounded.

In most cases, including the example used in this thesis, these assumptions are valid.

6. The model takes much more effort to characterize than the model without the absolute bounding.
7. The model will be more pessimistic than the model without absolute bounding.

Chapter 6

Model Usage and Future Research

6.1 The Simulator

The bounded timing model has not been integrated into a simulator. This section addresses the issues that need to be considered when incorporating this model into a simulator.

6.1.1 Set-up

Characterization

The characterization of the *die parameter* only needs to be done once for each process and environment expected. Once this is characterized, each gate needs to be characterized once. This involves a lot of computer time but most of it is automated and can be done overnight and on weekends. Thus one can accept this as done, and it has no impact on the actual simulator run time.

Simulator Set-up

To reduce the time it takes to simulate a device most simulators calculate all values that can be calculated before the timing simulation proper begins. This can be done with the bounded behavior simulator as well. Output load ranges and input *transition time* ranges can be fixed before simulation starts for most gates (*transition times* may not be fixed for gates in feedback loops). Therefore most delay transfer functions can be precalculated before timing simulation begins thus speeding up the rest of the simulation.

6.1.2 Timing Simulation

Standard Spread Event Driven Simulation

This simulator can proceed like most event driven simulators. Special care must be taken since simulator time is represented by two ranges of times. The first is a spread (two points) corresponding to a *dic parameter* of 0. The second is a spread corresponding to a *dic parameter* of 1. Also simulation will not proceed as quickly as a regular simulator because of the 4 fold increase in numbers to be calculated.

All Time Event Simulation

A modified event simulation may be used to allow for quicker coding of the simulator[Kat86].

The idea is that input stimuli are known for the entire length of the simulation. One can then simulate all gates that have their inputs connected to primary inputs and then know their outputs for all time. Using these new outputs (which are known for all time) other gates can be simulated for all time. This process can be continued for all gates that are not in a feedback loop or depend on outputs from feedback loops.

For gates in feedback loops the inputs are only known for a subset of the entire simulation time. The outputs can be calculated for this time and sent to other gates in the feedback loop to increase the time their inputs are known. Because gates have positive (non-zero) delays the feedback loops can be simulated in this manner.

Signals can store their values for all times and can be queried for these values during post processing.

This method was suggested in a paper by Dr. Ghosh[Gho86].

6.1.3 Post Processing

Timing Violations

Because all signals are stored, *timing violations* can be found by scanning the signal events. Since the signal is known for all time and for all possible chip speeds (and therefore all contracted cases of operation) a check can be done for any *timing violations*. *Set-up*, *hold*, or minimum pulse width violations can be checked.

Common Ambiguity Resolution

Common ambiguity resolution cannot easily be incorporated into a simulator. It is much easier to handle it as a post simulator.

Since the *timing violations* flagged might not be true violations, they are then sent to the common ambiguity resolver. Since only these violations, (and other paths specified for the user) are checked, the ambiguity resolver should not contribute grossly the overall time to check out a circuit. What is gained is that less information is returned to the designer, and thus he can spend his time working on the real *timing violations* instead of sorting through a list that contains many false violations.

6.1.4 Expected Results

There are no obvious impediments to incorporating either version of the bounded timing model into a simulator. The problem is only made difficult by the complexity of simulator that needs to support the relatively complex gate level timing model.

Improved Robustness

The simulator will give results that are more realistic than existing gate level simulators. This is because it accurately models affects of input *slew rates* and threshold levels on gate delays. It also gives more realistic results because uncertainty of device operation is included in the gate models and the simulator. The uncertainty is bounded and therefore the simulator can give a clear pass or fail to the circuit's timing. A great benefit is that when it says there are no violations, there are none. The simulator also generates few false violations due to *common ambiguity resolution*.

Improved Case Analysis

The simulator is also more robust than simulators that do not use spread simulation because it can check out all variations of environment for the chip in one pass by examining for timing violation for all values of the *die parameter*. The simulator is more robust than typical simulators with spread because typical simulators with spread can be overwhelmed by uncertainty caused by exhaustive case analysis. There is no nagging worry about missing a special case. Because of this if the simulator finds no problem, there will be no timing problems on the chip! The certainty of this information is very useful to an ASIC designer.

Disadvantages

The absolute bounded delay model's disadvantage is that it may fail chips that might otherwise pass. It also takes much longer to characterize than the simpler model characterized in chapter 4. Thus in choosing between the two models presented, one is choosing between a model that may be over pessimistic to one that might be a little optimistic.

6.2 The Timing Analyzer

Although this model can be used in simulators it was conceived as a part of a *timing analyzer* to solve the broader problem mentioned in section 1.2. What follows is some of the reasoning that went into choosing a *timing analyzer* over a simulator.

The bounded timing model is not presently incorporated into a *timing analyzer*.

6.2.1 The Use of Simulators

Simulators are used by an ASIC designer to check the behavior of the design. The ASIC designer must then take the information from the simulator and check the design's behavior against the specifications of the design. The emphasis here is that the designer is designing a chip, and *checking* his design.

It would be easier if the computer could check his design. This can be done with a combination of functional analysis (or verification) and timing analysis (or verification). The idea is to let the functional analyzer verify that the circuit implements the correct function and let the timing analyzer verify that the circuit does not violate any external or internal timing

constraints. If this can be done, then simulation of the design would be redundant.

Functional Analysis

Functional analyzers verify the function of a design. They usually compare the design against a specification which is often given in terms of a Boolean function. The functional analyzer then compares the specified Boolean function to the one that is implemented with logic gates. If they are not equivalent then an error is reported. If they are equivalent then the design is considered correct. This is an ongoing research topic[Bir88].

Timing Analysis

Timing analyzers analyze the timing within a design. Like the functional analyzer, it also is used to compare the design against a specification. To many designers, the proper design technique is a *synchronous design*. If all storage elements are edge triggered on one clock edge, it is quite easy to formulate proper timing behavior by a set of mathematical formulas. These restrictions on the design may be relaxed and still result in a set of mathematical formulas that allow complete checking of the circuits timing.

6.2.2 Timing Analysis versus Simulation

Why Analysis is Faster than Simulation

Timing analysis is inherently faster than simulation because it involves solving a fixed set of mathematical formulas. These formulas usually specify timing constraints within a single master clock cycle. Simulation requires hundreds to thousands of clock cycles to give meaningful results. Even if

there are a lot of computations involved in timing analysis, it can check for *timing faults* orders of magnitude faster than simulators.

Why Simulation is Better than Analysis

Simulation is better because it simulates chip function as well as timing. A timing analyzer seldom does functional simulation or analysis.

However this does not give a great advantage to a simulator that can simulate timing. If a circuit has been verified to have no *timing faults*, a simpler functional simulator (such as a unit delay simulator) may be used to find the functional behavior of the circuit. Since a functional simulator is 3 to 4 orders of magnitude faster than a timing simulator, timing simulators might not be needed if a robust *timing analyzer* has been used.

Only when multiple clocks exist, or when the function of the circuit affects timing, will the simulator with timing need to be used.

6.2.3 The Timing Analyzer

The Timing Formulas

There are many critical timing checks. They can all be formulated as inequalities. These all depend on the fact that all data transitions can be traced to a signal clock edge on the master clock input. Things are assumed to be stable before this clock input, and all time references are to this clock input edge.

Hold Time Check: At a synchronous latching element, a *hold time* check can be expressed as:

$$\begin{aligned}
 &(\text{latest clock arrival time}) + (\text{hold time}) \\
 &< (\text{earliest data arrival time}). \qquad (6.1)
 \end{aligned}$$

Note that these times may be correlated and ambiguities may be removed.

Set-up Time Check: At a synchronous latching element, a *set-up* time check can be expressed as:

$$\begin{aligned} & (\text{latest data arrival time}) + (\text{set up time}) \\ & < (\text{earliest clock arrival time}). \end{aligned} \quad (6.2)$$

Note that these times may be correlated and ambiguities may be removed by *common ambiguity resolution*.

Minimum Pulse Width: To check a high minimum pulse width:

$$\begin{aligned} & (\text{earliest signal fall time}) - (\text{latest signal rise time}) \\ & > (\text{minimum pulse width}). \end{aligned} \quad (6.3)$$

Note that these times may be correlated and ambiguities may be removed by *common ambiguity resolution*. The low minimum pulse width may be handled as in similar manner.

Clock Skew: To check the clock skew, one checks to see if the clock edges are within a maximum tolerated clock skew from each other.

$$\begin{aligned} & (\text{active clock edge } A) - (\text{active clock edge } B) \\ & + (\text{maximum tolerated clock skew}) \geq 0. \end{aligned} \quad (6.4)$$

$$\begin{aligned} & (\text{active clock edge } B) - (\text{active clock edge } A) \\ & + (\text{maximum tolerated clock skew}) \geq 0. \end{aligned} \quad (6.5)$$

It is easy to see that all these *timing violations* can be checked if these formula can be evaluated.

The Trapezium

The *trapezium* that existed in the simulator exist in the timing analyzer. They are characterized by two variables. The first is *die parameter*, and the second is a *local spread factor*. With reference to figure 4.3 on page 49 a delay could be represented as:

$$b + (d - b)x_0 + (a - b)x_1 + (b + c - d - a)x_1x_0 \quad (6.6)$$

where x_0 represents the *die parameter* and x_1 represents the *local spread factor*. Since all timing can be expressed as a *trapezium*, all that is needed is to ensure that the set of equations hold for all possible values of the *die parameter* and the *local spread factors*. These equations can be checked in linear time for each *delay path*. See appendix A for details of the algorithm.

Successive Solutions

Instead of solving all these equations at once, a succession of increasingly more accurate probes into *timing violations* can be done. This may speed up analysis by an order of magnitude.

Also a succession of timing analyses means that more and more accurate results can be obtained and yet allow for analysis to be stopped at any time. Better results are being generated as the analysis continues, but results are available after the first quick timing analysis.

Crude Timing Analysis: First a crude timing model of all gates can be used. This is a bound of the behavior of the gate for all rising or falling inputs and a wide range of input *transition times*. This crude timing model is used to speed up the timing analysis without sacrificing the bounding properties of the model.

The timing analysis is done using this crude model on all *delay paths*. This timing analysis can verify that paths have no *timing violations*. The analysis then orders the paths by some measure of how critical the path is to the function of the design. An example of a measure of a critical path is the propagation delay through this path. This information is then passed on to the refined timing analysis.

Refined Timing Analysis: An accurate timing model of all gates can be used for the second round of timing analysis. Since there are fewer paths to be checked (since the crude timing analysis can check paths as having no *timing violations*) less time is needed to perform this analysis because not as many paths will need to be checked. Also this timing analysis can examine the most critical paths first and stop the analysis as soon as a specified number of critical paths have been checked.

Ambiguity Resolution Checks: All *timing violations* may be checked for false violations cause by timing ambiguities. This may be incorporated into the earlier timing analysis tests.

False Path Resolution: Some *timing violations* may never occur since the logic of the circuit will not allow a particular *delay path* to be activated. The remaining suspected violations may be checked for these types of false paths[Ben87b].

Path Duplications The existing violations may be collapsed into a single violation if the paths causing the violation are identical. For example in a 32 bit ALU many *timing violations* will be flagged 32 times (one for each bit). If these violations are collapsed into one, there is more information given to the designer with fewer words.

Subpath Violations: A second set of violations may be flagged as traveling through a common sub-path. As before if this is flagged as a common sub-path problem, more information is given to the designer with fewer words.

6.3 Hierarchical Timing Analysis

Hierarchy in timing analysis allows for speedup of timing checks in the same way hierarchy in simulators speeds up simulation. This is accomplished by substituting a behavioral model for a gate level model.

6.3.1 Macro Models

Timing analysis may be performed on a subcircuit of a chip and analyzed for faults. If the subcircuit is fault free, *set-up* and *hold times* of the inputs may be specified for proper functioning and a maximum clock rate and duty cycle may be found. These numbers can be calculated by the timing analyzer. For example, this can be done to characterize a chip.

The whole block can be treated as a clocked state machine and incorporated into other timing analysis runs. This idea is suggested in [Ben87a].

Chip to Chip

Since the single chip environment does not exist when two chips are being considered special care must be taken not to correlate gate delays on different chips, since *process* and *environmental parameters* usually will not be the same on two different chips.

6.4 Functional Analysis

6.4.1 Simulators

Faster Simulators

If the timing analysis for a block has already been done, then only the function of a block needs to be verified. This can be done with a unit delay simulator, or state machine simulators. The advantage with these simulators is that they can run many orders of magnitude faster than standard event driven simulators.

Symbolic Simulators

Symbolic simulators simulate a design with algebraic variables instead of the usual '0' and '1'. With the timing of a circuit verified, the circuit can be collapsed into a state machine representation. This may allow for practical symbolic simulation which was not practical before because symbolic simulation cannot handle the timing details well.

6.4.2 Hierarchy

If as mentioned before, the circuit function is represented as a state machine, then multiple functional blocks can be merged by state machine mergers [Koh78]. This would allow an abstraction of function. This along with the abstraction of timing mentioned previously would allow for the building up of higher level gate models. This would allow for great savings of computer time in analyzing boards or systems.

6.6.2 Asynchronous clocks

Another case of *non-synchronous design* is in asynchronous data channels. The input signal bit rate may only be known to within $\pm 50\%$ and must be matched to another clock. In this case a FIFO is usually used.

Now can a FIFO model be incorporated into the proposed timing analyzer? And even more generally, can a FIFO be analyzed with the timing analyzer?

6.7 Timing information in data

The above circuits usually work by passing timing information between the blocks with asynchronous clocks. The timing information is usually passed in terms of flags that signal buffer full, NACK, slip request, etc.

The question is can explicit knowledge of these signals help the timing analyzer analyze the circuits that employ these signals? Could the timing analyzer infer that these signals exist? Could it make any use of this information?

6.6.2 Asynchronous clocks

Another case of *non-synchronous design* is in asynchronous data channels. The input signal bit rate may only be known to within $\pm 50\%$ and must be matched to another clock. In this case a FIFO is usually used.

Now can a FIFO model be incorporated into the proposed timing analyzer? And even more generally, can a FIFO be analyzed with the timing analyzer?

6.7 Timing information in data

The above circuits usually work by passing timing information between the blocks with asynchronous clocks. The timing information is usually passed in terms of flags that signal buffer full, NACK, slip request, etc.

The question is can explicit knowledge of these signals help the timing analyzer analyze the circuits that employ these signals? Could the timing analyzer infer that these signals exist? Could it make any use of this information?

Chapter 7

Conclusion

A gate level timing model has been developed that accurately emulates the affects of process and environmental variations. The interaction between gates is improved over most digital simulators by incorporating *transition times* into the timing models. In addition, the timing model allows for uncertainty in capacitive loading and *transition time*. The added uncertainty due to modeling is less than that caused by uncertainty in capacitive loading. All this was done without increasing the expected simulation time by more than a factor of four over existing simulators[PCS88][VER89][FUN89].

The timing model with absolute bounds incorporates all the desired properties in a gate level model. Although the simpler model does not incorporate the desired property of absolute bounds, it is easier to characterize than the model with absolute bounding. Even without the absolute bounding the simple gate level timing model is a large improvement over other gate level timing models[LSI85][TI86][NCR85].

7.1 Contributions in this Thesis

7.1.1 Die Parameter

The models major innovation is the use of the *die parameter*. This omnibus parameter allows the model to

- take into account all process and environmental variations affecting the chip.
- correlate all gate delays within the chip.
- bound gate timing values without introducing large timing uncertainties associated with large variations in process and environment.

The disadvantages of using the *die parameter* are small. They are:

- the effort involved in characterizing the die parameter. (This only needs to be done once.)
- a small uncertainty in timing values is added to the model. (This is much smaller than the uncertainty caused by capacitive loading.)
- the timing model of the gate has increased in complexity over standard models.

7.1.2 Provable Bound With Correlation

Another innovation is the ability to introduce a set of bounds on the timing model that can be proven. Although the characterization effort involved is large, the availability of an absolute bound can be beneficial to organizations that need high reliability. Also, these absolute bounds can be made smaller than the standard minimum to maximum bound by the use of the *die parameter* function.

7.1.3 Uncertainty in Circuit Parameters

The timing model introduced is believed to be the first to allow for uncertainty in capacitive loading as well as signal *transition time*.

7.1.4 Timing Verification

Timing verifiers formulate timing constraints as inequalities. The optimization theorem in appendix A shows that these inequalities can be tested by considering only two values of the *die parameter*. Thus, by using these new timing models, a standard *timing verifier* can verify the timing constraints for all variations in *process, environment* and *circuit parameters*.

7.2 Summary

In conclusion, the gate level timing model overcomes limitations of existing gate level timing models and gives more accurate results. Both the absolute bound timing model and the simpler bounded timing model permit true worst case analysis of a digital design. Although the simpler bounded timing model does not bound the timing values absolutely, it is easier to characterize and is less pessimistic than the absolute bounded timing model. Both these timing models should prove to be valuable in the design of timing verifiers and digital simulators.

Appendix A

Solving Constraint Equations

The general form of a constraint is

$$\text{clock period} + \sum_i \text{delay}_i - \sum_j \text{delay}_j \geq 0,$$

where

$$\text{delay}_i = \alpha_i + \beta_i x_i + \gamma_i x_0 + \delta_i x_i x_0.$$

Here x_0 represents the die parameter and x_i is used to represent the local spread factor.

The problem then becomes one of finding

$$\min \left\{ f(\hat{x}) = \sum_{i=1}^m (\alpha_i + \beta_i x_i + \gamma_i x_0 + \delta_i x_i x_0) \right\}$$

Such that

$$0 \leq x_i \leq 1 \quad i = 0, 1, \dots, n.$$

A.0.1 Optimization Theorem

The following theorem is useful to whether the constraint holds, or what constraints are critical.

Theorem:

The maximum value of $f(\tilde{x})$ occurs when $x_0 = 0$ or $x_0 = 1$.

Proof:

Let $x'_1, x'_2, \dots, x'_n$ and x'_0 be a solution that maximizes $f(\tilde{x})$. Rewrite $f(\tilde{x})$ in terms of the variable x_0 . This gives

$$f(\tilde{x}) = g(x_0) = mx_0 + b$$

where

$$m = \sum_{i=1}^m (\gamma_i + \delta_i x'_i)$$

and where

$$b = \sum_{i=1}^m (\alpha_i + \beta_i x'_i).$$

Since $g(x_0)$ is a linear function of x_0 , $g(x_0)$ obtains a maximum at the extreme values of x_0 . Thus the maximum of $f(\tilde{x})$ occurs at $x_0 = 0$ or $x_0 = 1$. *QED.*

To solve the original problem, two cases need to be considered.

1. If $x_0 = 0$ then

$$f(\tilde{x}) = \sum_{i=1}^n (\alpha_i + \beta_i x_i).$$

Therefore

$$\max\{f(\tilde{x})\} = \sum_{i=1}^n \alpha_i + \sum_{i=1}^n \max(0, \beta_i).$$

2. Or if $x_0 = 1$ then

$$f(\tilde{x}) = \sum_{i=1}^n (\alpha_i + \beta_i x_i + \gamma_i + \delta_i x_i).$$

Therefore

$$\max\{f(\tilde{x})\} = \sum_{i=1}^n (\alpha_i + \gamma_i) + \sum_{i=1}^n \max(0, \beta_i + \delta_i).$$

A.0.2 Results

Thus to evaluate the constraint equations all one has to do is to assume that the die parameter is 0 or 1. Then one evaluates the coefficients of x_i to find out whether the worst case occurs with $x_i = 0$ or $x_i = 1$.

All that is left is to check the inequality or to find the values of period that the inequality hold.

Appendix B

Glossary

The following is a short glossary of terms used in this paper. The numbers in the ‘{ }’s are the page on which the term is first used or best described.

An **absolute bound** { 60 } of a quantity is a range of values within which the quantity is guaranteed to lie.

Circuit parameters { 4 } are quantified properties of individual gates that are affected by placement within the circuit or chip. These parameters include capacitive loading and input signal slew rate.

Common ambiguity resolution { 23 } is a method to resolve uncertainty in timing. See section 2.3.2 for a description of the method.

Correlation { 14 } is a measure of how values are related. If timing values are well correlated then there is a function that can approximate one value knowing the other. Timing values are correlated by common parameters that affect delays.

Delay multipliers { 11 } are used in most gate level simulators. The maximum delay multiplier is the number which is multiplied by the

nominal delay to get the maximum delay of a gate. Likewise the minimum delay multiplier is the number which is multiplied by the nominal delay to get the minimum delay of a gate.

A **delay path** { 51 } through a gate is any path from an input to an output that can cause a different timing value to be measured. Because of this definition, rising and falling inputs will result in two different delay paths. Also a delay path may depend on other gate inputs.

The **die parameter** { 30 } is a synthesized parameter which represents all the environmental and process parameters. The die parameter is used to correlate gate timing values within a chip.

Environmental parameters { 4 } are quantified properties of a die caused by the external environment. These parameters include supply voltage, junction temperature and radiation levels.

Fall time { 3 } is the time it takes an output signal to fall from a high logic level to a low logic level.

Fast { 21 } chips refer to a chip whose gate's delays are at the extreme minimum. This corresponds to a *die parameter* value of 0.

A **global bound** { 71 } of a function is a range of values within which the function is guaranteed to lie under a wide range of possible inputs.

Hold time { 3 } is the length of time the data signal of a clocked element must remain stable after the clock triggers the storage element.

Inertial delay { 38 } is the time between the gate output reaching 10% of its range from the time the input reached the gate's switching threshold. See figure 3.3 on page 34.

Intrinsic delay { 31 } is the time between the gate output reaching 10% of its range from the time the input reached 10% of its range. See figure 3.3 on page 34.

The **local spread factor { 82 }** is a random variable (which happens to range from 0 to 1) that is used to represent uncertainty in gate timing values.

A **parameter { 4 }** is a value associated with a property, and is a numerical value.

Process parameters { 4 } are quantified physical properties of a die caused by manufacturing the chip. These parameters include gate length, oxide thickness and gate-drain overlap.

Propagation delay { 3 } is the time it takes a gate to produce the correct output logic level when the correct input levels are present at the inputs.

The **raw die parameter { 38 }** is a synthesized parameter which is meant to represent the gate delay variation with respect to environmental and process parameters. Once it is found it is scaled to $[0, 1]$ and becomes the die parameter.

Relative error { 44 } is defined to be the error divided by the actual value.

Rise time { 3 } is the time it takes an output signal to rise from a low logic level to a high logic level.

Set-up time { 3 } is the length of time the data signal of a clocked element must remain stable before the clock triggers the storage element.

Slew rate { 16 } is a measure of how fast a signal is changing. It is measured in volts per second and is inversely proportional to the transition time.

Slow { 21 } chips refer to a chip whose gate's delays are at the extreme maximum. This corresponds to a *die parameter* value of 1.

Spread simulators { 22 } simulate designs in time while keeping track of some measure of uncertainty about when things occur.

Switching threshold { 5 } is the value of the input voltage of a gate at which the output voltage would be at 50% of its range. Ideally this value would be $0.5V_{DD}$.

A **synchronous design { 79 }** is a digital design that has all storage elements clocked from a common clock edge.

A **timing analyzer { 20 }** is a program that analyzes the timing of a circuit. It is "smarter" than a timing verifier.

Timing faults { 14 } result in improper functioning of a chip because timing specifications of the chip or individual gates were violated.

A **timing verifier { 20 }** is a program that verifies proper circuit function against a specification. It is often used to find long delay paths in a design. See 'timing analyzer.'

Timing violation { 14 } is another name for timing faults.

Transition time { 5 } is the time it takes a signal to change from 10% of its range to 90% of its range. For a falling signal this is the time it takes to go from $0.9V_{DD}$ to $0.1V_{DD}$.

A **trapezium** { 31 } is a figure bounded by four lines, two of which are parallel. The trapezium is the shape of the region that bounds timing values of a gate.

References

- [Ben87a] J. Benkoski and A.J. Strojwas, 'A New Approach to Hierarchical and Statistical Timing Simulations,' *IEEE Transactions on Computer Aided Design*, Vol. CAD-6, No. 6, November 1987, pp.1039-1052.
- [Ben87b] J. Benkoski, E. Vanden Meersch, L. Claesen, and H. De Man, 'Efficient Algorithms for Solving the False Path Problem in Timing Verification,' *IEEE International Conference on Computer-Aided Design (ICCAD-87) Digest of Technical Papers*, November 8-12, 1987, pp. 44-47.
- [Bir88] Graham Birtwistle and P.A. Subrahmanyam editors, *VLSI Specification, Verification and Synthesis*, Proceedings of a workshop held in Calgary from 12-16 January 1987.
- [FUN89] Bell-Northern Research, "FUNSIM" - A gate level simulator used within Bell-Norther Research.
- [Gho86] Sumit Ghosh, 'A Rule-Based Approach to Unifying Functional and Fault Simulation and Timing Verification,' *23rd Design Automation Conference* (1986). pp.677-682.

- [Gre86] Roubik Gregorian and Gabor C. Temes, *Analog MOS Integrated Circuits for Signal Processing*. John Wiley & Sons. (1986) pp. 86-93.
- [Kat86] J. Katzenelson and E. Weitz, 'VLSI Simulation and Data Abstractions,' *IEEE Transactions on Computer Aided Design*, Vol. CAD-5, No. 3, July 1986, pp.371-378.
- [Koh78] Z. Kohavi, *Switching and Finite Automata Theory*. Rockville, MD: Computer Science Press, 1978.
- [LSI85] LSI Logic Corporation *CMOS Macrocell Manual*, (1985).
- [Nas86] Sani R. Nassif, Andrzej J. Strojwas and Stephen W. Director 'A method for worstcase analysis of integrated circuits.' *IEEE Transactions on Computer Aided Design*, Vol. CAD-5. No.1, (1986) pp.104-113.
- [NCR85] NCR Microelectronics Division *NCR Microelectronics Data Book*, (1985). "NCR Semicustom Design" section.
- [PCS88] Simucad Inc. *P/C SILOS Logic Simulator User's Manual*, (1988).
- [Pre88] William H. Press, Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling, *Numerical Recipes in C*. Cambridge University Press (1988).
- [Spe88] Robert Spence and Randeep Singh Soin *Tolerance Design of Electronic Circuits*. Addison-Wesley Publishers Ltd. (1988) pp.29-33.
- [TI86] Texas Instruments Incorporated *2 - μ m CMOS Standard Cell Data Book*, (1986).

- [VER89] Gateway Design Automation Corporation, *VERILOG Hardware Description Language and Simulator Reference Manual (V1.2)*, March 1989.
- [Wes85] Neil H. E. Weste and Kamran Eshraghian, *Principles of CMOS VLSI Design*. Addison-Wesley Publishers Ltd. (1985).

END

13|09|90

FIN