Performance Evaluation of an Event-Driven Logic Simulation Machine

Fumiyasu Hirose
Fujitsu Laboratories LTD.
1015 Kamikodanaka, Nakahara-ku, Kawasaki 211, Japan

ABSTRACT
This paper evaluates the performance of an event-driven logic simulation machine, the SP. Since an event-driven machine only schedules gates that have signal-changes on their inputs, it processes fewer gates than the level-sort machine does. However, if the event-driven machine spends too many clocks on dynamic scheduling, the simulation time cannot be reduced. We measured the overhead for dynamic scheduling, and found that it only averages 2% over the total process. The evaluation was done using the ISCAS'89 benchmark circuits, and the results are shown on a machine cycle basis. We individually evaluated some special functions of the SP for acceleration. Furthermore, we compared the simulation speed with that of a software simulator that uses the same data structure and algorithm as the SP.

1 Introduction
Logic simulation is essential for the quick development of faultless digital systems. Logic simulation machines[1] have made great progress in accelerating this process. The origin of the machines is said to be the Boeing Computer Simulator in 1970[2]. However, practical developing machines have become popular since the 1982 YSE[3]. The logic simulation machines are one of the most successful CAD engines in both technical and economic effects.

The logic simulator evaluates the output values of gates in a circuit. In the YSE, the gates are evaluated according to a static pre-defined order, where each gate is evaluated exactly once during each simulated clock cycle. This makes the hardware simple, but involves useless evaluation. An event-driven machine employs dynamic scheduling. It only schedules gates that have signal-changes (events) on their inputs. Far fewer gates need to be evaluated if the circuit activity is low, and the average event rate is estimated to be 15%[1]. This means that event-driven machines could reduce the number of evaluated gates by five-sixths. However, if the machine spends a lot of clocks on dynamic scheduling, this advantage cannot reduce the simulation time.

In the event-driven machine architecture of the SP[4], the gates that should be evaluated during the next simulation cycle can be scheduled in parallel to the main process of the evaluation and the update of gates for the present cycle. We have measured the overhead of dynamic scheduling, and found that it only averages 2% over the total process.

A performance evaluation of the level-sort machine has been reported in detail[5]. However, there is little information on event-driven machines. Furthermore, conventional reports have not used circuits publicly available, and the simulation speed is often compared with a software simulator that does not necessarily use the same data structures, models, or algorithms. We do not consider these evaluations sufficient to evaluate a machine architecture.

This paper evaluates the performance of the SP using the ISCAS'89[6] benchmark circuits, and reports it on a machine cycle basis. Special functions of the SP are evaluated one by one. Furthermore, the SP's speed is compared to a software simulator that uses identical data structures and algorithms.

2 Characteristics of benchmark circuits
2.1 Static characteristics
The five largest ISCAS'89 benchmark circuits were selected. These were originally synchronous circuits. However, we removed registers and made them combinational so that we could easily control the circuit event rate. The static characteristics of the circuits we obtained are shown in Table 1.

"#PO" is the number of primary output buffers, "#PI" is the number of primary input buffers, and "#Gates" is the number of internal gates. The internal gates do not contain the PIs and the PIs. "Ave. #FO" is the average number of fanout gates of the PIs and internal gates. "Ave. #FI" is the average number of fanin gates of the PIs and internal gates.

2.2 Simulation conditions
We generated input patterns for the circuits with a random pattern generator that generates input sequences at a given rate. Each input is independent of the other inputs. A single processor of the SP is used for the simulation. The unit delay model is used. When the circuit becomes stable for the current input pattern, the next
Table 1. Static characteristics

<table>
<thead>
<tr>
<th>Name</th>
<th>N*Gates</th>
<th>P</th>
<th>S</th>
<th>N*POs</th>
<th>N*POs</th>
<th>N*O</th>
<th>Avg N*O</th>
<th>Avg N*O</th>
</tr>
</thead>
<tbody>
<tr>
<td>s38584</td>
<td>22969</td>
<td>1264</td>
<td>1792</td>
<td>7303</td>
<td>1.60</td>
<td>1.57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s38417</td>
<td>21014</td>
<td>1564</td>
<td>1742</td>
<td>3692</td>
<td>1.39</td>
<td>1.38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s35932</td>
<td>17793</td>
<td>1763</td>
<td>2048</td>
<td>32045</td>
<td>1.64</td>
<td>1.62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s15850</td>
<td>10649</td>
<td>611</td>
<td>684</td>
<td>15206</td>
<td>1.35</td>
<td>1.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s13207</td>
<td>9116</td>
<td>700</td>
<td>790</td>
<td>13120</td>
<td>1.34</td>
<td>1.32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>16746</td>
<td>1240</td>
<td>1399</td>
<td>26673</td>
<td>1.48</td>
<td>1.47</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

one is applied. The period of the simulation for one input pattern is called a cycle, while the word clock refers to the machine clock cycles of the SP.

2.3 Dynamic characteristics

Gates that had events in their inputs are called active gates. The event-driven machine only evaluates the active gates. The ratio of active gates to all gates is the gate event rate. The ratio active gates whose output will change to all active gates is the event through rate.

Figure 1 shows the relation between the gate event rate and the event rate at the primary inputs (P1s). The relation is almost linear for all circuits. The gate event rate can exceed 100% because some gates are evaluated more than once based on the existence of circuit hazards. Figure 2 shows the relation between the event through rate and the event rate at the P1s. The event through rate is almost constant. On average, 82% of the active gates generate events on their output.

3 Basic performance evaluation

3.1 Ideal clocks

All logic simulator have data structures that hold the status of the circuit signals. This is called net status memory (NSM). The simulator computes the NSM from unit time to unit time. The NSM is read for gate evaluations, and written for status updates.

In the level-sort simulation, every gate is evaluated and updated exactly once during each simulation cycle. Let the number of gates G. Then, the NSM is accessed 2G times. On the other hand, the event-driven simulator only evaluates active gates, updates gates that will have events on their outputs, and schedules active gates for the next unit time. This scheduling is called fanout-fetch. Let the gate event rate be E, and the event through rate be R. Then, the NSM is read EG times for evaluation, and is written REG times for update. Therefore, the NSM is accessed E*(1+R)*G times during each simulation cycle. Remember that the average event through rate is 0.82. When we use a gate event rate of 15%, the event-driven NSM is accessed 0.15*1.82*G=0.273G times, which is about one-seventh as often as the level-sort simulator.

In the SP architecture, we partitioned the simulation process into three functions, Evaluate, Update, and Fanout-fetch, as shown in Figure 3. Each function is processed by a pipeline. Furthermore, we intended to mask the Fanout-fetch process by running it in parallel to Evaluate and Update. Evaluate and Update cannot run in parallel because they both access the NSM. Each stage of the pipeline can access its local memory once per clock in the SP. Therefore, the pipelines ideally process one gate per clock. If the Fanout-fetch is also completely masked, the machine can process a simulation cycle in E*(1+R)*G clocks, which is the number of accesses to the NSM. We define this as the number of ideal clocks for a simulation cycle. Also, (1+R) clocks is called the number of ideal clocks for an active gate.

3.2 Relation between ideal and measured clocks

We calculated the number of ideal clocks and measured the actual machine clocks of the SP for a simulation cycle. Figure 4 shows the relation. This rough observation shows that the SP operates in
We measured the number of clocks, "Y", spent to process an active gate. Figure 5 shows the relation between Y and the number of ideal clocks for a simulation cycle, "X=E*(I+R)*G". Ideally, the relation should be Y=(I+R), that is almost constant. However, it is observed that if X is small, in other words, if the number of gates to be processed by the pipelines is small, the SP needs more machine clocks.

The SP has overhead for pipelining and dynamic scheduling. The pipeline overhead is in inverse proportion to X. Figure 5 shows this relation. The pipeline overhead also exists in a level-sort machine.

### 3.3 Dynamic scheduling overhead

Figure 6 shows the number of gates evaluated per unit time in a simulation cycle for the two largest circuits, s38584 and s38417. The event rate at the primary inputs is 10%. About 80% of the active gates for a cycle are evaluated by unit time 10.

If the number of gates scheduled in Fanout-fetch(t+1) in Figure 3 is greater than the total number of gates in Evaluate(t) and Update(t) at unit time t, then there is some dynamic scheduling overhead. We found two points where overhead exists in the case of s38584. These are denoted in Figure 6 as A at unit time 1 and B at unit time 8. No overhead was observed in s38417. Receiving external events as FI events at unit time 1 is considered to be a part of the Evaluate pipeline process.

We added the number of overhead clocks for each time unit for a cycle. The total number is divided by the number of ideal clocks of a simulation cycle to obtain the dynamic scheduling overhead. These figures are shown in Figure 7. The overhead is almost independent of the event rate at the primary inputs. The maximum overhead is 5%, and the average is only 2%. Except for s38584, the only overhead observed was at unit time 1. s38584 only had overhead at unit times 1 and 8, and the overhead at unit time 1 was much heavier. This means that almost all of the overhead is caused when the external events are received at unit time 1. There is little overhead for the simulation of internal circuits. The overhead of dynamic scheduling in the SP is thus very small, so the number of machine clocks spent to process individual gates is almost the same as that of a level-sort machine, while the SP only selects the active gates.

### 4 Effect of special functions

We evaluated the effects of two SP functions. One is a registration filter for avoiding double registration of active gates. The other is input inverter that inverts input values without using
inverting gates. The registration filter is a built-in SP function. Its effect is included in the basic performance evaluation in Chapter 3. On the other hand, the input inverter is used by the circuit compiler. Its effect is not included in the basic performance.

We measured the number of gates that are scheduled more than once per unit time. The ratio of this number to the total number of active gates is shown in Figure 8 using black circles. This is the average ratio of the five circuits. When the event rate at the primary inputs increases, the double registration ratio per unit time also increases. The increase in number is trivial, but the ratio also increases. When the event rate at the primary inputs is 10%, about 2% of the active gates are scheduled more than once, if the SP does not have the registration filter.

If the registration filter is not used, extra gates are caused and processed in the following ways. First, if a gate is scheduled more than once, it must be evaluated more than once with identical input vectors. Second, if an event is detected at the output of the gate, it must be updated more than once by the same value. Third, the same fanout-fetch process must be done more than once. Then, the rescheduled gates will generate active gates that are registered more than once for the next cycle. The ratio of the extra gates that would be registered to that of the active gates actually registered in the SP is shown in Figure 8 using white circles. This is also the average ratio of the five circuits. When the primary input event rate is 10%, about 22% of the active gates are scheduled more than once per simulation cycle.

Some machines can invert input vectors without using invertors. It is especially important when ECL circuits are simulated. By using the input inverter, we can eliminate invertors from a circuit. The simulation may be accelerated, if few hazards are generated. Table 2 shows this effect. The size effect column shows the ratio of the original circuits to the new circuits. On average, the circuits are reduced by half. The speed effect column shows the acceleration of the input inverter. The event ratio at primary inputs is 10%. On average, the speed is accelerated 1.7 times.

5 Comparison to software simulator
We compared the SP's simulation speed with a software simulator that uses the same data structures and algorithms. When the event rate at the primary inputs is 10%, a single SP processor simulates 170 times faster than the software simulator does on average. The maximum speed-up is 181 times for s38417, and the minimum is 149 times for s13207. The software simulator runs on a 25 MHz SUN4/370 with 32 megabytes of memory.

6 Conclusion
We evaluated the performance of the SP event-driven simulation machine. The dynamic scheduling overhead averaged 2%. We used the ISCAS'89 benchmark circuits, and reported the performance on a machine cycle basis for objectivity. We also evaluated special functions of the SP, and compared the speed to a software simulator that uses identical data structures and algorithms.

SPs are being used as a main tool of logic verification for large digital system designs. We would like to extend this research to establish a basis of comparing the performance of other machine architectures, and discuss the tradeoffs of simulation algorithms.

References