Report Number: WUCS-86-19

1986-12-01

Performance Analysis and Design of a Logic Simulation Machine

Ken Wong and Mark A. Franklin

The high costs associated with logic simulation of large VLSI based circuits has led to the need for new computer architecture tailored to the simulation task. Such architectures have the potential for significant speed-ups over software-based logic simulators executing on standard sequential computers. This paper presents a model of one class of multiprocessor simulation architectures and compares the performance of some of these machines using data obtained from simulation of VLSI circuits. In addition, we discuss the implications of our results on machine design and examine the sensitivity of the model to variations in circuit characteristics.
... Read complete abstract on page 2.

Performance Analysis and Design of a Logic Simulation Machine

Ken Wong and Mark A. Franklin

Complete Abstract:

The high costs associated with logic simulation of large VLSI based circuits has led to the need for new computer architecture tailored to the simulation task. Such architectures have the potential for significant speed-ups over software-based logic simulators executing on standard sequential computers. This paper presents a model of one class of multiprocessor simulation architectures and compares the performance of some of these machines using data obtained from simulation of VLSI circuits. In addition, we discuss the implications of our results on machine design and examine the sensitivity of the model to variations in circuit characteristics.

PERFORMANCE ANALYSIS AND DESIGN OF A LOGIC SIMULATION MACHINE

Ken Wong and Mark Franklin

WUCS-86-19

December 1986

Computer and Communications Research Center
Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

This research was sponsored in part by funding from NSF Grant DCR-8417709 and ONR Contract N00014-8D-C-0761.

To be presented at the 14th International Symposium on Computer Architecture.

For Submission To
14th International Symposium on Computer Architecture

Performance Analysis and Design of a Logic Simulation Machine*

Ken Wong and Mark A. Franklin

Computer and Communications Research Center
Washington University
Campus Box 1115
St. Louis, MO 63130
(314) 889-6106

Abstract

The high costs associated with logic simulation of large VLSI circuits has led to the need for new computer architectures tailored to the simulation task. Such architectures have the potential for significant speed-ups over software-based logic simulators executing on standard sequential computers. This paper presents a model of one class of multiprocessor simulation architectures and compares the performance of some of these machines using data obtained from simulations of VLSI circuits. In addition, we discuss the implications of our results on machine design and examine the sensitivity of the model to variations in circuit characteristics.

Key Words

Logic simulation, logic simulation machines, design automation machines, hardware accelerators, performance evaluation.

---

* This research was sponsored in part by funding from NSF Grant DCR-8417709 and ONR Contract N00014-8D-C-0761

Performance Analysis and Design of a Logic Simulation Machine*

Ken Wong and Mark A. Franklin
Computer and Communications Research Center
Washington University
St. Louis, Missouri

1. Introduction

The high cost associated with the detection and correction of design errors once a VLSI chip has been fabricated has led to an increased reliance on simulation techniques in the logic design process. Logic simulation is used extensively to initially verify logic correctness and subsequently to develop vectors for testing fabricated chips. As circuit complexity has grown, the time delays and costs of performing logic simulation on standard, serial computers have grown until they can consume months of machine time [PF82].

These high costs have led to the development of a number of special-purpose processors dedicated to logic simulation [DE82, HO83, ZY83, VA84, HE85, SI85, XC86]. Such processors typically perform simulations at 10 to 1000 times the speed of standard, general-purpose computers. The techniques employed in achieving these speed-ups vary from microcode implementation of simulation algorithms to the development of special-purpose multiprocessors tailored to the simulation algorithm. In addition to the approaches found in commercial simulation engines, other possible simulation architectures have also been proposed [FR84, AS85, HA85].

In general, it has been difficult to compare these alternative approaches because data on the simulation process (e.g., event distributions) is not generally available in the open literature, and developing reasonable performance models over a range of complex architecture alternatives involves as much art as science. Furthermore, manufacturer-supplied speed

---

* This research was sponsored in part by funding from NSF Grant DCR-8417709 and ONR Contract N00014-8D-C-0761.

estimates are typically based on ideal circuit characteristics and other factors which are generally not under the control of the machine designer.

This paper shows that with a small number of special-purpose processors and a medium performance communication network, logic simulation can be accelerated by factors of hundreds and even thousands. These estimates were obtained from our model of multiprocessor simulation architectures whose workload characteristics were derived from data obtained from actual VLSI circuit simulations [WO86]. The development of this model is the central topic of this paper. The architecture performance model increases our capability to distinguish between good and poor designs. It also allows us to develop rules of thumb for designing multiprocessor simulation architectures and to identify candidates which should be evaluated in greater detail.

The next section reviews several acceleration techniques employed in logic simulation architectures. The section ends with a description of a simulation architecture taxonomy which defines broad classes of alternative architectures, one of which is analyzed in detail in this paper. Section 3 develops a run-time model for this class of simulators, and Section 4 derives a speed-up expression for comparing the relative performance of two architectures. Section 5 presents the data collected from simulations of five benchmark circuits. Section 6 presents some simplified performance bounds which result when synchronization and communication costs are ignored. Section 6 estimates the expected speed-up for five VLSI circuits running on a variety of hardware and discusses the implications of the results. The final section presents conclusions and discusses future research.

2. Logic Simulation Architectures

Numerous architectural options are available to the system designer for accelerating the logic simulation task. These options fall into the two broad categories shown in Table 1. This section reviews these techniques and describes in detail the architecture of the machine analyzed in this paper.

Functional Specialization	- Special event-list hardware - Special function evaluation hardware - Special hardware for net-list operations - Other special hardware
Concurrency Exploitation	- Multiple processors - Pipelining

Table 1: Logic Simulation Architectural Speed-up Techniques.

Functional specialization refers to those techniques which implement a subtask of the sequential, simulation algorithm in hardware or firmware to decrease the execution time. Prime candidates for functional specialization are event-list manipulation, function (gate/switch) evaluation, and net-list searching since these operations typically account for over 85% of the time consumed by a logic simulation algorithm [WO86].

Concurrency exploitation refers to the use of hardware structures which take advantage of either the actual concurrency inherent in the simulated circuit itself or the potential concurrency in the simulation algorithm. Concurrency exists in the simulated circuit as signals which propagate simultaneously over multiple paths in the circuit. This concurrency can be exploited by distributing the circuit components over multiple processors and letting each processor simulate different parts of the circuit. Potential concurrency exists between stages of the simulation algorithm because the data can be organized so that the algorithm can be viewed as a pipeline of operations.

The architecture alternatives associated with concurrency exploitation can be further clarified by classifying the space of design alternatives. Three classification components common to a wide range of logic simulation architectures can be identified and used to develop a taxonomy (Table 2) of logic simulation architectures. For example, the category UI/GC/Q=4/P=4/L=5 is a machine which has a Unit Increment time advance mechanism, a Global Clock, four event-lists (Queues=4), four event/function evaluators (Processors=4), and a five-stage pipeline (Length=5) in each event/function evaluator. Details of an earlier version of this taxonomy and associated simulation architectures are given in FR84.

Time Control Mechanisms		
Time Advance	Unit Increment	UI
	Event-Based Increment	EI
Time Synchronization	Global Clock	GC
	Local Clock	LC
Number of Event-Lists	Q = 1, 2, ...	
Event/Function Evaluation		
Number of Processors	P = 1, 2, ...	
Pipeline Length	L = 1, 2, ...	

Table 2: A Taxonomy of Logic Simulation Architectures.

This paper considers the architecture class UI/GC/Q=P/P/L (Figure 1) where the number of event-lists Q is assumed to be equal to the number of processors P, and both the number of processors and the number of stages L in the pipeline are design variables. A representative of this machine class is the ZYCAD LE-series logic simulation machines [ZY83]. The master processor maintains the global clock time C and synchronizes slave processors through a START signal which marks the beginning of a new unit time increment. There are P slave processors which each contain an event/function evaluator and an event-list. Each slave processor evaluates events scheduled for time C, communicates state information to other slave processors over a communication network, and sends a DONE signal to the master processor when it has completed processing all of its events scheduled for time C. Communication buffers between the slave processors and the communication network allow message transmission to proceed in parallel with event/function evaluation.

A *simulation cycle* begins when the master processor sends a START signal to the slave processors and ends when all slave processors have returned DONE signals to the master processor. When the master processor receives all of the DONE signals, it increments the global clock and starts the next simulation cycle by sending another START signal to the slaves. Simulation cycles are repeated until the clock reaches the termination time. Note that even when there are no events scheduled for a time point, the simulation cycle is still executed but consists of only the sending of START and DONE signals.

This general architecture was also analyzed by Levendel, Menon, and Patel [LE82]. Our model resembles theirs but considers a broader range of parameters, uses a workload model based on real data, includes pipeline evaluator length as a design parameter, and employs a random partitioning strategy in its communications model.

3. The Basic Model

The model presented in this section is developed in a top-down manner starting with a simple model which shows the primary contributors to simulation run-time. The section begins by describing the input (supplied) variables and design parameters in the model. Then, the processor and communication time components are detailed. Next, a model of the message volume as a function of the number of processors is developed for the case when the circuit components are randomly distributed among P processors. Finally, the effect of processor pipelining is added to the model. The performance measure used is the run-time of a simulation which generates E events.

3.1. Model Parameters

The output (dependent) variables, input (independent) variables, auxiliary variables, and design parameters of the model are shown in Table 3 below. The output variables are our performance measures of interest and are functions of the input variables. Values for the input variables can be obtained from measurements of a conventional, sequential simulator. The design parameters included in this model are the number of processors (P), the length of (number of stages in) the event/function evaluation pipeline (L), the average width of the communication network (degree of communication concurrency) during peak load (W), and the times for event/function evaluation (t_E), communication of one event message (t_M), and clock synchronization (t_S, t_D). Auxiliary variables denote values which are used to simplify expressions and are functions of other variables.

Variable	Type	Definition
R_P	Output	Simulation run-time using P processors
S_P	Output	Simulation speed-up over 1 processor using P processors
B	Input	Number of busy ticks
I	Input	Number of idle ticks
E	Input	Number of event/function evaluations
M_∞	Input	Number of messages when P=∞
β	Input	A measure of the degree to which work is unevenly distributed across processors (β=1 means even distribution, and β>1 means uneven distribution)
P	Design	Number of processors (event/function evaluators)
L	Design	Number of stages in the pipeline
W	Design	Average width of (number of concurrent messages which can be transmitted over) the communication network
t_E	Design	Average time to perform one event/function evaluation
t_M	Design	Time to transmit one event message over the communication network
t_S	Design	Time to send a START signal
t_D	Design	Time to send a DONE signal
F	Aux.	Average fanout (=M_∞/E)
N	Aux.	Average event simultaneity (=E/B)
T	Aux.	Speed-up due to functional specialization and technology improvements.

Table 3. Variable Definitions.

In a machine with a unit increment clock, the simulation clock runs for B+I simulation ticks (cycles) during which at least one event evaluation is performed in each of B (Busy) ticks and none during I (Idle) ticks. There are a total of E event evaluations which are distributed over the B busy ticks and P processors. When there are multiple processors, the number of event evaluations may be distributed unevenly across the P processors resulting in one processor having to perform more event evaluations than the other processors. During each busy tick, the most heavily loaded processor takes, on average, β times longer to perform its evaluations than it would if the evaluations were evenly distributed across all processors.

The average event/function evaluation time is t_E. Although some evaluations may take more time than others, we assume that all event/function evaluations take the average event/function evaluation time and that this time is invariant with the number of processors. This assumption assumes the existence of a near-constant-time event-list management

capabilities [UL78].

During the I idle ticks, the system must still spend some time (perhaps small compared to the event evaluation time) to skip over the idle ticks. This involves the transmission of the START signal by the master processor and the processing of the subsequent DONE reply from the slave processors, two activities which also occur during busy ticks. The time for these two activities is t_S and t_D.

A change at the output of a component is an event (representing a signal change) which needs to be propagated to the inputs of the fanout components (those receiving the new signal value). The propagation of the event requires one message to be sent to each fanout component and will take t_M time units for each message transmission. For simplicity, we have ignored approaches which transmit multiple events in one message transmission. During the course of the simulation, M_P such messages are transmitted over the communication network when there are P processors. The variable M_∞ denotes the limit of M_P as the number of processors P approaches the number of circuit components. When there is more than one processor trying to communicate across the communication network at the same time, only W messages can be transmitted concurrently. All other messages are queued until the network can accept more messages.

3.2. The Model Without Pipelining

The following model is a simple mean-value model of the selected architecture when an evaluation pipeline is not employed. Let t_{EVAL} be the average time required by each of P processors to evaluate all events and functions during a busy tick, and let t_{COM} be the average time required to transmit all messages generated during a busy tick. Then, if communication and event/function evaluation can proceed concurrently, the time required to simulate one busy tick is approximately the maximum of t_{EVAL} and t_{COM}. The overhead required by each processor to START and stop (i.e., DONE signal) a simulation cycle is the processor synchronization time t_{SYNC}. The total simulation run-time R_P is:

$$R_P = B\,(t_{SYNC} + \max\,(t_{EVAL}, t_{COM})) + It_{SYNC} \qquad (1)$$

This expression for the run-time makes several simplifying assumptions:

1) Event evaluations and message transmissions are evenly distributed over the B busy ticks.
2) When communication dominates event/function evaluation, message transmission completely overlaps event/function evaluation.
3) Evaluations begin simultaneously (i.e., all slave processors simultaneously receive a broadcasted START signal).

Machine speed is limited by the most heavily loaded physical component. In principle, this bottleneck will be either 1) the slave processors, 2) the communication network, or 3) the master processor. For example, when the communication network can process messages faster than the event/function evaluators can generate them and little time is required for synchronization, the slave processors will saturate, and the run-time will be approximately Bt_{EVAL}. However, if the communication network is slow compared to the processors, the communication network will saturate, and the run-time will be approximately Bt_{COM}. Finally, if synchronization dominates the run-time, the run-time will be approximately $(B+I)t_{SYNC}$. When synchronization is fast, the design problem is to balance the number and speed of the event/function evaluators with the communication network so that most of the hardware is utilized near its capacity at minimum cost.

If the E event/function evaluations are evenly distributed over the B busy ticks and P processors, each processor will perform E/(BP) evaluations during each busy tick so long as there is at least one event for each processor ($P \leq E/B$). Let N=E/B, the maximum degree of parallelism. When P=N and the load is evenly distributed, all processors will have one event/function evaluation per busy tick. Adding more processors will not reduce the run-time since there will be N processors evaluating one event and P−N idle processors. Thus, designs with more than N processors are not considered. If instead, the N evaluations in each busy tick were unevenly distributed over the P processors, the processor with the most evaluations will perform $\beta N/P$ evaluations. Since t_E is the time for one evaluation, the time required for all

evaluations during one busy tick is the time required for the most heavily-loaded processor to finish, or

$$t_{EVAL} = \beta \frac{N}{P} t_E, \qquad P \leq N,\ 1 \leq \beta \leq P,\ N = E/B \tag{2}$$

When β is one, perfect balance has been achieved while values of β greater than one indicate imbalance ($\beta = P$ means all evaluations are performed on one processor).

If M_P is the number of message transmissions when there are P processors, and these are evenly distributed over the B busy ticks, there will be M_P/B transmissions during each busy tick. With concurrent transmissions evenly spread over the W communications channels and t_M the mean time required to transmit one message, the time required for message transmission during one busy tick is

$$t_{COM} = \frac{M_P}{WB} t_M \tag{3}$$

In a single time-shared bus configuration, W is equal to one. Other communication networks like the crossbar and delta have W-values greater than one and are a function of the number of processors, the message traffic distribution, and the network design parameters.

Synchronization involves the broadcasting of a START signal to all slave processors and the subsequent sending of DONE signals from the slave processors. If the time required to begin a simulation cycle is t_S, and the time required to end a simulation cycle is t_D, the synchronization time is

$$t_{SYNC} = t_S + t_D \tag{4}$$

During busy ticks, DONE signal transmission can overlap either event/function evaluation or other DONE signal transmissions. During idle ticks, DONE signal transmissions overlap each other.

Substituting (2), (3), and (4) into (1) yields the run-time model for the selected architecture without pipelining.

$$R_P = (B+I)(t_S+t_D) + \max\left(\beta\frac{E}{P}t_E, \frac{M_p}{W}t_M\right), \qquad P \leq E/B \tag{5}$$

The model shows that when event/function evaluation is the dominant time component, the run-time decreases as the number of processors P increases (i.e., linear speed-up). When communication is the dominant component, the run-time will depend on the number of message transmissions M_P, the communication concurrency W, and the message transmission time t_M.

3.3. Random Partitioning

The message volume M_P depends on the number of processors P and how the circuit components are distributed over the P processors. In the random partitioning scheme, components are randomly distributed over the P processors. Although there may be applications where a better partitioning strategy can be found (e.g., placing highly connected components together on the same processor), the message volume associated with random partitioning can be taken as an upper bound for other useful partitioning strategies.

A message arises when the output of a component located on processor i must be propagated to a fanout component located on a different processor j ($i \neq j$). Since there are C/P components on each processor, the fanout component could be any of the other (C/P)−1 components on processor i or the C−(C/P) components on the other processors. Because of random partitioning, the fanout component could be any of the other C-1 components with equal probability. Since there are M_∞ signal propagations (message transmissions) when there are C processors, the number of messages transmitted when there are P processors is given by:

$$M_P = M_\infty \frac{(C-(C/P)}{(C-1)} \approx M_\infty(1-1/P), \qquad C \gg 1 \tag{6}$$

Note that the number of messages transmitted across the communication network is 0 when there is one processor and increases with increasing P until it is equal to M_∞ when each component is placed on a separate processor.

3.4. Pipelining

Event/function evaluation speed can be increased by using a pipelined architecture. The logic simulation algorithm allows about six stages [AB83] with a smaller number possible by merging adjacent stages. If an L-stage pipeline can perform one event/function evaluation in time t_E, it can perform n of these operations in time approximately equal to

$$t_n = \frac{t_E}{L}(n+L-1) \tag{7}$$

if the execution time of each stage is equal to t_E/L. This expression can be derived by noting that the n-th operation must wait for the n-1 operations ahead of it to complete the first stage of the pipe and then traverse L stages to get to the output of the pipe. Thus, it takes n−1+L stage completions before the n-th operation exits the last stage of the pipe. For typical average fanouts (F≈2), it is possible and advantageous to divide the stages into nearly equal execution times so that the maximum output rate can be achieved.*

The processor performing the largest number of evaluations during a busy tick must perform $n=\beta E/(BP)$ evaluations. The evaluation time can now be given as:

$$t_{EVAL} = \frac{t_E}{L}(n+L-1), \quad P \leq E/B \tag{8}$$

$$n = \beta\frac{E}{BP} \tag{9}$$

Note that when L equals 1, (8) reduces to (2), the case when no pipelining is employed. The run-time model with pipelining is obtained by substituting (6) into (3) and then (3), (4), and (8) into (1):

$$R_P = (B+I)(t_S+t_D) + \max\left(\frac{Bt_E(n+L-1)}{L}, \frac{M_\infty(1-1/P)}{W}t_M\right), \quad P \leq E/B \tag{10}$$

---

* Table III in AB83 indicates near-equal loading is possible for an average fanout of two which is typical of the circuits that were in our benchmark. When fanouts are much greater than two, the function evaluation stage of the pipeline will be the bottleneck and its (and the pipeline's) output rate will be smaller than the L/t_E predicted by our model.

4. Speed-Up

Designers of special-purpose machines are usually interested in comparing the performance of their machine with that of an unenhanced or base machine, with the relative speed expressed as a speed-up factor. If the run-times of the base and special-purpose machines for identical simulations are R_B and R_P respectively, the speed-up is

$$S_P = R_B/R_P \tag{11}$$

where R_P is given by (10) and R_B is given by (12). Typically, the base machine is a standard, general-purpose machine (e.g., VAX 11/750*). The time required by the base machine to evaluate E events is

$$R_B = Et_{E.B} \tag{12}$$

where $t_{E.B}$ is the time required to perform one event/function evaluation on the base machine†.

One aspect of this speed-up may be due to a decrease in the event/function evaluation time resulting from the use of functional specialization and technology improvements in the special-purpose machine. The parameter H reflects this reduction in t_E and is defined as

$$H = t_{E.B}/t_{E.S} \tag{13}$$

Two interesting bounds can be derived from (11) by simplifying (10) for the evaluation-time-dominant and communication-time-dominate cases and substituting (10) and (12) into (11). When the event/function evaluation time dominates all other time components in the special-purpose machine ($t_{EVAL} \gg t_{COM}, t_{SYNC}$) and the processor load is balanced (β=1), the speed-up becomes

$$S_P^* = \begin{cases} \dfrac{HNL}{N/P+L-1}, & P \leq N \\ HN, & P \geq N \end{cases} \tag{14}$$

---

* VAX is a trademark of Digital Equipment Corporation.

† The *dot notation* will be used to distinguish between the parameters of the base and special-purpose machines (e.g., $t_{E.B}$ and $t_{E.S}$ denote t_E for the base and special-purpose machines respectively).

where N=E/B. Note that the use of pipelining increases the speed-up by a factor of L, and the overall speed-up is approximately S_P^*=HLP when the processors are heavily loaded (N/P≫L). When the processors are lightly loaded (N/P≪L), the end effects of filling and emptying the pipe limit the speed-up to HN.

If instead, the communication time dominates all other time components ($t_{COM} \gg t_{EVAL}, t_{SYNC}$) and the processor load is balanced (β=1), the speed-up becomes

$$S_P^\dagger = \frac{EW(t_{E.B}/t_M)}{M_\infty(1-1/P)} \tag{15}$$

Note that in this case, the speed-up decreases as the number of processors increases and approaches a limiting value of

$$\lim_{P\to\infty} S_P^\dagger = \frac{EW(t_{E.B}/t_M)}{M_\infty} \tag{16}$$

Since the communication network is operating at maximum capacity, the increase in the message volume which accompanies greater partitioning (larger P) is directly reflected in a lengthening of the run-time of the special-purpose machine and therefore a decrease in the speed-up. Equations (14) and (15) indicate that the general shape of the speed-up curve as a function of the number of processors will first show an increase in the speed-up as a function of the number of processors until the system becomes limited by the capacity of the communication network. Then, the curve will asymptotically decrease toward the value given in (16).

5. Simulation Data

Data for our model was collected from standard, serial simulations of five VLSI circuits. This section describes the circuits, presents a normalization procedure for estimating the workload parameters as a function of circuit size, and summarizes the benchmark data for circuits scaled to 100,000 components.

The data was gathered using the *lsim* gate/switch-level logic simulator running on a

VAX 11/750. *Lsim* is a UNIX*/C-based simulator and was designed with data collection on the simulation process in mind [CH85, CH86a]. It can simulate systems containing both traditional unidirectional logic gates, and bidirectional MOS switches. Although *lsim* supports three types of delay models, the data presented in this paper were from the fixed delay model in which component delays are modeled by fixed low-to-high and high-to-low propagation times.

Random test vectors were applied to the circuits until aggregate statistics (e.g., average event-list size, circuit activity) remained stable and most components experienced at least one output change.

The five circuits in our benchmark were: 1) a stop watch, 2) a priority queue, 3) an associative memory, 4) a Radiation Treatment Planning (RTP) chip, and 5) a crossbar switch. The stop watch circuit determines the elapsed time between a start and a stop signal. The priority queue stores 48-bit records, each divided into four fields, and retrieves the record whose first field contains the smallest value (i.e., an event queue). The associative memory functions like a normal random access memory as well as a memory in which records can be retrieved by content (i.e., those matching a specified pattern). The RTP chip implements an algorithm used in cancer treatment planning which calculates the radiation dosage at a specified point. The crossbar switch provides an interconnection network between four input and four output ports.

These circuits reflect a mix of characteristics (Table 4) and are the product of five graduate student design teams. The two most prevalent VLSI technologies (nmos and cmos) and clocking schemes (synchronous and asynchronous) are represented. The circuit sizes range from approximately 650 transistors to 8,000 transistors. The priority queue, associative memory, and crossbar switch were designed so that they could be scaled to larger versions as required (assuming no pin or power limitations). The test circuits were kept small enough to insure that simulation run lengths were reasonable and disk storage availability was adequate. The Switches and Gates columns in Table 4 indicate the number of *lsim* bidirectional switches

---

* UNIX is a trademark of AT&T Bell Laboratories.

and unidirectional gates used in defining the circuit. The right column reflects the total number of transistors in each circuit.

Circuit	Tech.*	Type*	Switches	Gates	Total	Approx. Trans.*
Stop watch	nmos	sync	216	131	347	650
Assoc. memory	nmos	async	296	454	750	1,700
Priority queue	cmos	sync	2,960	720	3,680	5,100
RTP chip	nmos	sync	1,422	1,746	3,169	6,100
Switch	nmos	async	0	2,648	2,648	8,000
Average			979	1,140	2,119	4,300

* *Tech*nology, *sync*hronous, *async*hronous, *Approx*imate number of *trans*istors

Table 4: Circuit Characteristics.

Table 5 shows the circuit dependent data used in our model for circuits with 100,000 components. The data was obtained by linearly scaling the measured data to represent 100,000 component circuits. For example, since the priority queue has 3,680 components, a priority queue with 100,000 components will have values of E and M_∞ which are 100,000/3,680=27.2 times larger than measured values. Since the measured E-value for the priority queue was 592,206, the scaled value is approximately 16.1×10^6.

Circuit	X*	B	I	E (millions)	M_∞ (millions)
Stop Watch	288.2	4,587	515,414	15.1	33.3
Assoc. Mem.	133.3	3,140	25,061	2.9	11.0
Priority Q.	27.2	10,620	57,631	16.1	24.5
RTP Chip	31.6	10,225	55,274	5.8	7.8
CB Switch	37.8	155,000	480,189	12.5	25.1

* X = 100,000/(Component Count)

Table 5. Model Data Normalized to 100,000 Components.

Some statistics which give an indication of the nature of logic simulation using the fixed delay model are given in Table 6. B/(B+I) is the fraction of time points with one or more scheduled events. N=E/B is the average number of simultaneous events (when there is at least one event). The activity is the average fraction of components with output changes. $F=M_\infty/E$

is the average fanout. The general picture which emerges from the data is that nothing is happening at most simulation time points (B/(B+I) is small)*. When there is activity, only a small fraction of the circuit is involved (the activity over busy time points N/(100,000 components) is small). However, there is a substantial amount of event simultaneity at each busy time point (N=E/B is large) making the use of parallelism potentially rewarding. The simultaneity for synchronous circuits (all circuits except the crossbar switch) is greater than that for asynchronous circuits (crossbar switch) where the events tend to be spread over a greater fraction of the time points (higher value of B/(B+I)). The table also shows that, except for the crossbar circuit, attaining maximum acceleration exclusively from parallelism (N) will require processor populations in the hundreds and thousands.

Circuit	B/(B+I)	Sim. Ev.* N=E/B	Activity N/100,000	Fan Out $F=M_\infty/E$
Stop Watch	.0088	3,294	.033	2.2
Assoc. Mem.	.1113	938	.009	3.7
Priority Q.	.1556	1,517	.015	1.5
RTP Chip	.1561	567	.006	1.3
CB Switch	.2440	80	.001	2.0
Average	.1351	1,279	.013	2.1

* Sim. Ev.: Average number of simultaneous events

Table 6. The Nature of Logic Simulation.

6. Performance Bounds

Simple performance bounds give a quick indication of the potential speed-up offered by a machine architecture. Equation (14) is an expression for the speed-up which can be obtained when communication and synchronization costs are ignored and when events are evenly distributed over busy ticks and processors. This section shows how the performance bounds can

---

* The small value of B/(B+I) for the stop watch is somewhat misleading since the clock period for the stop watch was much larger than necessary and led to a large number of idle time points at the end of each clock period. The values for B/(B+I) for the other circuits are much more representative of the fixed delay model parameters.

be used to get a quick estimate of the speed-up potential of a variety of special-purpose, multiprocessor machines.

Figure 2 shows curves of the idealized speed-up S_P^* (event/function evaluation time dominant) as a function of the number of processors for a special-purpose machine running simulations of 100,000-component circuits from the benchmark. The machine has a five-stage pipeline (L=5) and has a processor whose single-event-evaluation time is 100 times faster than the base machine (H=100). The speed-up curves show a rapid increase in the speed-up as a function of the number of processors for small processor populations. In this region where the event simultaneity is much larger than the number of processors (N>>P), (14) indicates that the speed-up will be approximately HLP=500P. But as the number of processors increases, the event simultaneity will become comparable to the number of processors, and the speed-up curve will approach HN, independent of the number of processors. This limiting case is shown in Figure 2 for the crossbar switch where N=80, and thus the speed-up bound is HN=100(80)=8,000 when $P \geq 80$. The idealized speed-up limit HN for the other four circuits can be computed from the N-values in Table 6. Note that these limits are a function of the circuit size since the event simultaneity N increases (decreases) with increasing (decreasing) circuit size.

A simple calculation shows the potential for speed-up even when a special-purpose machine has only one processor but uses functional specialization and an evaluation pipeline. When the processor is heavily loaded, the speed-up is approximately $S_1^* \approx HL$. A special-purpose machine with H=10 and a five-stage pipeline (L=5) will yield a speed-up of approximately 50 for typical VLSI circuits ($S_1^* \approx 5(10)$). Since a typical general-purpose machine such as a VAX 11/750 evaluates one simple event/function in about 400 μsec (a speed of 2,500 events/sec), such a special-purpose, uniprocessor machine could have a speed of 125,000 events/sec. If the special-purpose machine uses a five-stage pipeline (L=5) with H=100, the speed-up becomes $S_1^*=500$ or 1.25M ev/sec (million events/sec). Larger speed-ups can be obtained at higher costs. Commercial experience suggests that H-values as large as 1000 may

be possible [BL84, PF82, ZY83]. However, five stages is about all that can be expected from pipelining the simulation algorithm*.

Further acceleration can be achieved through the use of multiple processors. When the number of simultaneous events is large enough to keep each processor heavily loaded ($N/P \gg L-1$), the speed-up is approximately $S_P^* \approx HLP$. All of the 100,000-component circuits in benchmark except the crossbar switch have values of N large enough to keep the processors in a 10-processor system with a five-stage pipeline heavily loaded.

The performance bounds discussed above are overly optimistic for several reasons. First, the workload will seldom be evenly distributed over all busy ticks, leading to idle processors during busy ticks when the number of processors is nearly equal to N=E/B. This inability to keep all processors busy is particularly damaging to the speed-up when a substantial portion of the speed-up is gained through parallelism and can lead to large errors in our speed-up estimates for large processor populations. Second, the workload will seldom be evenly distributed over all processors during a busy tick. This slowdown will probably be less than an order of magnitude under typical circumstances. Third, the distribution of events over multiple processors leads to communication costs which will eventually bend the speed-up curves downward when the communication component of the run-time becomes more significant than the event/function evaluation component. The effect of these communication costs is examined in the next section.

7. Machine Designs

The designer of a logic simulation machine is confronted with a wide range of design options. Since higher performance components typically have higher costs, poorly matched components can lead to unnecessarily expensive designs with unused capacity in parts of the system. This section presents design curves and performance estimates for machines with a

---

* Six stages are possible, but two of them have short execution times and can be combined without decreasing the event/function evaluation rate.

variety of parameter combinations. The results indicate both the machine speeds which can be expected as well as some of the resource requirements necessary for attaining the speed estimates.

7.1. Design and Workload Parameters

The factors considered in our design space are shown in Table 7. The imbalance factor β is assumed to be 1, and the synchronization time t_{SYNC} is assumed to be constant. In all cases, the time unit assumed on all time parameters (e.g., t_E, t_M) is one synchronization time interval (i.e., $t_{SYNC}=t_S+t_D$) called a *sync*. The base machine is assumed to be a VAX 11/750 running a standard, logic simulator with $t_{E.B}$=4,000 syncs where a sync=100 ns (i.e., $t_{E.B}$=400 μs).

Parameter		Possible Values
L	pipeline depth	1, 5
t_{SYNC}	synchronization time	1 sync
t_M	message trans. time	2 syncs, 3 syncs
W	communications concurrency	1, 2, 3
H	tech./specialization speedup	1, 10, 100
P	number of processors	1, 2, ... , 50

Table 7. Design Space.

The H-values were chosen to reflect the range from no acceleration in the low end, to acceleration due to microcoding or special hardware in the mid- and high-range [SM86]. The values for communication concurrency (W) correspond to values typical of one to three time-shared busses. The values for pipeline length (L) were chosen to correspond to no pipelining or full pipelining. The values for the event message transmission time are typical values for busses a few feet long, and the processor populations were chosen to reflect low to moderate populations.

The speed-up curves and estimates in this section were derived for an "average" benchmark circuit. The data for an average circuit is shown in Table 8 and was derived from Table 6 by normalizing the simulation run-length to 60,000 ticks (busy and idle combined). For

example, the average value of B/(B+I) from Table 6 is .1351. If we assume that B+I=60,000, the number of busy ticks is B=.1351(B+I)=8,106 and and the number of idle ticks is I=60,000–B=51,894. Then, since the average value of N=E/B is 1,279, E=NB=10.4 million. Finally, since the average value of M_∞/E is 2.1, M_∞=2.1E=21.8 million. The choice of 60,000 for the simulation run-length was arbitrary and does not affect any of our performance results.

Parameter	Value	Parameter	Value
B	8,106	I	51,894
E	10,367,574	M_∞	21,771,905

Table 8. Average Workload Characteristics.

7.2. A Comparison of Alternative Designs

Figures 3 to 5 show the speed-up for that half of the design space when t_M=3 syncs. Each figure shows the speed-up curves for the six designs derived from varying the pipeline length (L=1,5) and the communication concurrency (W=1,2,3) for a given performance range H. The number of processors range from 1 to 50.

The top curve in Figure 3 (H=1, t_M=3 syncs) is for the three cases when the machine is pipelined (L=5) and W=1, 2, and 3. The bottom curve is for the corresponding non-pipelined machine (L=1, and W=1, 2, and 3). Only two curves are shown because the communication concurrency has no effect on the speed-up when the processor population is less than or equal to 50. This insensitivity to the value of W indicates that the message rate arriving to the communication network is so low that there is still excess capacity even when W=1 so that greater communication concurrency will not improve the performance of the machines. As expected, the two curves in Figure 3 are approximately separated by a factor of L=5.

In Figure 4 (H=10, t_M=3 syncs), the top three curves are for the three cases when the machine is pipelined (L=5) and W=1, 2, and 3 with greater speed-up provided by the machine with the larger communication concurrency. The bottom curve is for the three non-pipelined machines (L=1, and W=1, 2, and 3).

The top three curves indicate that the pipelined processors are now supplying messages at a high enough rate to saturate the communication network. Once the communication network is saturated, (15) indicates that the machine speed-up is approximately proportional to the reciprocal of the communication time and therefore a decreasing (very slowly) function of the number of processors. Note further that it takes approximately twice as many processors to saturate the communication network with W=2 as it does when W=1.

The bottom curve is for the non-pipelined case and is identical to the bottom curve in the preceding figure except that the speed-up is 10 times larger. In fact all curves in this figure represent speed-ups which are 10 times larger than the corresponding curves in the preceding figure for processor populations which are small enough so that the system is not communication limited.

The effect of insufficient communication capacity is more dramatically shown in Figure 5 which shows the speed-up curves for the high-performance machines (H=100). All six of the high-performance designs have sufficient event/function evaluation speed to consume the entire capacity of the communication network even for small to moderate processor populations. There are six curves corresponding to the six designs. However, for very small processor populations ($P<3$), the speed-up is insensitive to the communication concurrency because there is excess communication capacity. On the other hand, for moderate processor populations ($P>10$), the speed-up is insensitive to the pipeline length because there is excess event/function evaluation capacity. The maximum speed-up occurs in the region between these two extremes.

The curves for the case when the message transmission time is $t_M=2$ syncs are not shown because they are qualitatively identical to the cases above where $t_M=3$ syncs. Quantitatively though, the faster communication network accelerates the communication-limited designs by a factor of approximately 1.5 using processor populations which are also approximately 1.5 larger.

Table 9 summarizes the data from the speed-up curves for the designs considered in Table 6. The processor populations shown in the table indicate the number of processors less than or

equal to 50 which yield the maximum speed-up for each design. The entries with the number of processors less than 50 reflect designs in which the event/function evaluation time is approximately equal to the communication time (i.e., a balanced system).

			t_M=3 syncs		t_M=2 syncs	
H	W	L	P	S_P	P	S_P
1	1	1	50	50	50	50
		5	50	216	50	216
	2	1	50	50	50	50
		5	50	216	50	216
	3	1	50	50	50	50
		5	50	216	50	216
10	1	1	50	50	50	500
		5	15	680	50	970
	2	1	50	50	50	50
		5	29	1,313	50	1,938
	3	1	50	50	50	50
		5	45	1,943	50	2,155
100	1	1	8	725	11	1,046
		5	2	992	3	1,426
	2	1	14	1,365	20	1,994
		5	4	1,689	5	2,373
	3	1	20	1,994	30	2,943
		5	5	2,373	7	3,317

Table 9. A Comparison of 36 Designs.

An estimate of the absolute event/function evaluation speed can be obtained by multiplying the entries in Table 9 by 2,500 ev/sec, the approximate speed of our base machine. For example, the fastest machine in Table 9 (H=100, W=3, L=5, and t_M=2 syncs) has a speed of 8.3M ev/sec.

8. Summary

This paper has analyzed a number of design alternatives for speeding up the simulation process. A performance model of a class of logic simulation machines was developed. The input parameters to the model included such items as number of events and number of messages and represent the circuit dependent aspects of the problem. The design parameters concerned

architecture choices such as number of processors, pipeline length, and hardware speed.

The paper presented data on the simulation process collected from fixed delay simulations of five VLSI circuits. The nature of logic simulation is one of low temporal and spatial activity occurring over large component populations. This leads to sufficient parallelism for concurrency exploitation to multiply speeds by two to three orders of magnitude for large circuits. However, the parallelism is highly dependent on the circuit.

A range of machine designs were examined and their speed-up over a conventional, uniprocessor base machine was estimated. The analysis showed that low-performance machines (i.e., H=1 to 10) could gain some acceleration through the use of either pipelining or a large number of processors but probably not both since large processor populations can reduce the effectiveness of the pipeline. High-performance machines (i.e., H=100) on the other hand make effective use of their pipelines but are limited by the capacity of the communication network.

The results indicate that the use of a moderate performance communication network limits the speed of a high-speed machine to around 8 million events/sec even though much more parallelism remains untapped. A faster communication network, better circuit partitioning, or another architecture will be required to significantly increase the simulation speed beyond this range.

Work on more accurate models which include statistical distributions and more accurate communication network performance models are underway. Also, related research on the circuit partitioning problem is also in progress. Experiments are being conducted to measure the performance of heuristics in reducing the communication volume. We are also developing simple performance models of other architectures. Asynchronous architectures may offer a higher degree of parallelism and may smooth out any effects due to low processor loading found in the machines analyzed in this paper. We are also looking at problems in the design of more general-purpose simulation machines for VLSI design automation. The problem here is to design a machine which can accelerate a variety of VLSI design automation functions using a small

number of powerful, integrated techniques [CH86b].

References

[AB83] M. Abramovici, Y. H. Levendel, and P. R. Menon, "A Logic Simulation Machine," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* CAD-2:2 (Apr. 1983), pp. 82-94.

[AS85] V. Ashok, R. Costello, and P. Sadayappan, "Distributed Discrete Event Simulation Using Dataflow," *Proc. 1985 Int. Conf. on Parallel Processing*, IEEE Computer Society Press, 1985, pp. 503-510.

[BL84] T. Blank, "A Survey of Hardware Accelerators Used in Computer-Aided Design," *IEEE Design and Test of Computers* 1:3 (Aug. 1984), pp. 21-39.

[CH85] R. D. Chamberlain, "Lsim: A Gate-Switch Level Logic Simulator," M.S. Thesis, Dept. of Computer Science, Washington University, St. Louis, MO., May 1985.

[CH86a] R. D. Chamberlain and M. A. Franklin, "Collecting Data About Logic Simulation," *IEEE Trans. on Computer-Aided Design* CAD-5:3 (July 1986), pp. 405-412.

[CH86b] R. D. Chamberlain and M. A. Franklin, "A Unified Approach to Mixed-Mode Simulation," Technical Report WUCS-86-20, Dept. Computer Science, Washington University, St. Louis, MO., Nov. 1986.

[CH81] K. M. Chandy and J. Misra, "Asynchronous Distributed Simulation via a Sequence of Parallel Computations," *Comm. of the ACM* 24:11 (Apr. 1981), pp. 198-206. CAD-4:3 (July 1985), pp. 239-250.

[DE82] M. M. Denneau, "The Yorktown Simulation Engine" *Proc. 19th Design Automation Conf.*, June 1982, pp. 55-59.

[FR84] M. A. Franklin, D. F. Wann, and K. F. Wong, "Parallel Machines and Algorithms for Discrete-Event Simulation," *Proc. 1984 Int. Conf. on Parallel Processing*, Aug. 1984, pp. 449-458.

[HA85] W. Hahn and K. Fischer, "MuSiC: An Event-Flow Computer for Fast Simulation of Digital Systems," *Proc. 22nd Design Automation Conf.*, July 1985, pp. 338-344.

[HA82] J. P. Hayes, "A Unified Switching Theory with Applications to VLSI Design," *Proc. of the IEEE* 70:10 (Oct. 1982), pp. 1140-1151.

[HE85] P. M. Hefferan, et. al., "The STE-264 Accelerated Electronic CAD System," *Proc. 22nd Design Automation Conf.*, 1985, pp. 352-358.

[HO83] J. K. Howard, R. L. Malm, and L. M. Warren, "Introduction to the IBM Los Gatos Logic Simulation Machine," *Proc. IEEE Inter. Conf. on Comp. Design (ICCD'83)*, Oct. 1983, pp. 580-583.

[LE82] Y. H. Levendel, P. R. Menon, and S. H. Patel, "Special-Purpose Computer for Logic Simulation Using Distributed Processing," *The Bell System Technical Journal*, Dec. 1983, pp. 2873-2909.

[MI86] J. Misra, "Distributed Discrete-Event Simulation," *ACM Computing Surveys* 18:1 (Mar. 1986), pp. 39-65.

[PF82] G. F. Pfister, "The Yorktown Simulation Engine: Introduction," *Proc. 19th Design Automation Conf.*, June 1982, pp. 51-54.

[SI85] Silicon Solutions Corp., "The Mach 1000 Simulation Engine," Product Description, Menlo Park, CA, 1985.

[UL78] E. Ulrich, "Event Manipulation for Discrete Simulations Requiring Large Numbers of Events," *Comm. of the ACM* 21:9 (Sep. 1978), pp. 777-785.

[VA84] Valid Corp., "Realfast Simulation Accelerator," Product Description, 1984.

[WO86] K. F. Wong, M. A. Franklin, R. D. Chamberlain, and B. L. Shing, "Statistics on Logic Simulation," *Proc. 23rd Design Automation Conference*, June 1986, pp. 13-19.

[XC86] Xcelerated Computer Aided Technology Inc., "MX and MXT Series," Product Description, Hopkins, MN, 1986.

[ZY83] Zycad Corp., "The Zycad Logic Evaluator," Product Description, N. Roseville, MN, 1983.

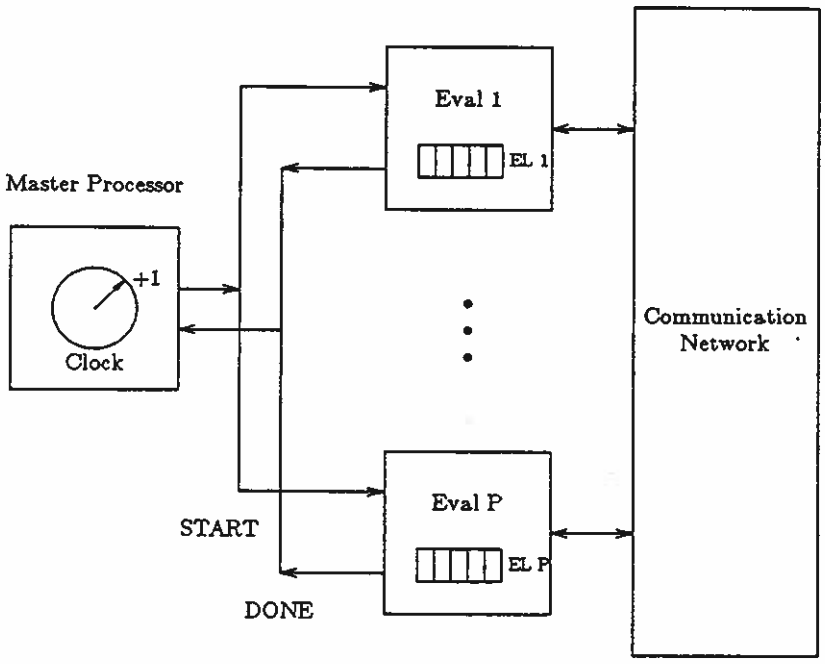

Figure 1. A Multiprocessor Logic Simulator.

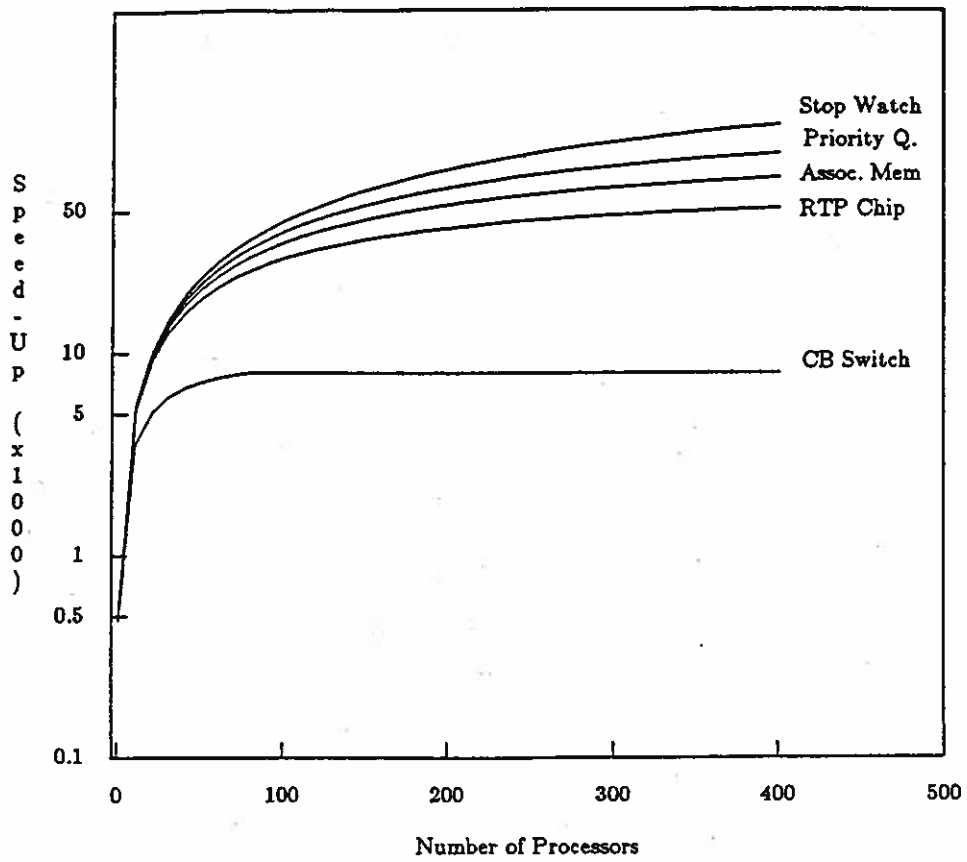

Figure 2. Ideal Speed-Up Versus Number of Processors.
(H=100, L=5, 100,000 Components)

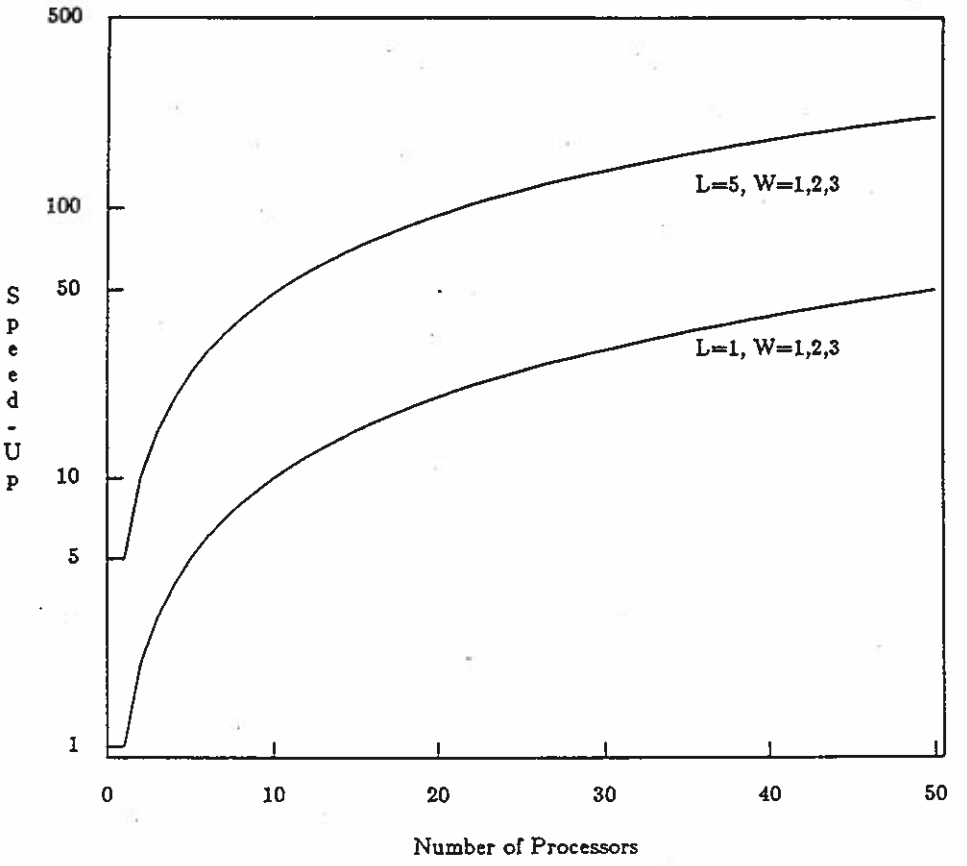

Figure 3. Speed-Up Versus Number of Processors (H=1,t_M=3 syncs)

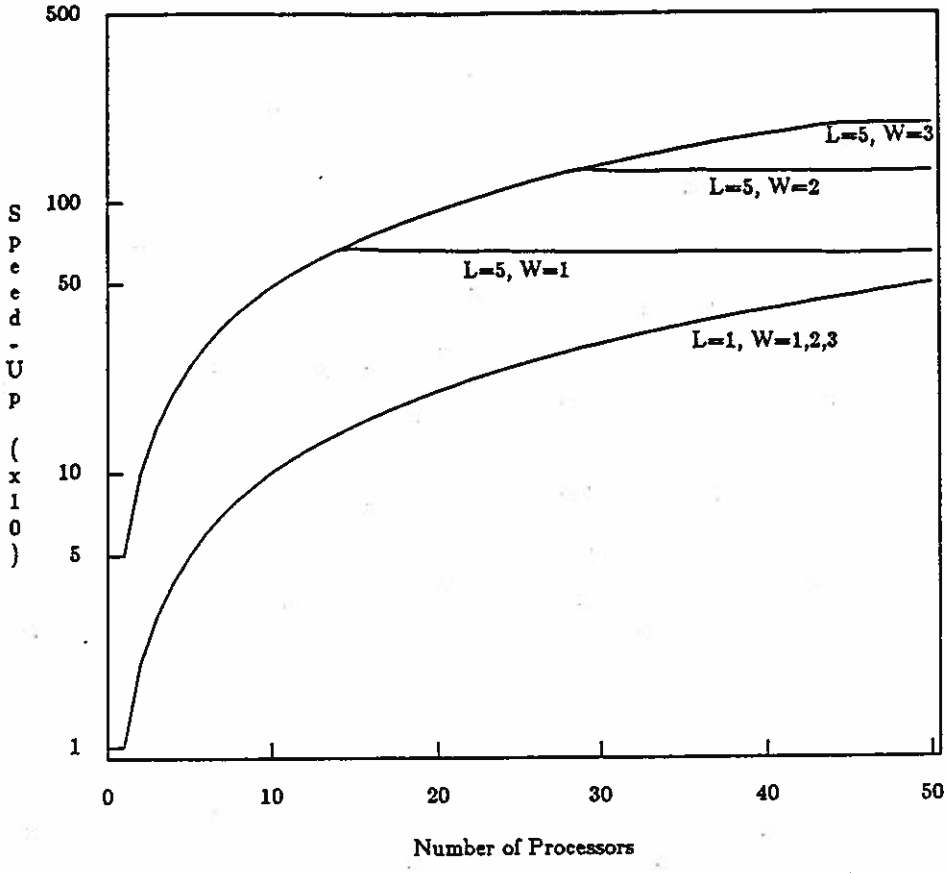

Figure 4. Speed-Up Versus Number of Processors (H=10,t_M=3 syncs)

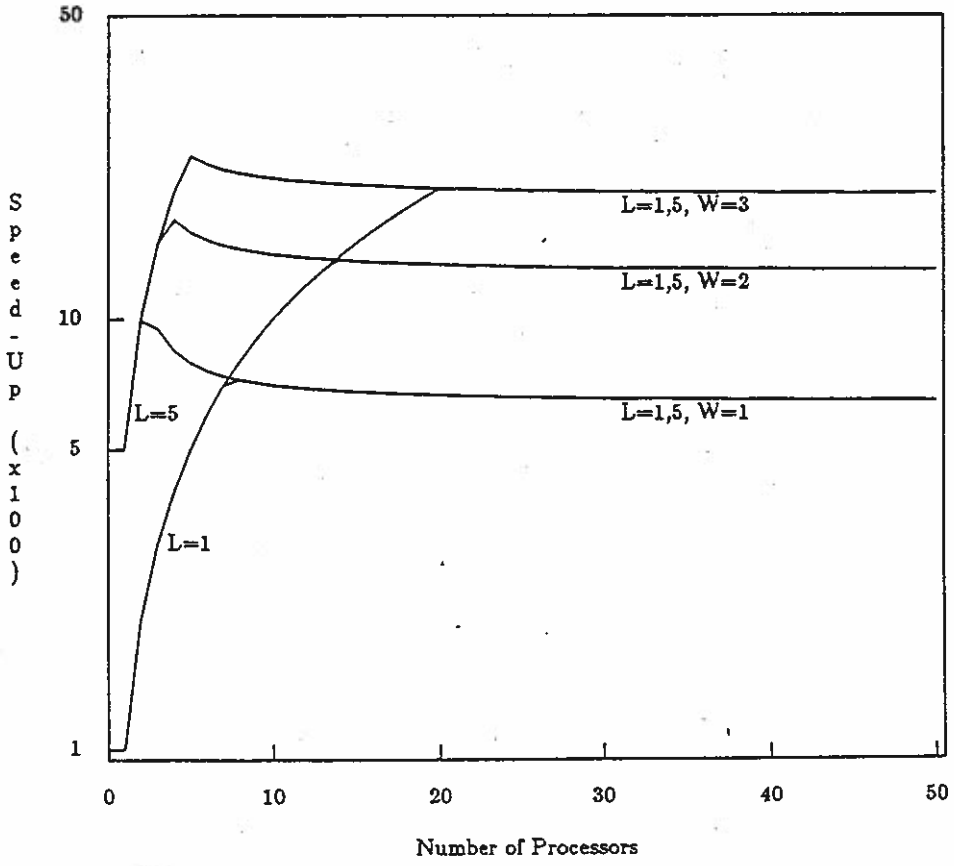

Figure 5. Speed-Up Versus Number of Processors (H=100,t_M=3 syncs)