Acceleration of Behavioral Simulation on Simulation Specific Machines

Minoru Shoji, Fumiyasu Hirose, Shintaro Shimogori, Satoshi Kowatari, Hiroshi Nagai
CAD group, FUJITSU LIMITED
4-1-1, KAMIKODANAKA NAKAHARA-KU, KAWASAKI, 211 JAPAN
E-mail: shoji@fd.cad.fujitsu.co.jp

Abstract

Behavioral simulation is faster than gate-level logic simulation, however, the simulation speed is too slow for large systems. Simulation specific machines accelerated simulation by parallel processing. We developed the method to extract parallelism from behavioral descriptions for fast simulation utilizing these machines.

We evaluated our methods utilizing CAD accelerator TP5000. By the extraction of the parallelism the simulation speed is accelerated about 7 times.

1 Introduction

In order to quickly develop a large and complex digital system, we need logic verification at as early design stages possible. Therefore, we describe the design in high-level behavioral descriptions by languages as VHDL and employ behavioral simulation. Behavioral simulation is about 10 to 100 times faster than gate level one, however, the simulation speed is still slow for a large scale design.

For the acceleration of behavioral simulation, we must employ simulation specific machines composed of a number of processors that simulate a part of the description in parallel. These machines accelerate simulation utilizing the parallelism of gate-level simulation models. If we utilize these machines for behavioral simulations, the simulation speed is limited by the parallelism of the description. High-level behavioral descriptions consist of a number of sequential statements. The parallelism of those descriptions is not high as that of the gate-level descriptions. For the speeding up of simulation by utilizing simulation machines, we must parallelize those behavioral descriptions.

It is the purpose of this paper to present methods to parallelize high-level behavioral descriptions. Experimental results showed that the simulation speed utilizing a simulation specific machine is accelerated 7 times by our methods.

2 Simulation model and the problem

To accelerate logic-simulation, we employed logic simulation specific machines[1][2]. These machines are composed of many processors, simulating gate-level simulation models in parallel. The speeding-up by these machines was very high[3][4] when simulating gate-level descriptions. If we utilize these machines for behavioral simulation, we must convert behavioral descriptions to the models suitable for these machines. In this section, we describe the simulation model for high-level behavioral descriptions, and the problem of the model for speeding-up of simulation.

2.1 Simulation model

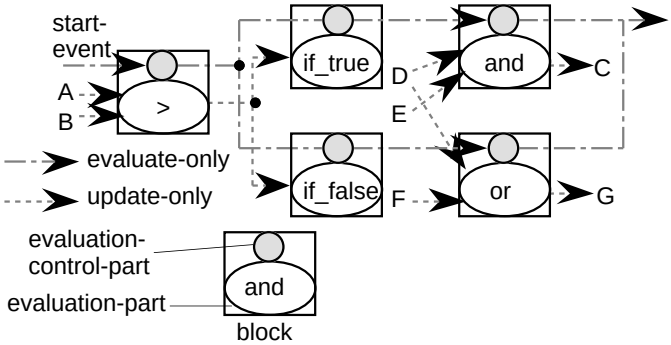
The simulation models for behavioral descriptions are divided into concurrent models and sequential models. Concurrent signal assignment statements of VHDL are converted

ED&TC '97 on CD-ROM

to concurrent simulation models. Concurrent simulation models are equivalent to the gate-level simulation models. Each sequential model corresponds to a sequence of statements as process statement of VHDL. The models consist of operators and the controllers that control the execution order of operators. The operators and controllers consist of gate-level simulation models. To realize this model, we split the function of event of event-driven simulation algorithm. The first one is value delivery. The second one is instruction to start the evaluations of fan-out gates. We call the event that has only the first function as “update-only event”, and the event that has only the second function as “evaluate-only”. Figure 1 shows an example of VHDL sequential statements and the corresponding simulation models. Each rectangle shown in figure 1(b) denotes the ‘block’ corresponds to an operator of the statement. Line connecting blocks denotes connections of events. Each block consists of an evaluation-part and an evaluation-control-part. When the evaluation-control-part receives an “evaluate-only event” from the previous block through a broken line, then the result of the operation is evaluated by the evaluation-part. The result is fed to other blocks through dotted lines by “update-only event”. The blocks labeled ‘if_true’ or ‘if_false’ send an event only if the input value is true or false respectively.

```
if A > B then
    C := D and E;
else
    G := D or F;
end if;
```

(a) an example of VHDL description for sequential statement



(b) simulation model for (a)

Figure 1 simulation model for sequential statements

2.2 Problem

Simulation specific machines accelerated simulation utilizing a number of processors. Each processor simulates a part of circuit concurrently. For the gate-level simulation models, each gate is simulated in parallel in each processor. If the time to simulate a gate is unit-time, each processor simulates each part in parallel. It is unnecessary to wait the end of the simulation by other processors.

The delta-delay of VHDL is defined as the number of unit-time between the start of process statements and the end of simulation of all process statements. For the gate-level description, a delta-delay is simulated in a unit-time. However, for the behavioral simulation, the time for simulation of a delta-delay is determined by the longest process statement that started simulation. For the processor that simulates shorter model, the time after the end of its simulation is the waste of time. The parallelism factor is reduced to one for the duration. To utilize the parallelism of simulation specific machines for the fast simulation, the length of each process statement model should be minimized.

Behavioral descriptions consist of many sequential statements because of the ease of writing algorithms or protocols. Therefore, the problem of long process statements arises during behavioral simulations. In the following section, we describe the methods to minimize the length of statements and extract maximum parallelism from behavioral descriptions.

3 Increasing parallelism factor

We developed following three methods to increase parallelism factor, (1) extraction of independent parts from descriptions, (2) conversion to concurrent models, (3) extraction of parallel parts from descriptions.

3.1 Independent parts extraction

There are descriptions that contain parts that can be simulated independently. The behavioral descriptions written in VHDL contain many those parts. Figure 2 shows a part of VHDL process statement. The part A of the example con-

tains only one conditional branch (if statement) and a number of signal assignment statements within the conditional branch. VHDL defines that the values assigned to signals become valid after delta delay. In this example, the value of signal assigned in part A becomes valid at the next simulation of the process statement. Therefore, there is no dependencies between part A and part B. This means that we can simulate part A and part B in parallel. In this example, part B consists of three statements having no dependencies each other. These statements can be simulated in parallel.

The study of real descriptions showed that the longer models usually contain loop statements. Circuit designers always use loop statements for the descriptions of operations between array objects. In most cases, we can determine the number of loop iterations. We utilized 'loop-unrolling and constant propagation' method to reduce the length of sequential simulation models. Figure 3 shows an example of the application of this method. Figure 3(a) shows a part of a description with a loop statement. Each rectangle shown in the figure denotes a block defined as figure 1. Line denotes the connection of the events. The connections of the values are omitted in this figure. The simulation of the model shown in figure 3(b) requires total number of 19 blocks evaluated. By the application of this method, the length of the model becomes 19 to 4.

There are cases where the size of the simulation model increases by this method. For the object code of conventional computers, large code size increases the probability of miss hit of instruction cache. This causes the decrease of execution speed. However, simulation speed of simulation specific machine is determined by the number of evaluations and the length of descriptions. This method reduces the number of evaluations and the length of the description. Therefore, the simulation speed is accelerated.

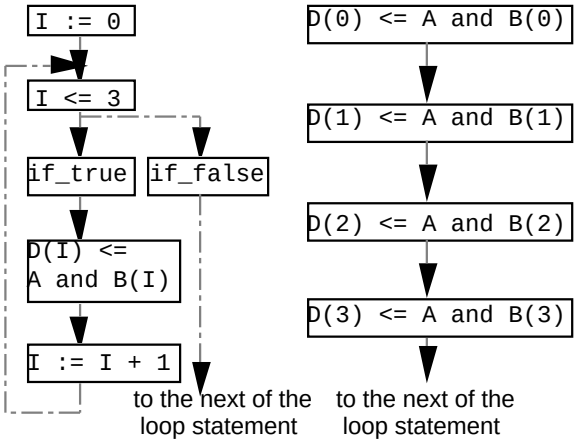
```
process
begin
.......
if A = '0' then
    S <= SOLD;      part A
    T <= TOLD;
end if;

U := S and T;
V := S or T;        part B
W := S xor T;
.......
end process;
```

Figure 2 dependencies of signals and variables

Furthermore, in many cases, the relations between the statements disappear after the application of this method. Figure 3(c) shows that there are no relations between each statement. Therefore, we can split the description into 4 small descriptions.

```
for I in 0 to 3 loop
    D(I) <= A and B(I);
end loop;
```

(a) a VHDL example



(b) original simulation model (c) after loop-unrolling and constant propagation

Figure 3 an example of loop-unrolling and constant propagation

3.2 Conversion to concurrent models

Because simulation specific machines are designed to accelerate gate-level simulation, the gate-level simulation models are suitable for those machines. If we can convert sequential descriptions into gate-level simulation models, the simulation speed is accelerated.

If a sequential description satisfies some conditions, we can convert the corresponding simulation model to a number of concurrent simulation models. The conditions are described below.

(1) There is no loop statement or the variable/signal dependency that forms a loop.

(2) The statement has sensitivity list and all the signals read from the process statement is declared in the sensitivity list.

(3) There is no variable that should be stored after the end of the simulation of process statement.

Figure 4 shows an example of VHDL process statement and the corresponding sequential and concurrent models. Figure 4(b) shows the blocks and the event connections. In

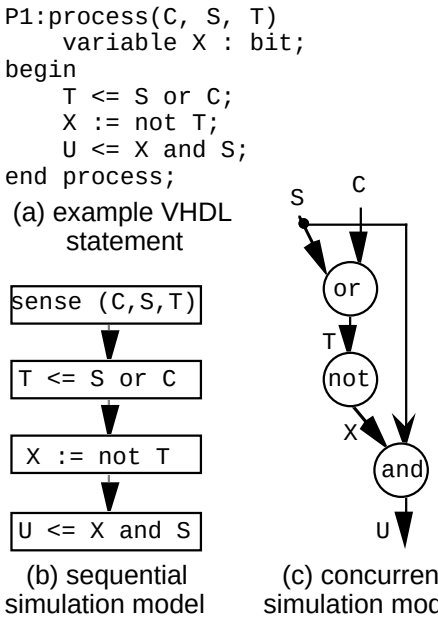


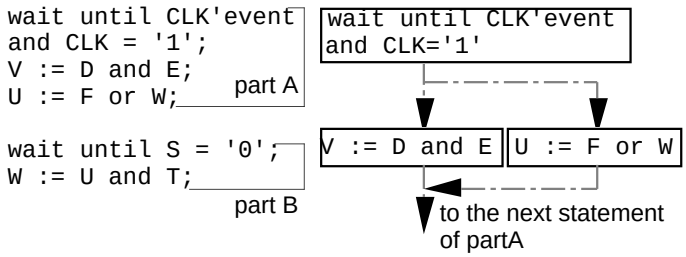
Figure 4 an example of conversion to concurrent model

this case, we can convert this model to the concurrent simulation model shown in figure 4(c). A circle shown in the figure denotes a concurrent simulation model for each operation This model corresponds the evaluation part of figure 1(b) except that it starts the evaluation only when any input values are changed. A line connecting them denotes the data flow that may start other operations. The original description is simulated every time when at least one of the input signals changed the values. If we utilize event-driven simulation algorithm, we don't have to simulate the part whose input values have not changed. Furthermore, concurrent models do not need any evaluation-control-part for sequential models, which reduces the number of evaluation for the part. This increases the simulation speed furthermore.

3.3 Extraction of parallel parts

Figure 5(a) shows another example of VHDL statements. In this case we must consider the activation and suspension statements that is 'wait' statement of VHDL. Because the description contains wait statements, we can not utilize the methods described in previous sections. In this section, we describe the method that accelerates simulation of those descriptions.

If the simulation of the statements reached to the wait statement, the simulation of the process statement is suspended until the condition written in the wait statement becomes true. In this case, we must split the process statement to a number of parts starting from the wait statements. During the simulation, only one part is simulated. Therefore the dependencies of variables are removed at each wait statement. In this example, the variable U is assigned new value in part A, and the value is used in part B. This dependency is ignored because there is a wait statement between them.



(a) a part of VHDL description with wait statements (b) parallelized simulation model of part A

Figure 5 an example of VHDL with wait statement

After this partitioning, we can find dependencies between statements within each part. The part A of figure 5(a) consists of 3 statements. There is no dependency between the second and the third statement. Therefore, we can simulate those statements in parallel. Figure 5(b) shows the parallelized simulation model for the part A of figure 5(a).

Furthermore, we can apply algorithms utilized by the compilers for super scalar machines, VLIW machines, or vector machines[7][10] for the extraction of parts simulated in parallel.

4 Experimental results

We experimented the methods described before using unit-delay event-driven VHDL simulator running on CAD accelerator TP5000. First we describe the specifications of TP5000. Next we show the experimental results.

4.1 TP5000

TP5000 accelerates gate-level simulation at about 100 times than software simulators. It consists of a number of processor groups, each of which contains 15 processors. Each processor has micro code memory to store the executable code

for the processor. They also have fast memory to store a part of simulation models. The processor group forms a pipe-line realizing event-driven unit-delay logic simulator. Each pipe-line simulates a part of simulation models concurrently.

4.2 Results

We used the description of two commercial circuit systems for the experimentation. The first system (circuit system A) consists of descriptions of a processor and a number of pseudo circuit for the simulation. The descriptions consist of 106k VHDL description steps and contain 936 process statements. The second system (circuit system B) consists of descriptions of a processor. The descriptions consist of 40k VHDL description steps and contain 172 process statements.

Table 1 shows the number of unit-time spent for each delta-delay of the descriptions, and the simulation speed ratio of (a) original descriptions, (b) apply methods described in section 3.1 and 3.2, and (c) apply all methods. The maximum number of unit-time is the largest number of unit-time for a delta-delay during the simulation. The simulation speed is the ratio to the original description. By the application of our methods, the average number of unit-time is reduced to 15% of the original. The simulation speed becomes 6.7 times faster than that of the original descriptions. For the circuit system B, the average number of unit-time is reduced to 6%. The simulation speed for circuit system B becomes 7.8 times faster. By those results, our method accelerates simulation at about 7 times.

Table1 results of experiments

Circuit system	A			B		
Number of process stmt.	936			172		
Simulation model	(a)	(b)	(c)	(a)	(b)	(c)
Ave. number of unit-time	141.4	40.57	21.12	51.4	3.1	3.0
Max. number of unit-time	4209	4209	4195	8419	527	86
Simulation speed	1	4.5	6.7	1	7.5	7.8

5 Conclusions

We showed that speeding-up of the behavioral simulation utilizing simulation specific machines is restricted by the length of statements. We developed the methods to reduce the length of each sequential simulation model and to convert sequential simulation models to concurrent simulation models for maximizing the parallelism of the simulation machines. The experimental results using real systems descriptions showed that the methods accelerated simulation at about 7 times.

References

[1] T. Blank, “A survey of hardware accelerators used in computer aided design,” IEEE Design and Test of Computers, 1, pp.21-39, 1984.

[2] F. Hirose, “Simulation Processor SP,” Proc. of IEEE International Conference on Computer Aided Design, pp. 484-487, 1987.

[3] S.Shimogori, K.Takayama, H.Matsuoka, “TP5000: Reconfigurable Hardware Accelerator for CAD Applications,” FUJITSU Sci. Tech. J., 31, 2, pp.152-160, 1995.

[4] F. Hirose, “Performance Evaluation of an Event-Driven Simulation Machine,” Proc. of 29th Design Automation Conference, pp. 428-431, 1992.

[5] M. Shoji, and F. Hirose, “High-Level VHDL Simulator Running on Logic Simulation Machines,” Proc. of VHDL International Users’ Forum Spring 1993, pp. 145-154, 1993.

[6] M. Shoji, F. Hirose, and K. Takayama, “VHDL compiler of behavioral descriptions for ultrahigh-speed simulation,” Proc. 2nd Asian Pacific Conference on Hardware Description Languages, pp.85-88, 1994.

[7] A.V.Aho, R.Sethi, and J.D.Ullman, “Compilers,” Addison-Wesley Publishers, 1986.

[8] “IEEE Standard VHDL Language Reference Manual,” IEEE Std. 1076-1987. IEEE, New York, NY, 1988

[9] R.Lipsett, C.Schaefer, and C.Ussery, “VHDL: Hardware Description and Design,” Kluwer Academic Publishers, Norwell, MA, 1989.

[10] J.L.Hennessy and D.A.Patterson, ”Computer Architecture: A Quantitative Approach,” Morgan Kaufman Publishers, San Mateo, CA, 1990.