

High-Level VHDL Simulator Running on Logic Simulation Machines

Minoru Shoji, Fumiyasu Hirose
FUJITSU LABORATORIES LTD.
1015, KAMIKODANAKA NAKAHARA-KU
KAWASAKI 211, JAPAN

Abstract

VHDL High-level description is useful to describe the behavior of circuits with abstractions at their early development stages. Software VHDL simulators are, however, not adequately fast for large circuits.

Logic synthesis can be applied to obtain gate level description from VHDL description so that gate-level logic simulation machines can be used. However, since only fairly low level conventional logic synthesis can be treated, high-level descriptions have to be written down to this level.

We developed a method to obtain gate-level descriptions from high-level behavioral description. Each gate of the description represents a control and data flow node of a high-level behavioral description. We estimated the simulation speed of high-level descriptions employing simulation machines, and found that it is accelerated 400 times over software simulators.

1. Introduction

In the early stages of circuit development, the VHDL high-level behavioral description is useful for brief descriptions and circuit simulation[1][2]. Software simulators are not fast enough, however, to simulate the high-level descriptions of large circuits. This is because only one processor controls the simulator, in addition to evaluating the logic values.

To accelerate the simulation speed of gate-level descriptions, gate-level logic simulation machines can be used. Logic synthesis can be applied to obtain gate-level descriptions from VHDL descriptions. However, since only relatively low-level conventional logic synthesis can be treated, circuits must be designed in detail to obtain gate-level descriptions.

We developed a method to obtain gate-level descriptions from high-level behavioral descriptions. Each gate in this description represents control and data flow nodes of high-level behavioral description. With this method, we can employ gate-level logic simulation machines from the early development stages. This method works on the VHDL subset that is enough to describe high-level behavioral descriptions for logic verification of synchronous circuits.

We estimated the simulation speed of high-level behavioral descriptions using a simulation machine SP[3] by our method, and found that the simulation speed is accelerated 400 times over a software simulator.

2. The VHDL Subset

The following three kinds of statements are excluded from our high-level circuit descriptions.

- 1) Dynamic memory allocation

This statement does not have to be supported because it cannot be directly realized with a circuit. Recursive function calls are also excluded for the same reason.

- 2) Time expressions

Because the real delay values are not determined until circuit placement and routing is done, it does not make sense to include these values in high-level behavioral descriptions. Moreover, if circuits are designed synchronously, and we only want to check the logic, delay information is not needed.

- 3) Text messages

Text messages are provided for display if an error is detected during simulation. Because simulation machines have no display terminals, this statement can be excluded. Other methods can be substituted for.

With these premises, high-level behavioral descriptions can be translated into a kind of gate-level description so that a logic simulation machine can be used.

3. Simulation Algorithms

3.1 Event-Driven Simulation Algorithm

The event-driven algorithm is implemented on the simulation machine we use. If the primitive's value changes, the simulator finds connected primitives and sends them an event with new value. The evaluation function of a primitive is any logical equation.

We used a simulation machine that has two kinds of events.

- 1) Update

If a primitive receives this type of event, it updates the input value.

- 2) Evaluate

If a primitive receives this type of event, it evaluates the value.

Figure 1 shows an example of event-driven simulation. The line types show the kind of event sent to the connected primitive. Solid lines show that both update and evaluate events are sent if the value changes. Dotted lines show that update events are sent. Dot-dashed lines show that evaluate events are sent. Figure 1(a) shows the first phase when the logic value of I1 changes from 0 to 1. Figures 1(b) and 1(c) show the new events for the second and third phases. Because no evaluate event is sent to any primitives in Fig.1(c), no primitive evaluates the value in the next phase.

3.2 VHDL Simulation Algorithm for Simulation Machines

VHDL simulation is basically event-driven simulation. When we ignore physical delay values as described in section 2, the delay types of VHDL simulation may be defined by UNIT, DELTA, and CLK delays as described below.

1) UNIT

We introduced this delay type as the delay for value assignment on VHDL variables and evaluation of each primitive.

2) DELTA

This delay type is the same as the one defined by VHDL.

3) CLK

In this paper, we simulate a synchronous circuit. We introduced this delay type to simulate clock cycles. Process statements which are synchronous with the clock signal receive events of this delay type.

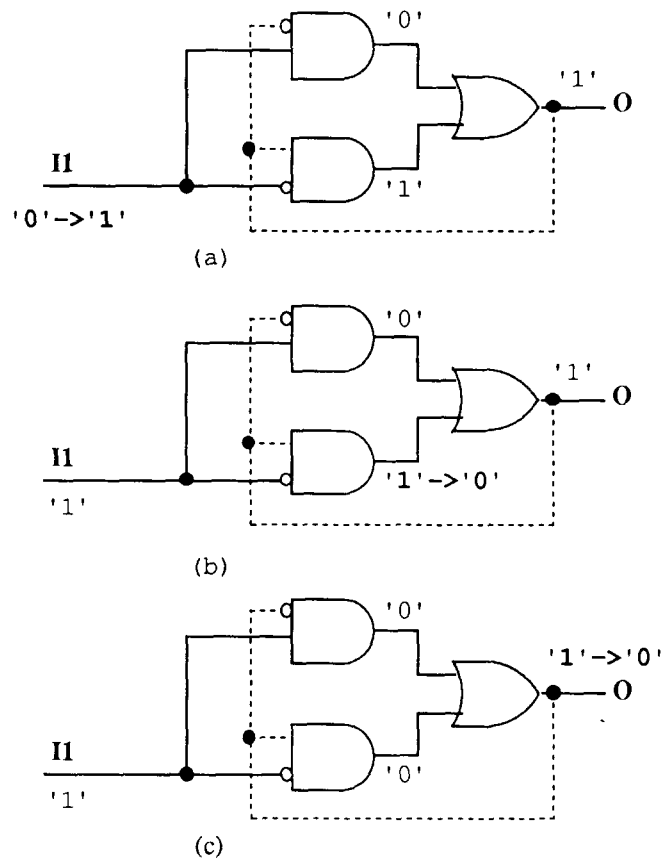


Fig. 1 An example of event driven simulation

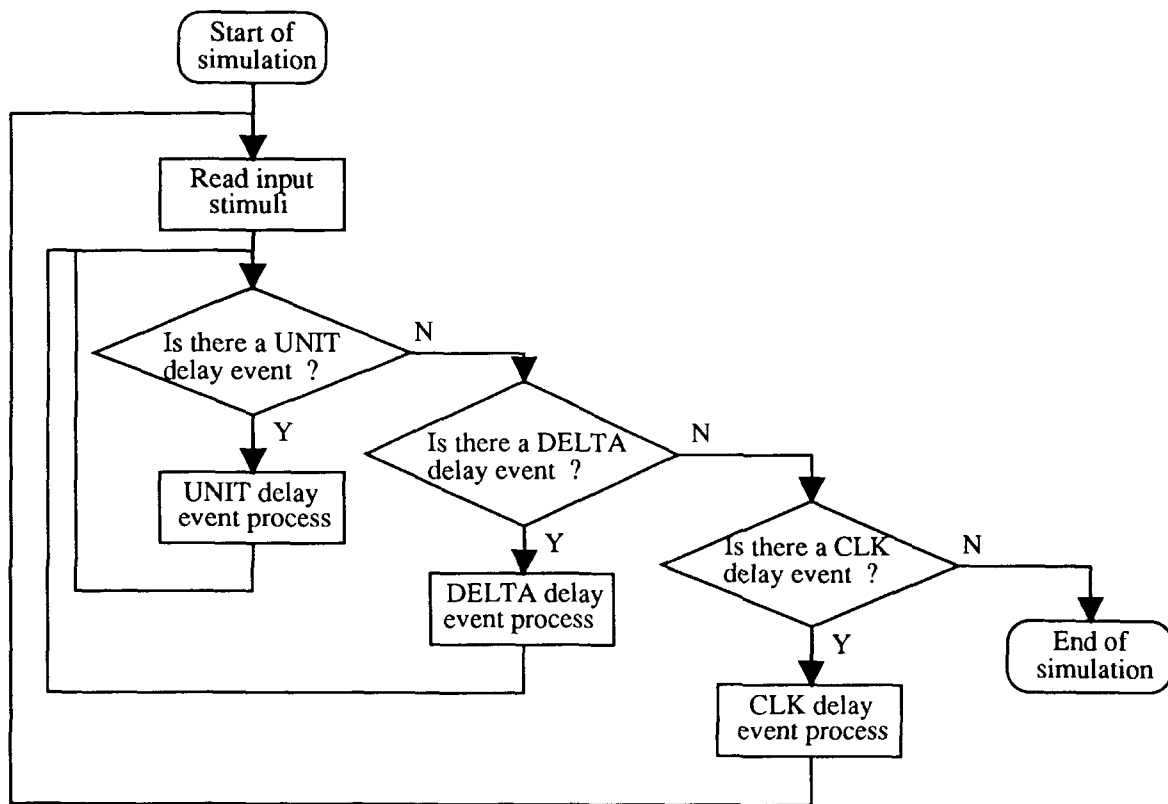


Fig.2 Execution flow of VHDL simulation with ZERO, DELTA, and CLK delay

Each of event has one of these delay types. Figure 2 shows the evaluation phase flow.

Figure 3 shows the processing order for each delay type. Vertical lines denote event processes for the delay type identified by the characters "C", "D", or "U" representing CLOCK, DELTA, or UNIT.

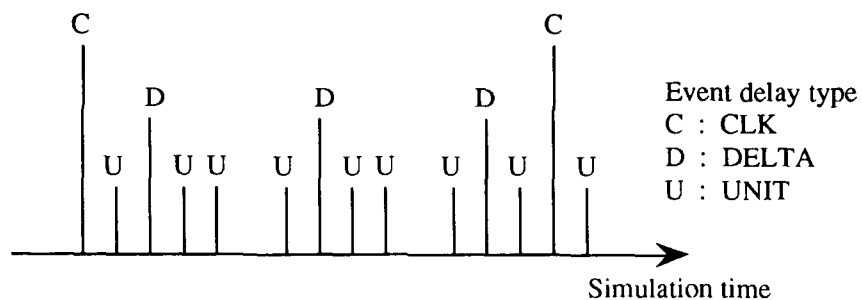


Fig.3 Processing order during VHDL simulation

DELTA, and UNIT delays.

4. Conversion Methods

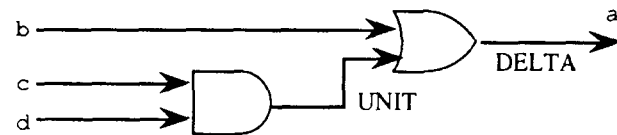
4.1 Concurrent Signal Assignment Statements

Concurrent signal assignment statements are realized by combining simulation machine primitives. Except for assignments which require DELTA delay, the delay type of each event is UNIT. Figure 4 shows an example statement where Fig.4(a) shows a part of VHDL listing and Fig.4(b) shows the simulation model that realizes the VHDL listing. The delay type of each event is identified by "UNIT" or "DELTA" strings next to each connection.

In some cases, the order of evaluation must be determined exactly in advance. We realize these statements with the method described in the next section.

```
a <= b or (c and d);
```

(a) VHDL code of concurrent signal assignment statement



(b) Connection between each primitive

4.2 Sequential Statements

VHDL provides the capability of writing sequential statements as programming languages for workstations. Compilers for workstations decompose statements into simple operations. The execution order of the operations is expressed by flow graphs. Our method applies this to simulate sequential statements by simulation machines.

Fig.4 Simulation model for concurrent signal assignment statement

Figure 5(a) shows part of a VHDL statement listing. Gate-level descriptions cannot be directly synthesized from the listing. Our method can, however, make the simulation model for gate-level logic simulation machines from statements. Figure 5(b) shows the simulation model that realizes the listing. Each rectangle denotes a group of primitives that executes a complicated evaluation. The line type meanings are the same those explained in the section 3.1. Except for the line with the "DELTA" string, the delay type of these lines is UNIT. The order group execution is controlled by evaluate events. Each group belongs to one of the 3 types described below.

a) Data evaluation

Groups that have ">", "+", "MEMREAD," or "SEL" strings in Fig.5(b) belongs to this type. Each SEL group has two input pins. It outputs the value of the input pin

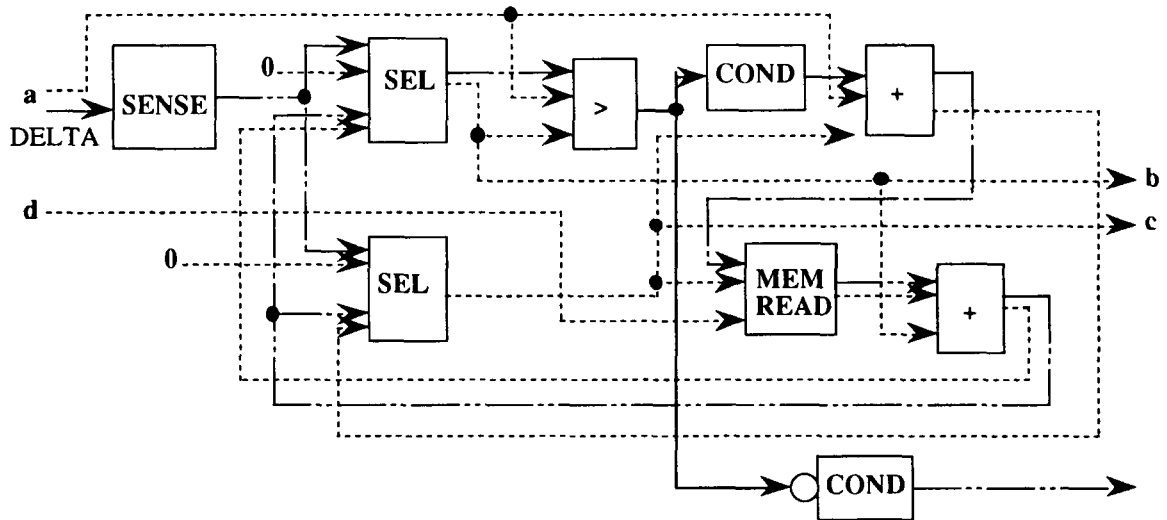
```

signal d : ARRAY_INT;
.....

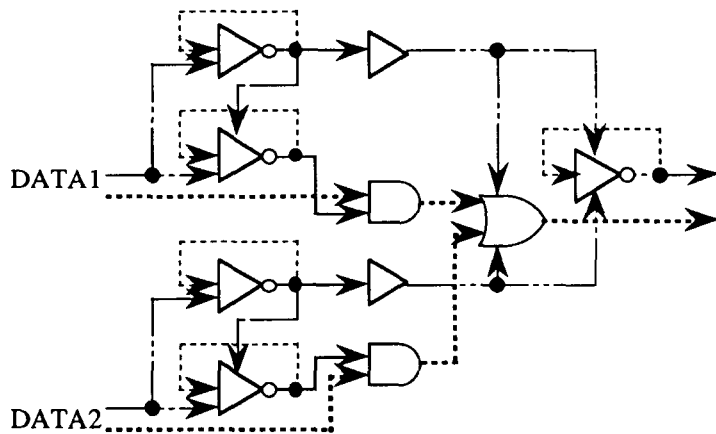
process(a)
  variable b, c : Integer;
begin
  b := 0;
  c := 0;
  Loop1:
  while a > b loop
    c := a + c;
    b := b + d(c);
  end loop Loop1;
  .....
end process;

```

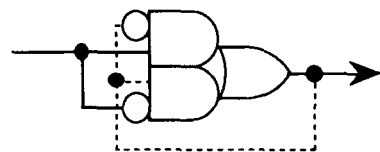
(a) Part of VHDL listing for sequential statement



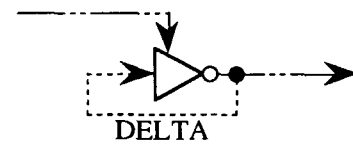
(b) Simulation model for VHDL listing of (a)



(c) SEL group



(d) COND group



(e) SENSE group

Fig.5 Simulation models by group of primitive

that received evaluate event. Figure 5(c) shows the simulation model that realizes the SEL group.

b) Execution control

Groups with the "COND" string in Fig.5(b) belong to this type. This group sends an evaluate event if the input pin's value is '1'. It is used to execute conditional branches. Figure 5(d) shows the simulation model that realizes the COND group.

c) Event sense

The group that has a "SENSE" string in Fig.5(b) belongs to this type. This group sends only one event at each DELTA delay process, even if input pins receive plural events. Figure 5(e) shows the simulation model that realizes this group.

5. Performance Estimation

5.1 Gate-Level Logic Simulation Machine

We estimated the simulation speed when we use the gate-level logic simulation machine SP [3]. First, we show the machine specifications:

- (1) SP has up to 64 processors, that execute simulate in parallel.
- (2) One of two event delay types (i.e., "0" or "1") is independently set for each processor. Figure 6 shows the SP execution flow.

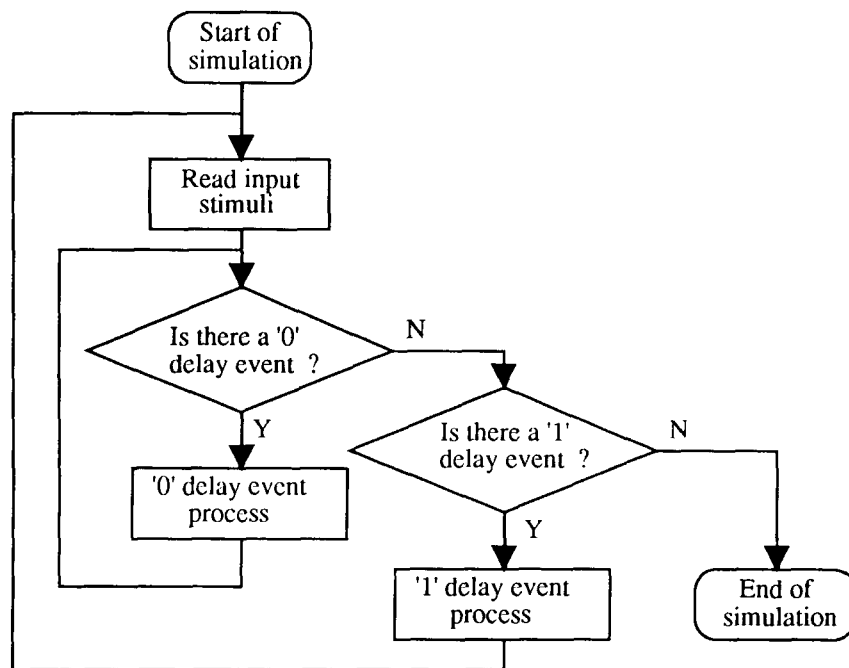


Fig.6 Execution flow of logic simulation on SP

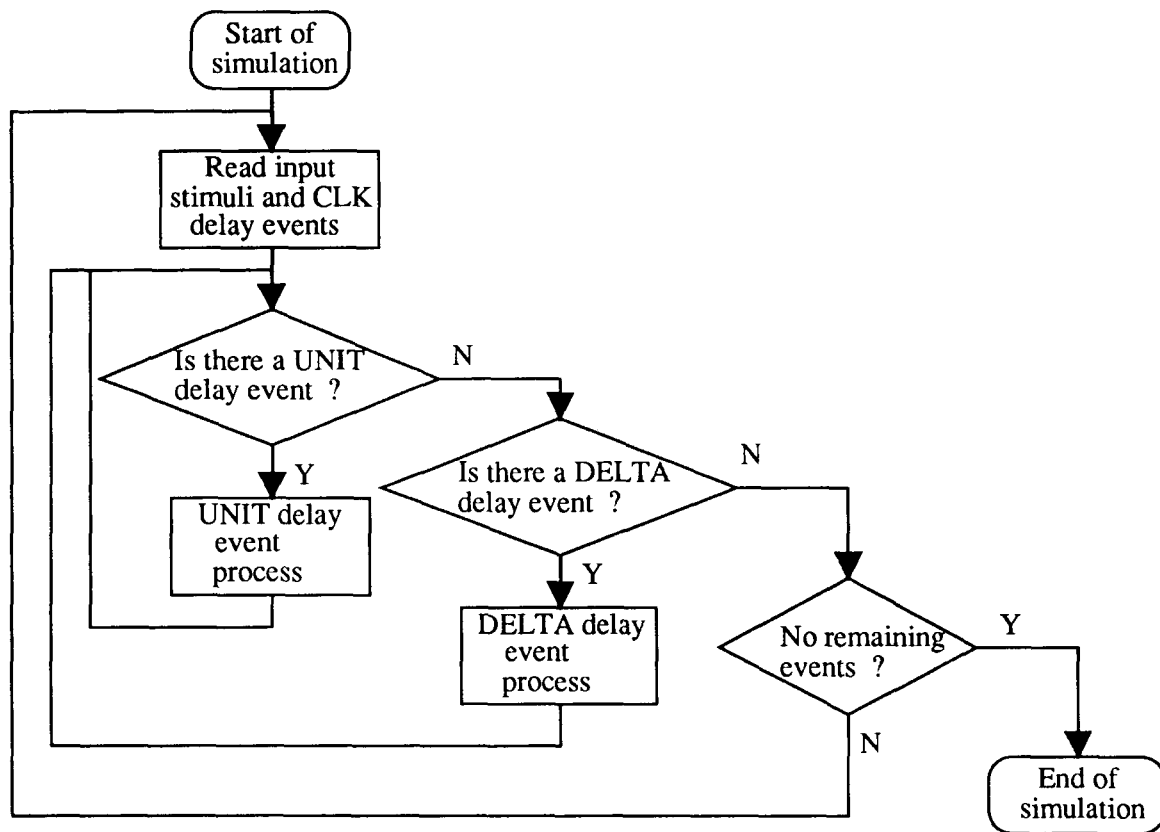


Fig.7 Execution flow of VHDL simulation on SP

The three types of delay of VHDL simulator are implemented on SP by assigning "0" for UNIT delay or "1" for DELTA delay. The CLK delay type is realized by reading input stimulus when no processor still contains an event. Figure 7 shows the flow of evaluation phases for VHDL simulator using SP.

5.2 Extracting Flow Graphs

The flow graphs for sequential statements are extracted from VHDL listings using a CLSI parser and pre-processor for circuit synthesis. Because the pre-processor has restrictions for supported statements, this system does not support some statements and data types.

5.3 Simulation Speed

Naturally, simulation speed depends on the statements. We evaluated the simulation speed of a real circuit, consisting of 200k gates which was described by 20k lines of VHDL statements.

We estimated the simulation speed by the number of simulated clock cycles in a second. The simulation speed using software simulator on a workstation (SUN4/470) is described below.

The elapsed time was 40 seconds to simulate 105 clock cycles that evaluated 1.2M lines of the description.

Hence the simulation speed is about 2.5 clock cycles per second.

We estimated the SP simulation speed as follows:

Assuming the efficiency due to parallel simulation with these processors is

$$64 \times 0.3 = 19.2,$$

the number of clock cycles this machine must spend to evaluate one primitive is 1.82[4].

By the investigation of this example, each line of VHDL listing is converted to about 10 primitives. Hence, the circuit under the analysis is comprised of about 200k primitives.

We assumed the event rate during one circuit clock cycle is 50%.

If we assume the SP clock cycle is 100 ns, the estimated SP simulation speed is:

$$\frac{1}{200000 \times 0.5 \times \frac{1.82}{64 \times 0.3} \times 10^{-7}} = 1055 \text{ clock cycles per second.}$$

The SP simulation speed is about 400 times faster than that of software simulators. Table 1 summarizes these results.

5.4 Required Memory Size

When we simulated the LSI described in 4.3 with software simulator, the required memory size was about 15 Mbytes. If we use SP, we will be able to simulate up to 4M primitives.

Table 1 Simulation speed

	Circuit	Simulation speed	
			CLK/s
Software simulator	20,000 lines	30,000 [lines/s]	2.5
SP (estimated)	200,000 gates	105,494,505 [events/s]	1,055

Since we estimate that one line corresponds to 10 gates on average, the capacity of 400k VHDL lines corresponds to 4M gates.

If we simulate this VHDL listing by software simulator, 300 Mbytes of memory would be required. This is too large for fast software simulation. In this case, the ratio of simulation speed will be larger than the result obtained in the previous section.

6. Conclusion

We developed the method to directly simulate VHDL high-level behavioral descriptions using gate-level logic simulation machines. This method is effective for a VHDL subset that is enough to describe high-level behavioral descriptions for logic verification of synchronous circuits. The subset is (1) no dynamic memory allocation, (2) no time type, (3) no string messages. We introduced 3 types of delay to simulate synchronous circuits.

We estimated the simulation speed by our method using SP as the gate-level logic simulation machine. We found that the simulation speed of our method is about 400 times faster than software simulator.

We are presently developing a compiler that uses this method.

References

- [1] "IEEE Standard VHDL Language Reference Manual," IEEE Std. 1076-1987, IEEE, New York, NY, 1988.
- [2] R. Lipsett, C. Schaefer, and C. Ussery, "VHDL: Hardware Description and Design," Kluwer Academic Publishers, Norwell, MA, 1989.
- [3] F. Hirose, "Simulation Processor SP," Proc. of IEEE International Conference on Computer Aided Design, pp. 484-487, 1987.
- [4] F. Hirose, "Performance Evaluation of an Event-Driven Simulation Machine," Proc. of 29th Design Automation Conference, pp. 428-431, 1992.