

AD-A220 748

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NW-LIS-89-08-05	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The On-Chip Parallelism of VLSI Circuits (THESIS)		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mary Lane Bailey		8. CONTRACT OR GRANT NUMBER(s) N00014-88-K-0453
9. PERFORMING ORGANIZATION NAME AND ADDRESS Northwest Laboratory for Integrated Systems University of Washington Dept. of Comp. Science, FR-35 Seattle, WA 98195		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA-ISTO 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE August 1988
		13. NUMBER OF PAGES 145
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research - ONR Information Systems Program - Code 1513: CAF 800 North Quincy Street Arlington, VA 22217		18. SECURITY CLASS. (of this report) Unclassified
		18a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Distribution of this report is unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) NW-LIS, VLSI, parallelism, logic simulation, circuits, CMOS, synchronous, asynchronous, unit-delay, event-driven.		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report uses two approaches. First, the formulation of a model for studying circuit parallelism and the potential speedup of parallel logic-level simulation. Second, the development of a methodology for measuring circuit parallelism, and its use to determine the parallelism of nine circuits using two different simulators. (see attached Abstract)		

DTIC
ELECTE
APR 20 1990
D
E

**The On-Chip Parallelism
of VLSI Circuits**

Mary Lane Bailey

**Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195**

**Technical Report 89-08-05
August, 1989**

The On-Chip Parallelism of VLSI Circuits

by

Mary Lane Bailey

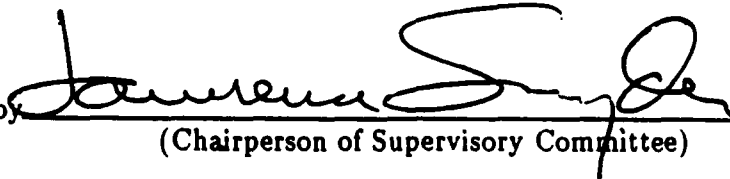
A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

1989

Approved by



(Chairperson of Supervisory Committee)

Program Authorized
to Offer Degree

Department of Computer Science

Date

July 19, 1989

© Copyright by
Mary Lane Bailey
1989



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Doctoral Dissertation

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 300 North Zeeb Road, Ann Arbor Michigan 48106, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature Mary L. Bailey

Date 24 July 1987

①

University of Washington

Abstract

The On-Chip Parallelism of VLSI Circuits

by Mary Lane Bailey

Chairperson of the Supervisory Committee: Professor Lawrence Snyder

Department of Computer Science

Simulation is a bottleneck in VLSI circuit design. Not only are there many simulation runs throughout the design cycle, but each run can take hours or days to complete. One often suggested means of speeding up event-driven simulation is to use multiple processors to exploit the natural parallelism present in the circuit, that is to partition the circuit among multiple processors, with each executing the same algorithm on its portion of the circuit. This approach assumes that there is sufficient activity, or circuit parallelism, in the circuit to keep all of the processors busy.

The author) ~~We have~~ used two approaches in this work. First, ^{She} ~~we have~~ formulated a model for studying circuit parallelism and the potential speedup of parallel logic-level simulation. Using this model ~~we have~~ ^{She} considered the effect of the choice of timing model and synchronization strategy on speedup. ~~We have~~ ^{She} also investigated the effect of circuit size on parallelism.)

← Additionally, ^{She} ~~we have~~ developed a methodology for measuring circuit parallelism, and used it to determine the parallelism of nine circuits using two different simulators. Empirical measurements have also been used to validate portions of the formal model. (KR)

The major results of the model are:

↑

- Unit-delay timing provides at least as much parallelism as variable-delay or fixed-delay timing.
- Asynchronous timing strategies can improve simulation speed over synchronous strategies. However, for unit-delay timing, if all event evaluation times are equal, the asynchronous strategies do not provide additional speedup over synchronous simulation.
- In general, the percentage of parallelism is *not* constant over circuit size, even for members of the same circuit family.

We have used empirical results to validate the parallelism results for variable-delay and unit-delay synchronous simulation. We also have empirical results that show that the percentage of parallelism changes as the instance size changes for these strategies.

Finally, the empirical measurements have provided a set of benchmarks for the parallelism of circuits. These measurements are remarkably low for variable-delay timing. Thus, the direct application of circuit activity to synchronous parallel simulation for variable-delay timing is doomed! It may be feasible for unit-delay timing, or for variable-delay timing using asynchronous strategies.

Table of Contents

List of Tables	v
List of Figures	viii
Chapter 1: Introduction	1
1.1 Timing Models	3
1.2 Synchronization Strategies	5
1.3 Contributions of This Work	6
1.4 Thesis Organization	7
Chapter 2: Related Work	8
2.1 Switch-Level Algorithms	8
2.2 Parallelism Measurements	10
2.3 Parallel Simulators	12
2.3.1 Parallel Hardware Simulation Engines	12
2.3.2 General Purpose Parallel Simulators	14
2.4 General Parallel Simulation Strategies	15
Chapter 3: A Formal Model for Circuit Simulation	17
3.1 The Circuit Value Problem	17
3.2 The Model	20
3.3 Synchronous Simulation	25

3.3.1	Variable-delay Model	25
3.3.2	Unit-delay Model	31
3.3.3	Fixed-delay Model	32
3.3.4	Time	33
3.4	Conservative Asynchronous Strategy	37
3.5	Optimistic Asynchronous Algorithms	46
3.6	Modeling the Effects of Circuit Size on Parallelism	49
3.6.1	Tree Additions	50
3.6.2	Subtree Addition	51
3.6.3	Dilation	52
3.6.4	Mixed Changes	54
3.7	Summary	55
Chapter 4: Corroborating the Model		56
4.1	How Do We Measure Circuit Parallelism?	56
4.2	SwitchSim	58
4.3	RNL	59
4.3.1	Variable-Delay Algorithm	59
4.3.2	Pseudo Unit-Delay Algorithm	60
4.3.3	Calibrating RNL	61
4.4	The Queue Metric	70
4.4.1	Adding the Queue Metric to the Model	71
4.4.2	Calibrating RNL	74
4.5	Experimental Methodology	76
4.6	Measuring the Parallelism of a Multiplier Circuit	77
4.6.1	RNL Measurements	78
4.6.2	SwitchSim Measurements	82
Chapter 5: Baseline Measurements		84

5.1	Test Data	88
5.2	RNL Measurements	89
5.3	SwitchSim Measurements	93
5.4	Summary	95
Chapter 6: Variable-delay Measurements		96
6.1	Effects of Increasing RNL's Timebase	96
6.1.1	Baugh-Wooley Multiplier	98
6.1.2	Booth Multiplier	104
6.1.3	Decoder	108
6.2	Effects of Decreasing RNL's Timebase	114
6.3	Assumptions of the Formal Model	116
6.3.1	Overhead	116
6.3.2	Model Granularity	117
6.3.3	Event Sequence	118
Chapter 7: Varying Circuit Size		122
7.1	Baugh-Wooley Multiplier	123
7.2	Booth Multiplier	126
7.3	Shift Register	130
7.4	Decoder	133
Chapter 8: Conclusions		136
Bibliography		140

List of Tables

3.1	Event Sequence Favoring Optimistic Strategy	18
3.2	Event Sequence Favoring Conservative Strategy	49
3.3	Summary of How Circuit Growth Affects the Percentage of Parallelism . .	54
4.1	Comparing SPICE and RNL Transition Times for the 2 To 4 Decoder. . .	62
4.2	Event Times for the 2 To 4 Decoder	64
4.3	Comparing SPICE and RNL Transition Times for the Shift Register. . . .	66
4.4	Event Times for the Shift Register.	67
4.5	Comparing SPICE and RNL Transition Times for the Baugh-Wooley Multiplier Cell.	69
4.6	The Event Times for the Baugh-Wooley Multiplier Cell.	69
4.7	The Transition Times for the 2 To 4 Decoder	74
4.8	The Transition Times for the Shift Register.	75
4.9	The Transition Times for the Baugh-Wooley Multiplier Cell.	75
4.10	Data for Test Case 1	78
4.11	Parallelism for the 8×8 Booth Multiplier Using the Event Metric and RNL	81
4.12	Parallelism for the 8×8 Booth Multiplier Using the Queue Metric and RNL	82
4.13	Parallelism for the 8×8 Booth Multiplier Using SwitchSim	83
5.1	Circuit Parallelism Using the Event Metric and RNL	90
5.2	Percentage of Circuit Parallelism Using the Event Metric and RNL	91
5.3	Circuit Parallelism Using the Queue Metric and RNL	92

5.4	Percentage of Circuit Parallelism Using the Queue Metric and RNL	93
5.5	Circuit Parallelism Using SwitchSim	94
6.1	Static Experiment for 16 × 16 Baugh-Wooley Multiplier using the Event Metric	99
6.2	Dynamic Experiment for 16× 16 Baugh-Wooley Multiplier using the Event Metric	99
6.3	Percentage of Events due to Charge-Sharing for the Baugh-Wooley Mul- tiplier	101
6.4	Dynamic Experiment for 16× 16 Baugh-Wooley Multiplier using the Queue Metric	103
6.5	Static Experiment for the 16 Booth Multiplier using the Event Metric . .	105
6.6	Dynamic Experiment for the 16 Booth Multiplier using the Event Metric	105
6.7	Percentage of Events due to Charge-Sharing in the Booth Multiplier . . .	106
6.8	Dynamic Experiment for 16 × 16 Booth Multiplier using the Queue Metric	108
6.9	Static Experiment for the 6 to 64 Decoder using the Event Metric	110
6.10	Dynamic Experiment for the 6 to 64 Decoder using the Event Metric . . .	110
6.11	Percentage of Events due to Charge-Sharing in the Decoder	111
6.12	Dynamic Experiment for 6 to 64 Decoder using the Queue Metric	112
6.13	Decreasing the Timebase for the Baugh-Wooley Multiplier	115
6.14	Decreasing the Timebase for the Booth Multiplier	115
6.15	Events in the Baugh-Wooley Parallelism Measurements - Event Metric . .	119
6.16	Events in the Booth Multiplier Parallelism Measurements - Event Metric	120
6.17	Events in the Decoder Parallelism Measurements - Event Metric	121
7.1	Parallelism of the Baugh-Wooley Multiplier Using the Event Metric and RNL	124
7.2	Percent Parallelism of the Baugh-Wooley Multiplier Using the Event Met- ric and RNL	124

7.3	Parallelism of the Baugh-Wooley Multiplier Using the Queue Metric and RNL	125
7.4	Percent Parallelism of the Baugh-Wooley Multiplier Using the Queue Metric and RNL	125
7.5	Parallelism of the Baugh-Wooley Multiplier Using SwitchSim	125
7.6	Parallelism of the Booth Multiplier Using the Event Metric and RNL	126
7.7	Percent Parallelism of the Booth Multiplier Using the Event Metric and RNL	127
7.8	Parallelism of the Booth Multiplier Using the Queue Metric and RNL	127
7.9	Parallelism of the Booth Multiplier Using SwitchSim	128
7.10	Parallelism of the 24 × 24 Booth Multiplier Using the Event Metric and RNL	129
7.11	Percent Parallelism of the 24 × 24 Booth Multiplier Using the Event Metric and RNL	129
7.12	Parallelism of the 24 × 24 Booth Multiplier Using the Queue Metric and RNL	130
7.13	Parallelism of the 24 × 24 Booth Multiplier Using SwitchSim	130
7.14	Parallelism of the Shift Register Using the Event Metric and RNL	131
7.15	Percent Parallelism of the Shift Register Using the Event Metric and RNL	131
7.16	Parallelism of the Shift Register Using the Queue Metric and RNL	132
7.17	Parallelism of the Shift Register Using SwitchSim	133
7.18	Parallelism of the Decoder Using the Event Metric and RNL	133
7.19	Percent Parallelism of the Decoder Using the Event Metric and RNL	134
7.20	Parallelism of the Decoder Using the Queue Metric and RNL	134
7.21	Parallelism of the Decoder Using SwitchSim	134

List of Figures

1.1	Comparing Unit-Delay and Pseudo Unit-Delay Timing	4
3.1	A Stable State of a Circuit	21
3.2	An Unstable State of a Circuit	21
3.3	Initial State of the Example Circuit	23
3.4	Dependency Graph	24
3.5	Example of Applying P_3 to \mathcal{P}_1	28
3.6	Example where Unit-Delay Execution is Greater than Variable-Delay Execution	38
3.7	Example Circuit for Conservative Asynchronous Strategy	40
3.8	Dependency Forest for the Example	41
3.9	Comparing Asynchronous Strategies with Limited Processors	48
4.1	2 To 4 Decoder Circuit	63
4.2	Portion of the Decoder Circuit.	65
4.3	2-Stage 1-Bit Shift Register Circuit	66
4.4	Schematic of the Multiplier Cell	68
6.1	Timebase Experiments for the 16×16 Baugh-Wooley Multiplier Using the Event Metric	98
6.2	The Effect of Charge-Sharing in the 16×16 Baugh-Wooley Multiplier . .	101

6.3	Timebase Experiments for the 16×16 Baugh-Wooley Multiplier Using the Queue Metric	102
6.4	Comparing the Event and Queue Metrics for the Baugh-Wooley Multiplier	103
6.5	Timebase Experiments for the 16×16 Booth Multiplier using the Event Metric	104
6.6	Effect of Charge-Sharing on the Parallelism of the 16×16 Booth Multiplier	107
6.7	Timebase Experiments for the 16×16 Booth Multiplier Using the Queue Metric	107
6.8	Comparing the Event and Queue Metrics Timebase Changes in the Booth Multiplier	109
6.9	Timebase Experiments for the 6 to 64 Decoder using the Event Metric . .	109
6.10	Effect of Charge-Sharing on the Parallelism of the Decoder	112
6.11	Timebase Experiments for the 6 to 64 Decoder Using the Queue Metric .	113
6.12	Comparing the Event and Queue Metrics for Timebase Changes in the Decoder	114

Acknowledgements

I would like to thank my advisor, Larry Snyder, for the years of guidance, understanding, encouragement and support he has so generously provided during my tenure as his student. I also would like to thank the other members of my thesis committee, Carl Ebeling, Burton Smith, Jean-Loup Baer, Hank Levy, and Don Marshall for their guidance during this work. Jane Cameron and Fran Berman deserve credit for helping me in the important process of selecting my advisor.

The members of the NWLIS also supported this work. Warren Jessop always seemed to be able to find more disk space for me when necessary. Gaetano Borriello, Larry McMurchie, Wayne Winder, Bill Barnard, and Karen Bartlett were wonderful technical resources. Vicky Palm and Tony Marriott were invaluable when it came to travel, getting papers out at the last minute, and all of the other administrative headaches that one encounters in the process of doing research.

Many colleagues have contributed to the successful conclusion of my graduate career. I would specifically like to thank my fellow graduate students Dave Socha, Magda Konstantinidou, Bob Cypher, Sam Ho, and Jason Lin and my friends and colleagues at Data I/O, Kyu Lee, Michael Bradley, Michael Holley, Brian Durwood, and Paul Brownlow. Larry Mayhew of Data I/O created a small fellowship when I returned to school full time.

Finally, I would never have been able to survive graduate school without the unending support of my family. My parents deserve credit for encouraging me to choose my own path, and their enthusiastic support of my progress. My husband, Hop, and children, Robby and Katie, financed this endeavor and supported me throughout with love and understanding.

In Memory of Joseph E. Lane, Jr.

Chapter 1

Introduction

Simulation is the principal tool used in VLSI design to determine the correctness of a circuit and for analyzing its performance before fabrication. Because fabrication is so expensive, both in time and money, it is essential that a circuit has a high probability of working correctly the first time. However, even moderately large circuits can take hours or days to simulate.

In recent years, technology has allowed larger and larger circuits to be placed on a single chip, a trend which should continue in the near future. Unfortunately, as circuit size increases, so does simulation time, and advances in simulation have not kept up with those in technology. Thus, circuit simulation, already a time-consuming part of the design process, is becoming an increasing bottleneck in the VLSI design process.

One way to decrease simulation time is to increase the abstraction level of the simulator. Circuit-level simulators, for example SPICE [Nagel 75], solve the differential equations describing the state of all structures on a chip. They provide detailed timing information and, if parametrized with the correct process values, are widely regarded as reliable. They use and report actual voltage values. These simulators typically can only handle relatively small circuits, on the order of hundreds of transistors.

Switch-level simulators idealize a transistor as a switch and compute the resulting state of a circuit. Voltages are also abstracted to a small number of discrete values.

typically 0, 1, and X. These simulators can handle much larger circuits than circuit-level simulators, including most circuits that can fit on VLSI chips today. They cannot correctly simulate the analog characteristics of circuits, but do correctly model the bidirectionality of transistors. Switch-level simulators come in two varieties: those with timing and those which only provide functional results.

There are at least two other abstraction levels that should be mentioned. Logic-level or gate-level simulators model circuits as boolean functions. Here the bidirectionality of transistors is lost, so if this is essential to the circuit's function, this abstraction level will not yield the proper results. The Yorktown Simulation Engine [Denneau 83] is an example of a gate-level simulator. Finally, behavioral or functional-level simulators represent large portions of the circuit by a single model. N.2 [Ordy 83] is an example of this type of simulator.

Though simulating at higher levels of abstraction is useful during the design process, it is usually desirable to simulate the entire chip at the switch-level. Some designers do this early in the design process, using a netlist representation of the chip. Then the final layout can be compared to this netlist by using a verification tool such as Gemini [Ebeling 88] to ensure correctness. Others do this just before fabrication, using a simulation file extracted from the actual chip layout. In either case, the entire chip is usually simulated at the switch-level abstraction. Because the entire chip is simulated at the switch-level, the simulation time can be quite long. We would like to decrease this time, so we are primarily concerned with switch-level simulation.

Parallel simulation has often been suggested as a means of speeding up circuit simulation. For switch-level, logic-level, and functional-level simulation, one common approach is to partition the circuit among multiple processors, with each processor executing the same algorithm on its portion of the circuit [Smith 86]. For synchronous event-driven simulation, this has several implications. First, there must be a large amount of circuit activity, or circuit parallelism, in the circuit. Second, there must be a good partition which spreads out this activity. Finally, the overhead of communication and synchro-

nization must be reasonable.

We focus on the first issue, circuit activity or circuit parallelism. Circuit parallelism provides an upper limit on the potential speedup of synchronous parallel simulation, since this is the average number of events that can be executed simultaneously, assuming an infinite number of processors and no cost for communication and synchronization. If there is little parallelism, then the other issues are not important. No matter how good the partition and communication overhead is, without sufficient parallelism, parallel synchronous event-driven switch-level simulation is doomed.

1.1 Timing Models

There are several different timing models that are commonly found in logic-level simulators. Three are variable-delay, fixed-delay and unit-delay. Each model causes a different simulation strategy to be employed. Thus, we need to understand these models in order to discuss their effects on circuit parallelism.

Variable-delay simulators generally provide the most reliable timing information of the three models. Each simulator event, a node changing value, is queued with a specific delay which depends on both the circuit topology and the characteristics of the current state of the circuit. There is a wide variability in the delay times that are used, and in principle, there are an infinite number of delays available for use. The most widely used switch-level timing simulator is RNL.

Fixed-delay simulators use a relatively small fixed number of delays in the circuit. These simulators are primarily gate-level simulators, with each gate type having identical delays, although there may be several delays per gate. For example, there are often different rising and falling delays for each gate type. The simulator may also support multiple gate types for a family of gates. For example, there may be several AND gate types, representing different speeds. The delays depend on the dynamics of the circuit only to the extent that they depend on the node values, but they do not depend on the topology of the circuit. Lsim, a mixed gate and switch-level simulator, is an example

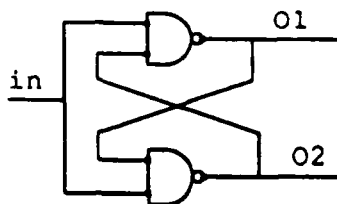


Figure 1.1: Comparing Unit-Delay and Pseudo Unit-Delay Timing

fixed-delay simulator [Chamberlain 86].

Unit-delay simulators provide a simple delay mechanism at the expense of providing timing. For a gate-level simulation, the definition of unit-delay timing is simple. At each timestep, all gates whose input(s) have changed are evaluated using the current values of the nodes, and then all of the resulting new outputs are updated and their gates are queued for the next timestep. The two important issues here are that *every* gate takes one timestep to change, and that all of the resulting node changes take effect simultaneously. For switch-level, a similar algorithm is used, except a gate evaluation is replaced by the evaluation of a transistor group, a set of nodes which are connected via transistor sources and drains. MOSSIM II, SwitchSim, and COSMOS are examples of switch-level unit-delay simulators.

For completeness, we also discuss pseudo unit-delay timing. This timing model is analogous to unit-delay timing, with the exception that node changes are imposed on the circuit as soon as they are evaluated. This means that node changes within a timestep take place incrementally instead of simultaneously. Because the node changes take place incrementally, the event sequence depends on the order that events are placed on the queue, and in some cases, this can affect the outcome of the simulation. For example, consider the example circuit in Figure 1.1 [Bryant 81, Terman 83]. Assume that in the initial state of the circuit the input is 0 and both outputs are 1. We want to change

the input to 1 and see how this affects the output. With unit-delay, the next state is 0 for both outputs since in the computation both previous outputs are used. The outputs will then return to 1 and will continue to oscillate forever. In the pseudo unit-delay algorithm, one output, say $O1$, will be evaluated first, and its value will be changed to 0. Then when $O2$ is evaluated, the value of $O1$ is 0, so its value doesn't change. This provides a stable solution, with one output staying at 1 and the other one making a transition to 0. Which output changes, however, depends on the evaluation order. If $O2$ is evaluated first, the output values are reversed.

1.2 Synchronization Strategies

Early parallel logic-level simulators were generally synchronous hardware simulation engines [Blank 84, Denneau 83]. In these systems, all processors are synchronized at the end of each timestep. If the circuit activity is uneven, then some processors are idle while others finish their computations. Because optimal speedup requires the activity to be spread out during each timestep, as opposed to spreading out activity over the entire simulation step, partitioning is critical and poor partitioning can greatly reduce the efficiency of the simulation.

Asynchronous strategies attempt to reduce the synchronization overhead and also to relieve the global event queue bottleneck. In asynchronous algorithms, each processor has its own simulation clock which may differ from the simulation clocks of other processors. Even though the clocks are distributed, the strategies must ensure that the resulting simulation produces the same results as a synchronous simulation. There are two basic techniques for assuring this: conservative and optimistic strategies.

The conservative strategies were pioneered by Bryant [Bryant 77] and Chandy and Misra [Chandy 81]. Here the simulation clock can only proceed if it is sure that no other events will arrive with timestamps less than the current clock time. This means that it "knows" that the events are processed in the correct sequence. The problem with this strategy is that the system can deadlock since all processors may be waiting

for messages from other processors before continuing. Thus a deadlock detection and recovery scheme is necessary, or there must be schemes to avoid deadlock. The deadlock avoidance mechanisms generally involve passing NULL messages to ensure that time can always proceed.

Optimistic strategies were pioneered by Jefferson [Jefferson 85]. Here a processor continues to process events, even if there may be later arriving events with smaller simulation times. Periodically, the processor also checkpoints its state. When an event arrives with a simulation time smaller than the current simulation clock (i. e. an event arrives in the past), the processor rolls back its state to a time less than this time, and cancels all erroneous events produced by the premature processing of events. The rationale for this approach is that (1) some of the time this strategy will proceed correctly whereas the conservative strategy would idle, and (2) the time it will spend processing erroneous events would be spent idling in the conservative strategy. However, there is additional overhead in this strategy required to checkpoint the state (in both time and space), and for rollback and cancellation of erroneous messages.

1.3 Contributions of This Work

In this dissertation we discuss the issues of circuit parallelism and the potential of parallel simulation, focusing on logic-level simulation as opposed to circuit-level simulation. We provide a formal model for comparing logic-level parallel simulation using three timing strategies and three synchronization strategies. Assuming an infinite number of processors, we show that for synchronous simulation, unit-delay timing provides the greatest speedup, and that for a given timing strategy, the conservative asynchronous strategy performs better than the synchronous strategy. We also show that for a fixed number of processors, there are cases where the optimistic strategy is better than the conservative strategy and vice versa. The asynchronous strategies should not provide a great increase in speedup for unit-delay timing, since if the event evaluation times are all equal, the conservative asynchronous strategy provides the same speedup as the synchronous strat-

egy. We also use the formal model to show that using a synchronous strategy and a given circuit family, the percentage of parallelism may change as the size of the circuit increases.

In addition to the formal model, we provide a methodology for measuring circuit parallelism. We use two simulators to demonstrate the methodology, and provide empirical results to corroborate the model. We find that while the model abstracts many specific characteristics of the simulators, it still provides reasonable results.

Finally we provide measurements that show how much parallelism is available in VLSI circuits using a synchronous strategy. Since we do not expect asynchronous techniques to provide markedly faster simulation times than the unit-delay measurements, and they may be much lower for variable-delay simulation, we can use the unit-delay measurements to estimate the potential speedup. These measurements range from 35 to 593. These figures represent the speedup one can obtain, assuming no communication and synchronization cost, perfect partitioning, and an infinite supply of processors. We believe that these measurements show that there is a small amount of parallelism available for exploitation in parallel simulation, and only a small number of processors can be effectively used to speed up sequential logic-level simulation.

1.4 Thesis Organization

In this dissertation we first present the theoretical results, and follow this with empirical data. In particular, the dissertation is organized as follows. In Chapter 2 we provide a summary of related work. Chapter 3 contains the formal portion of the thesis. Here we define the formal model for considering circuit parallelism and use it to investigate ways of speeding up logic-level simulation. The remainder of the dissertation is spent in evaluating the model via empirical results. Chapter 4 lays the foundations for the empirical studies by describing a methodology for measuring circuit parallelism. Chapters 5 through 7 then analyze portions of the model using empirical results. Finally, in Chapter 8 we conclude and discuss future work in this area.