Report Number: WUCS-88-35

1988-10-01

Performance Evaluation of Hierarchical Simulation Systems

Mark N. Edelman and Mark A. Franklin

Simulation needs for design analysis, verification, and testing have become increasingly important as integrated circuit size and complexity have grown. One technique for dealing with this problem is to utilize hierarchical modeling and simulation methods. This paper presents an analysis of hierarchical simulation systems in terms of two performance measures; the number of statements required for describing a system, and the simulation system execution time associated with a given hierarchical system representation. A model of hierarchical simulation system performance is developed. The performance of the hierarchical simulator, lsim2, is examined through its use on the set of benchmark circuits... Read complete abstract on page 2.

Performance Evaluation of Hierarchical Simulation Systems

Mark N. Edelman and Mark A. Franklin

Complete Abstract:

Simulation needs for design analysis, verification, and testing have become increasingly important as integrated circuit size and complexity have grown. One technique for dealing with this problem is to utilize hierarchical modeling and simulation methods. This paper presents an analysis of hierarchical simulation systems in terms of two performance measures; the number of statements required for describing a system, and the simulation system execution time associated with a given hierarchical system representation. A model of hierarchical simulation system performance is developed. The performance of the hierarchical simulator, lsim2, is examined through its use on the set of benchmark circuits and the results discussed in light of model predictions.

PERFORMANCE EVALUATION OF HIERARCHICAL SIMULATION SYSTEMS

Mark N. Edelman and Mark A. Franklin

WUCS-88-35

October 1988

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.

For Submission To
Ninth International Symposium
on
Computer Hardware Description Languages

PERFORMANCE EVALUATION OF HIERARCHICAL SIMULATION SYSTEMS *

M. Edelman and M. A. Franklin
Computer and Communications Research Center
Washington University
St. Louis, MO 63130

ABSTRACT

Simulation needs for design analysis, verification, and testing have become increasingly important as integrated circuit size and complexity have grown. One technique for dealing with this problem is to utilize hierarchical modeling and simulation methods. This paper presents an analysis of hierarchical simulation systems in terms of two performance measures; the number of statements required for describing a system, and the simulation execution time associated with a given hierarchical system representation. A model of hierarchical simulation system performance is developed. The performance of a hierarchical simulator, *lsim2*, is examined through its use on a set of benchmark circuits and the results discussed in light of model predictions.

Mail Address:

M. Franklin
Washington University
Campus Box 1115, Bryan 307
One Brookings Drive
St. Louis, MO. 63130-4899

Tel: (314) 889-6107

* This research was sponsored in part by funding from NSF Grant DCR-8417709.

Performance Evaluation of Hierarchical Simulation Systems *

M. Edelman and M.A. Franklin
Computer and Communications Research Center
Washington University
St. Louis, MO. 63130

1. Introduction

As integrated circuits have increased in size and complexity, simulation needs for design analysis, verification and testing have become increasingly important. Two problems have arisen in simulating such large circuits; representational complexity, and excessive simulation execution times.

The problem of representational complexity corresponds to the forest and the trees problem. On the one hand the designer must be concerned with the details of the trees, yet on the other hand he must maintain an understanding and view of the overall system. Hierarchical modeling and simulation techniques attempt to deal with this representational problem by providing the user with capabilities for obtaining various simultaneous views of the system in terms of interacting subsystems, each of which may be modeled at a different level of detail. By doing this the user may focus on the details of one subsystems while maintaining the correct interactions and interfaces with the remainder of the system (typically modeled at a higher level).

The problem of large simulation execution times has been attacked in a number of ways. One approach has been to design and produce special purpose logic simulation engines to substantially speed up simulation at the logic gate and switch levels [1,2,3,4]. Another approach has been to exploit the commercial availability of general purpose parallel processors for simulation purposes [5,6,7,8]. Finally, one can utilize hierarchical simulation techniques to reduce simulation execution times. In this approach, the speed comparison is between performing simulations entirely at a low level, versus simulating a part of the system at a low level, with the remainder of the system a higher levels. Often it is not possible to simulate an entire system at the lowest gate/switch levels, and one must simulate only a

* This research was sponsored in part by funding from NSF Grant DCR-8417709.

smaller subsystem. This can lead to various problems relating to developing the proper interface signals with those portions of the system which are not being simulated at the time. Hierarchical simulation techniques avoid such problems by allowing for simulation of the entire system. Since low level models generally take longer to execute then high level models, hierarchical simulation can provide for higher simulation speeds when compared with single low level models **.

In this paper, preliminary results are presented which indicate the sort of increase in simulation performance which accrues when using hierarchical simulation techniques. We begin by briefly describing the *lsim2* simulation system which permits the designer to represent circuits in terms of a three-level modeling hierarchy (gate/switch, MSI, RTL). A simple performance model is then presented with allows one to determine the sort of speedup one can achieve using hierarchical simulation techniques. Three benchmark circuits are then considered and the fraction of the system modeled at the different levels varied. Simulation execution time is measured and the results are presented along with an indication of representational complexity (measured by the number of language statements needed).

2. The *lsim2* Simulation Language

The *lsim2* [9] hierarchical simulation system was used in this experiment. *Lsim2* is an extension to *lsim* [10], a language which permits the modeling of circuits at the SSI logic gate-switch levels and was initially designed to aid in gathering data about the logic simulation process [11, 12]. *Lsim2* adds to *lsim* MSI level component modeling capabilities and a simple RTL language for modeling subsystems. The remainder of this section illustrates the basic language constructs available in *lsim2* by way of a simple system. The system takes the contents of two registers, Ra and Rb, adds them together and places the result in third register, Rc (Figure 1). The system could be represented at the RTL level as shown in Figure 2. The description is straight forward. Initially, a set of inputs (A, B, load), an output (C), registers (Ra, Rb, Rc), and a system clock (clkin) define the environment for the model. The body of the model is then defined (begin body). The load signal determines whether the registers are being loaded or

** We do not discuss here the use of simulation engines or general purpose parallel processors in conjunction with hierarchical simulation.

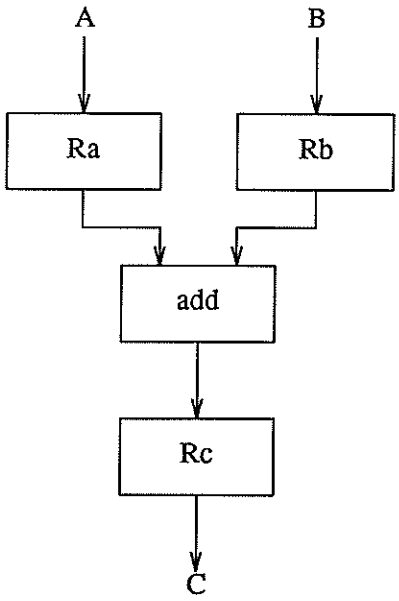

Figure 1. Simple System Description

```
begin circuit
    begin environment
       sysclk = clkin;
       inputs = (A,B,load);
       outputs = (C);
       registers = (Ra,Rb,Rc);
    end environment;
    begin body
       if(load == 1)
           then par
              Ra <- A;
              Rb <- B;
              C <- Rc;
           endpar;
           else
              Rc <- Ra + Rb;
       endif;
    end body;
end circuit;
```

Figure 2. Register Transfer Level Description of Simple System

whether the addition operation is performed. When load is high, the registers are loaded in parallel using the par-endpar construct. When the load signal goes low, addition takes place. These actions all occur synchronized with the positive-going edge of the clock signal. The inputs (including the clock) are easily defined using the interactive language available with Lsim. At this level verifying the design is akin to debugging a higher level language program.

Implementing this system at the next, more detailed level requires using various MSI level components. This is shown in Figure 3. The MSI level components correspond to the registers and adder shown in the figure. An SSI level component, not, is also present. The MSI components are defined in terms of their width, bit level inputs including a control input (load and load_bar), and bit level outputs. Note that for this simple case, the number of statements associated with the RTL and MSI levels of description are about the same, although the RTL description is clearly easier to understand.

In Figure 4, a gate-level implementation of the system is presented. Even with the extensive use of macros to simplify the description, the size of the description has multiplied and the nature of its operation has become less obvious. We could proceed further and present a switch level model of the simple system. This model would continue the trend demonstrated by the three levels shown.

Aside from representational simplicity, the difference in evaluation times between systems simulated at a single low level, versus a hierarchical model where only a small fraction of the system is simulated at the low level, is potentially quite large. There are two factors associated with this difference. First, a given RTL model will execute faster than the equivalent gate level model since the model is more compact and fewer instructions need to be executed to perform the simulation. Second, it is assumed that overall simulation times grow faster than linearly with the number of components being simulated. Therefore, the potential gains from hierarchical simulation will be more significant the larger the system

```
begin circuit
    begin environment
       sysclk = clkin;
       inputs = (A,B,load);
       outputs = (C);
    end environment;
    begin components
       invt = (not,inputs=(load),outputs=(load_bar));
       Ra = (register,width=8,inputs=(clkin,load,A[7-0]),outputs=(Ra_out[7-0]));
       Rb = (register,width=8,inputs=(clkin,load,B[7-0]),outputs=(Rb_out[7-0]));
       Rc = (register,width=8,inputs=(clkin,load_bar,sum[7-0]),outputs=(C[7-0]));
       Sum = (adder,width=8,inputs=(Ra_out[7-0],Rb_out[7-0]),
                    outputs=(sum[7-0]));
    end components;
end circuit;
```

Figure 3. MSI Module Level Description of Simple System

```
begin circuit
    begin environment
       sysclk = clkin;
       inputs = (A,B,load);
       outputs = (C);
    end environment;
    begin macros
       reg_mac = (
           inputs=(clk,ld,in[7-0]),
           outputs=(out[7-0]),
           components=(
              R0=(dff,inputs=(clk,ld,in[0]),outputs=(out[0])),
              R1=(dff,inputs=(clk,ld,in[1]),outputs=(out[1])),
              R2=(dff,inputs=(clk,ld,in[2]),outputs=(out[2])),
              R3=(dff,inputs=(clk,ld,in[3]),outputs=(out[3])),
              R4=(dff,inputs=(clk,ld,in[4]),outputs=(out[4])),
              R5=(dff,inputs=(clk,ld,in[5]),outputs=(out[5])),
              R6=(dff,inputs=(clk,ld,in[6]),outputs=(out[6])),
              R7=(dff,inputs=(clk,ld,in[7]),outputs=(out[7])))));
       add_mac = (
           inputs=(x[7-0],y[7-0]),
           outputs=(z[7-0]),
           components=(
              A0=(add_cell,inputs=(x[0],y[0],gnd),outputs=(z[0],c0)),
              A1=(add_cell,inputs=(x[1],y[1],c0),outputs=(z[1],c1)),
              A2=(add_cell,inputs=(x[2],y[2],c1),outputs=(z[2],c2)),
              A3=(add_cell,inputs=(x[3],y[3],c2),outputs=(z[3],c3)),
              A4=(add_cell,inputs=(x[4],y[4],c3),outputs=(z[4],c4)),
              A5=(add_cell,inputs=(x[5],y[5],c4),outputs=(z[5],c5)),
              A6=(add_cell,inputs=(x[6],y[6],c5),outputs=(z[6],c6)),
              A7=(add_cell,inputs=(x[7],y[7],c6),outputs=(z[7],c7))));
       add_cell = (
           inputs=(x,y,cin),
           outputs=(sum,cout),
           components=(
              x1=(xor,inputs=(x,y),outputs=(t1)),
              x2=(xor,inputs=(t1,cin),outputs=(sum)),
              a1=(and,inputs=(x,y),outputs=(t2)),
              a2=(and,inputs=(x,cin),outputs=(t3)),
              a3=(and,inputs=(y,cin),outputs=(t4)),
              o1=(or3,inputs=(t2,t3,t4),outputs=(cout))));
    end macros;
    begin components
       invt = (not,inputs=(load),outputs=(load_bar));
       Ra = (reg_mac,inputs=(clkin,load,A[7-0]),outputs=(Ra_out[7-0]));
       Rb = (reg_mac,inputs=(clkin,load,B[7-0]),outputs=(Rb_out[7-0]));
       Rc = (reg_mac,inputs=(clkin,load_bar,sum[7-0]),outputs=(C[7-0]));
       Sum = (add_mac,inputs=(Ra_out[7-0],Rb_out[7-0]),outputs=(sum[7-0]));
    end components;
end circuit;
```

Figure 4. Gate Level Model of Simple System

being simulated.

Note that while the movement toward higher levels of description abstraction has advantages, it also reduces the level of information available for analysis. At the higher levels, issues such as timing and

fault analysis are difficult if not impossible to evaluate.

3. A Simple Performance Model

In this section a simple performance model is presented. The model focuses on the gains which accrue when using hierarchical simulation techniques in large simulations. The main parameters of the model are given in Table 1.

Parameter	Typical Values	Meaning
X		The total number of components in the circuit, if the entire circuit were represented at the lowest level.
p_i	0 - 1.0	The fraction of X represented at level i, $i \geq 1$, $0.0 \leq p_i \leq 1.0$, $\sum p_i = 1$.
X_{ci}		The number of lowest level components represented at hierarchical level i, $X_{ci} = p_i X$.
g_i	.1 - 1.0	The representation compaction factor at level i relative to the number of components at level 1, $g_1 = 1$.
X_{si}		The number of statements at hierarchical level i needed to represent the fraction of the circuit modeled at that level, $X_{si} = g_i X_{ci}$.
k	1.2 - 3.0	The execution time factor. Execution time is proportional to the number of statements raised to the kth power.

Table 1: Model Parameters

If X is the total number of components needed to model the circuit if it were represented entirely at the lowest level, then $X_{ci} = p_i X$, is the number of *lowest level components* represented at level i.

For $i > 1$, fewer then X_{ci} components (statements) will be needed because of the higher level of abstraction used. The compaction factor g_i determines how many level 1 components can be represented with a single statement at level i ($i > 1, g_1 = 1$). Since there are $p_i X$ level 1 components represented at level i, the number of statements needed for representation at level i is $X_{si} = g_i X_{ci} = g_i p_i X$.

In logic simulation, the time required to perform a simulation is proportional to the total number of events required for the simulation. This relation is shown in equation 1.

$$T = \gamma E \tag{1}$$

Here, T is the simulation run time (measured in CPU seconds on a SUN-3/280), E is number of events, and γ is a proportionality constant that represents the time required for a single event to be evaluated.

We would prefer to relate the simulation run time to a more easily identified parameter of the circuit than number of events (for instance, number of statements for correspondance with our observed relation). If we assume that the number of events can be related to the number of components (or statements) as shown in equation 2, we can then substitute equation 2 into equation 1 and arrive at a simulation run time dependant upon circuit parameters.

$$E = \alpha n N X^k \tag{2}$$

In equation 2, n is sum of average fan-in and fan-out for the components in a circuit, X is number of components in the circuit, N is the total number of user input events (setting of signals for example), and α is a proportionality constant. This equation can be interpreted as representing the multiplication of input events caused by the fan-out of signals and component outputs. The constant α then represents the average probabilty that a component input change generates a change in state and a corresponding propagation of the state change to the output(s) as well as an average intensity of activity for the circuit.

Substituting equation 2 into equation 1, combining constants and generalizing to a three level hierarchy of components (i.e., gate/switch = level 1, MSI = level 2, RTL = level 3), we arrive at equation 3.

$$T_{total} = \beta_1 n_1 N X_1^{k_1} + \beta_2 n_2 N X_2^{k_2} + \beta_3 n_3 N X_3^{k_3} \tag{3}$$

where $\beta_i = \gamma_i \alpha_i$, γ_i is the event proportionality constant for circuit abstraction level i, and α_i is as described above for circuit abstraction level i.

The simulation run time speed up can then be defined as

$$su = \frac{\beta_1 n_1 N X^{k_1}}{T_{total}} \tag{4}$$

where X is the number of components of level 1 type that are required to represent the circuit entirely at level 1. Reducing equation 4 we arrive at

$$su = \left[\frac{X_1^{k_1}}{X^{k_1}} + \delta_2 \frac{X_2^{k_2}}{X^{k_1}} + \delta_3 \frac{X_3^{k_3}}{X^{k_1}} \right]^{-1} \tag{5}$$

where $\delta_i = \frac{\beta_i n_i}{\beta_1 n_1}$. Using the relation $X_i = g_i p_i X$ Equation 5 can be reduced to

$$su = \left[(p_1 g_1)^{k_1} + \delta_2 (p_2 g_2)^{k_2} X^{k_2 - k_1} + \delta_3 (p_3 g_3)^{k_3} X^{k_3 - k_1} \right]^{-1} \tag{6}$$

Typical speedup curves are shown in Figure 5. The g parameters correspond to typical values as measured from the benchmark circuits discussed in the next section. The δ parameters were determined experimentaly from simulation results. As expected, the curves show that as a greater fraction of the circuit is modeled at higher levels of abstraction the speedup that can be achieved over a single (lowest) level modeling increases. Thus, as p_3 increases (for a fixed p_2), S increases, and also as p_2 increases (for a fixed p_3), S increases. Note that if only the highest and the lowest levels are used, and the modeling strategy is to have only five percent of the circuit modeled at the lowest level at any time, then speedups better than 20 can be expected.

4. The Experiment

To explore the effects of hierarchical description on simulator performance, several experiments were run. These experiments consisted of describing three benchmark circuits at several levels of abstraction (gate/switch, mixed gate/switch and MSI, MSI, mixed MSI and RTL, RTL) and then executing the circuits following the same simulation sequence. Data was gathered on the number of statements required to describe the circuit and the execution time of the simulation.

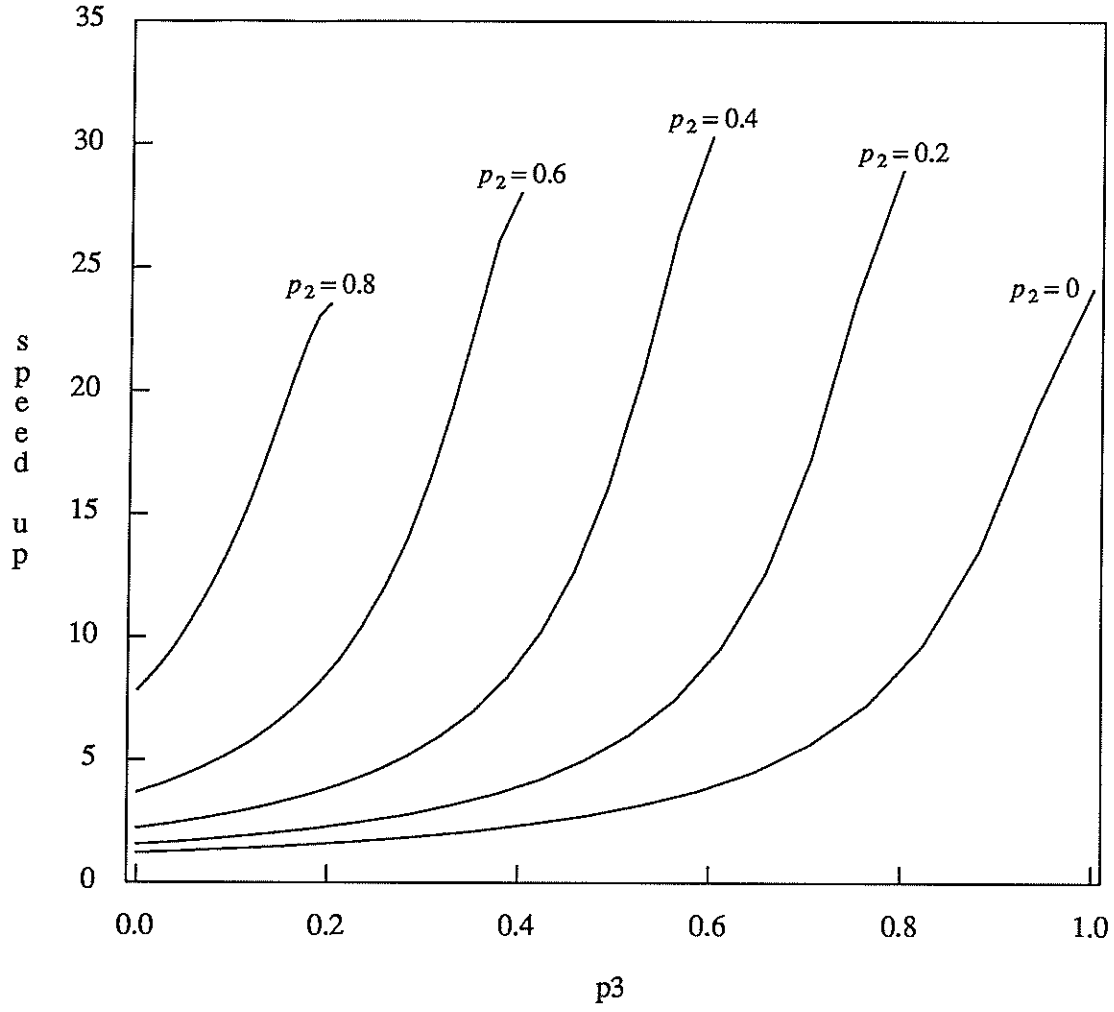

Figure 5: Speedup of Hierarchical over Single Level Simulation Versus Percent of Components at Level 3 (k = 1.5, g_3 = .08, g_2 = .02)

4.1. Benchmark Circuits

Our experiments utilized three integrated circuit designs (described below) that were done as student design projects at Washington University.

The first circuit models a simple digital stopwatch (stopw). The stopwatch consists of several cascaded BCD counters that keep track of the time (by counting clock cycles) between successive activations of a start/stop input. The circuit is highly modular and extensive use of the *lsim2* macro facilities was made in generating the gate/switch level description.

The second circuit implements a priority queue (pqueue) which is designed to maintain a partially ordered list of records. It does this by using a compare and swap algorithm on values associated with successive pairs of records in the queue. The first pair consists of the received record and the prior top of the queue. The circuit modeled is a single cell pair in the queue. It consists of a "head" and a "tail" register with a processor interposed between them to do the comparison and swap operations. Each of the registers is 48 bits wide (each of the registers is intended to hold an event record for use in implementing an event list for a hardware based simulation engine).

The third circuit is a radiation treatment planning chip (RTP). It was designed to implement an algorithm for determining the effective dose levels for patient radiation treatments. The RTP chip is divided into two portions; a control portion and a dataflow portion. The control portion consists of a large PLA, several registers and a few counters. It implements a 30-state finite state machine. The data flow portion of the RTP chip consists of two arithmetic/logic units, several registers, and a few multiplexers.

Table 2 summarizes the characteristics of the benchmark circuits and the results of the simulation experiments. The table shows the number of statements (after macro expansion) for each of the circuits. For the SSI case, the number of statements corresponds to the number of components (gates/switches) needed to model the circuit at this lowest level. Note that, in general, we can expect that the number of

circuit	ssi	ssi-msi	msi	msi-rtl	rtl
stopw					
#stmts	347	217	17	48	58
time	13.8	10.2	4.4	5.5	5.9
pqueue					
#stmts	1864	838	186	186	108
time	125.6	67.8	23.7	25.8	16.4
rtp					
#stmts	3168	3020	2749	238	142
time	240.3	232.7	82.3	24.8	13.2
average					
#stmts	1793	1286	984	157	102
time	126.6	103.6	36.8	18.7	11.8

Table 2: Benchmark Data and Simulation Results

statements decreases as we increase the percentage of the circuit which is modeled at higher levels of abstraction. In the case of the stop watch, however, this does not hold because the stop watch was designed with a number of MSI level components in mind (e.g., BCD counters). The result is that its description maps very well to the MSI representation level, and thus more statements are needed, in this case, for RTL representation than for MSI level representation.

4.2. Data Collection

Each of the circuits was run through a random sequence of inputs and allowed to reach an equilibrium state. The input was delivered through the use of the programming interface provided by *lsim2*.

For the stopwatch chip, the test consisted of randomly setting the state of the stopwatch, providing a random set of control inputs, and allowing the simulation time to advance one clock cycle. The state of the stopwatch is then compared against a software model of the stopwatch. The stopwatch was run through a total of sixteen randomly generated states.

The basic operation of the priority queue consists of pushing a data value onto the queue, comparing the new head of the queue with the old head and swapping the two if they are out of order. To test the priority queue, a sequence of push-compare-pop operations was done and the results compared against a simple software model of the circuit. In our experiments, eight randomly generated values were used as the data values to be pushed onto the queue in this sequence of operations.

The RTP chip was designed to be a part in a system consisting of the RTP chip and several memory blocks that store parameters for use in the calculations. Since models for the companion circuits were not available, the experiment consisted of stepping the state machine within the circuit through each of its states. In this manner, all of the control signals and a large portion of the data paths were exercised. Over thirty execution cycles of the RTP circuit were evaluated.

A simple count was made to obtain the number of statements in the circuit models at each description level. The run-time for the simulations was obtained using the UNIX time utility. The

resulting data is found in Table 2.

4.3. Results

From Table 2, we see a general correlation between the number of statements in the description and the run-time of the simulation. The results show that as the number of statements in the circuit description decreases, the time required to run the same simulation also decreases. To explore this trend further, the priority queue circuit was described at two additional levels of abstraction. These two additional levels fall between the gate/switch level and the MSI module levels, and between the MSI module level and the RTL levels. Utilizing these additional data points, the curves presented in Figure 6 were generated. In the curves, the parameter used as an independant variable is the level of abstraction of the circuit description. The parameter is calculated using the formula given below:

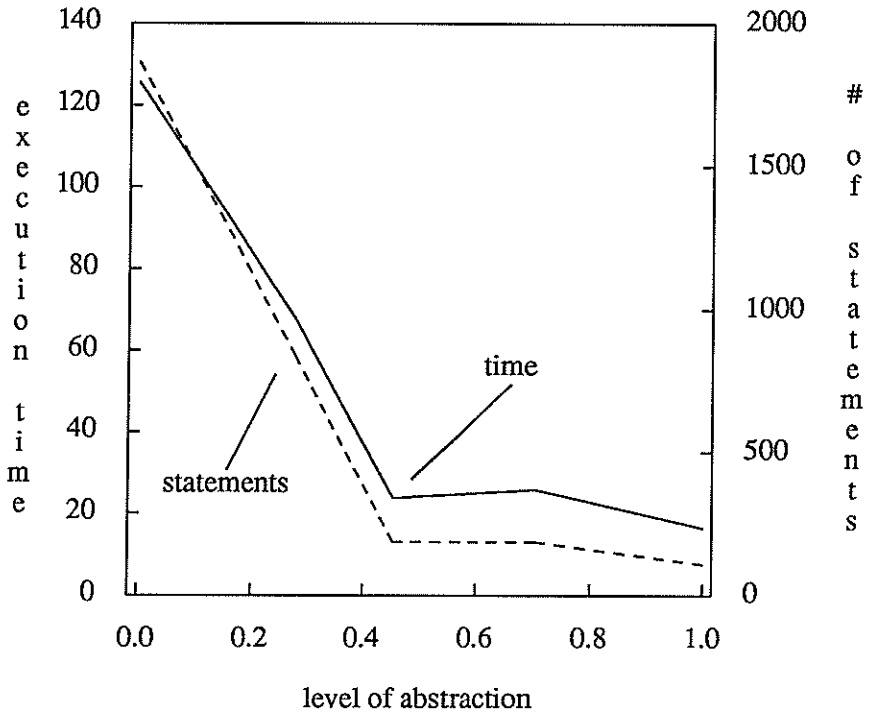

Figure 6. Priority Queue Description Statements and Execution Time

$$level\ of\ abstraction = 0.5 * (\%SSI\ replaced\ by\ MSI) + (\%SSI\ replaced\ by\ RTL)$$

This measure generates values between zero and one. A "zero" indicates that the representation is entirely at the SSI or gate/switch level. A "one" indicates that the representation is entirely at the RTL level. If an MSI description replaced all of the SSI components in a circuit, the level of abstraction would be 0.5. Similarly, if the MSI portion replaced 50 percent of the SSI description and the RTL portion replaced 35 percent, the level of abstraction would be 0.60.

To investigate the statement/execution time relation still further, the results of the three circuits were averaged to provide a composite picture of the entire set of circuits. In Figure 7, the curves representing the averaged results for the three circuits are presented. The results are similar in nature to those obtained for the priority queue.

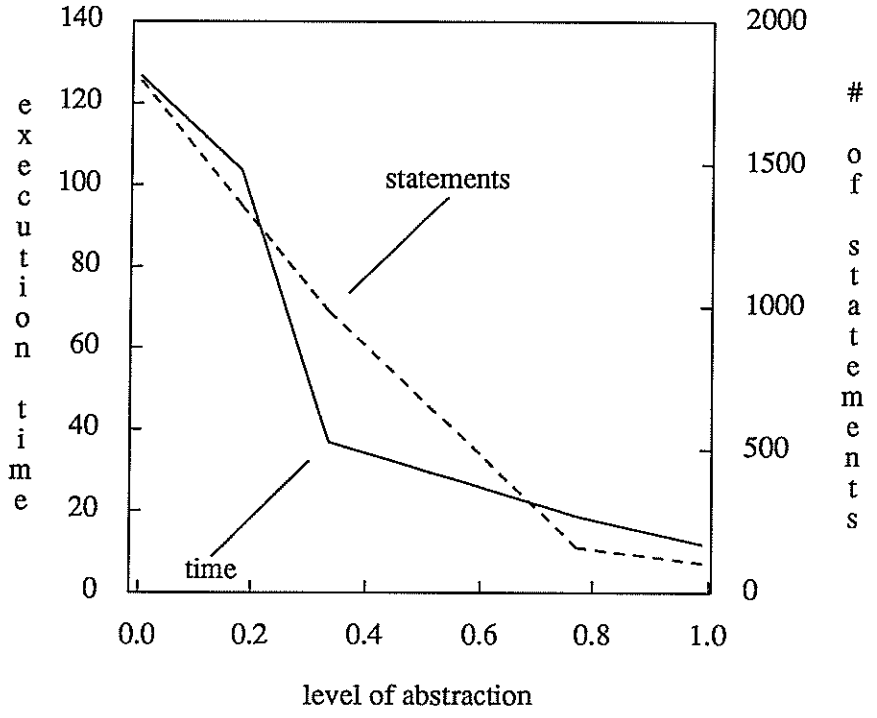

Figure 7. Averaged Results for Three Benchmark Circuits.

Within each of the circuits, the results obtained follow the general statement/execution time relation to varying degrees. The priority queue most closely matches our predictions of performance. The stop watch, however, shows some performance degradation as we move from the MSI module level to the RTL level for the reasons given earlier (i.e., the stop watch was designed with MSI level components of the sort available in *lsim2* in mind).

Finally, comparing the experimental results with those predicted by the performance model presented earlier, we find that while general trends are preserved, the speedups obtained are lower (e.g., the model predicts a speedup of approxiamtely 25 for an all RTL circuit, while we actually obtained 7.65 for the all RTL priority queue and 18.2 for the all RTL RTP circuit). The full reason for this discrepancy is not yet fully understood. One aspect reflects the fact that in the real simulation there is a constant overhead time which relates to setup and input/output time. This overhead does not change significantly with the number of components at each modeling level and is not included in the performance model. For circuits of the size considered in this experiment this overhead could be fairly considerable and would reduce the speedups achievable to below those predicted by the model. More fundamentally, however, is the fact that we do not yet have enough data to determine the appropriate value associated with the model parameters k or δ_i. These parameters can have significant effect on predicted performance.

5. Conclusions

In this paper we have presented the *lsim2* hierarchical modeling system and have investigated the performance of general hierarchical simulation systems. A simple performance model has been derived which relates performance to the fraction of the system modeled at each level. Performance curves were produced which show how the speedup (over a single level of modeling at the lowest level) increases as the fraction of the system modeled at the higher levels in the hierarchy increases. Experimental results were presented which verify the general trends shown in the performance model, although at somewhat lower values.

These results allow us to envision a simulation environment where the majority of the circuit design is described at the higher levels and only small portions at the low levels. In this manner, the designer can investigate small portions of his design at the lowest level of detail, while maintaining the higher performance of the higher level modeling language.

REFERENCES

1. Zycad Corp. , *The ZYCAD Logic Evaluator: Product Description,* Zycad Corp., N. Roseville, MN. (1983).
2. M. M. Denneau, "The Yorktown Simulation Engine: Architecture and Hardware Description," *ACM IEEE 19th Design Automation Conference,* pp. 55-59 (June 1982).
3. T. Blank , "Survey of Hardware Accelerators Used in Computer-Aided Design," *IEEE Design and Test,* pp. 21-39 (Aug. 1984).
4. M. A. Franklin , D. F. Wann, and K. F. Wong , "Parallel Machines and Algorithms for Discrete-Event Simulation," *Proc. 1984 Inter. Conf. on Parallel Processing,* pp. 449-458 (Aug. 1984).
5. R. Chamberlain and M.A. Franklin, "Discrete-Event Simulation on Hypercube Architectures," *Proc. Inter. Conf. on Comput. Aided Design (ICCAD'88),* (Nov. 1988).
6. NCUBE Computer Systems, *The NCUBE Parallel Processor*
7. Bolt Beranek and Newman Inc., "Butterfly 1000," *BBN Product Announcement,* (1987).
8. Intel Scientific Computers, "iPSC: The first family of concurrent supercomputers," *INTEL Product Announcement,* (1985).
9. M. Edelman, "Design and Performance of a Hierarchical Logic Simulation System ," *M.S. Thesis, Dept. of Elec. Engr., Washington Univ.,* (May 1989).
10. R.D. Chamberlain, "LSIM: A Gate-Switch Level Logic Simulator," *M.S. Thesis, Dept. of Comput. Sci., Washington Univ.,* (May 1985).
11. R. D. Chamberlain and M. A. Franklin , "Collecting Data About Logic Simulation," *IEEE Trans. on Computer-Aided Design* **CAD-5** (3) pp. 405-412 (July 1986).
12. K. F. Wong , M. A. Franklin , R. D. Chamberlain , and B. L. Shing , "Statistics on Logic Simulation," *Proc. 23rd Design Automation Conf.,* (June 1986).