병렬처리 논리 시뮬레이션에서 클럭 진행의 개선

Enhancement of Clock Advancement in Parallel Logic Simulation

정연모*
Yunmo Chung

Abstract

Efficient event evaluation and propagation techniques are proposed to enhance the advancement of simulation clocks of conservative and optimistic logic simulation protocols on parallel processing environments. The first idea of the techniques proposed in this paper is to allow more than one event evaluation per simulation cycle and to pack more than one propagation event in a single message. The second idea is to use advancement windows in both simulation protocols. The proposed multi-event techniques and advancement windows resulted in good performance in parallelism and execution times.

1. Introduction

In parallel logic simulation using traditional distributed event-driven simulation protocols such as the Chandy-Misra algorithm[3,4], which is a well-known conservative simulation technique, and Time Warp[7,8], which is the most popular optimistic simulation, an event is used to carry a value associated with a timestamp. Each gate computes a local virtual time (LVT) which is the smallest timestamp of unprocessed events. All the events with timestamps equal to LVT are involved in event evaluation at each gate during a simulation cycle (or iteration) based on the event selection (or scheduling) policy of a simulation protocol. Event evaluation means that an output value is computed based on the gate type with all the chosen events. After event evaluation, an output event may be propagated to successors.

The main idea proposed in this paper is to allow more than one event evaluation per simulation cycle and to pack more than one propagation event in a single message. In other words, evaluation of many events in each simulation cycle is allowed, reducing the number of simulation cycles, communication costs, and execution times. A sequence of events to be propagated as a single message is called a *multi-event* in this paper. In parallel processing environments, communication overhead becomes very significant as the number of processors increases. With multi-events, we can significantly reduce communication cost since a set of events is sent as a single message.

* 경희대학교 전자공학과

In conservative logic simulation, each gate computes an advancement window containing events which can be safely executed without violating event execution precedence. In optimistic simulation, an aggressive advancement window is given which contains events to be aggressively evaluated. More than one evaluation is allowed per simulation cycle even though rollback frequency increases.

The proposed advancement windows and multi-event techniques were implemented for some ISCAS'85 benchmark circuits on the CM-2. Good performance, measured by parallelism and execution time, was obtained for some benchmark circuits. We investigate the effects of window sizes on performance. As the window size increases, execution time decreases initially, but then increases since time taken for each simulation cycle increases due to the increases window size.

2. Multiple Events in a Message

Gate-level logic simulation is different from other simulations, such as queueing network simulation, high level circuit simulations, etc. since simulation is performed based on fixed propagation routing, fixed delay, good lookahead capabilities, and non-preemption. The fixed propagation routing allows a gate to execute more than one event in a simulation cycle based on a specific simulation protocol: optimistic or conservative.

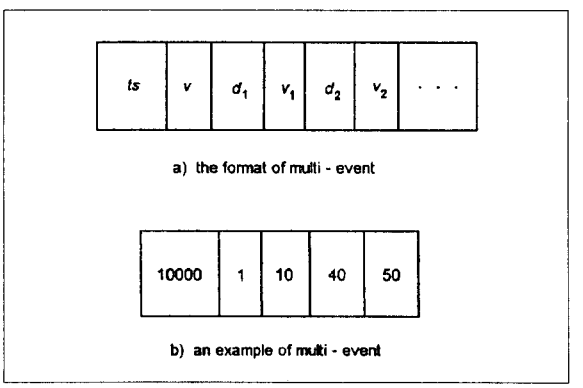

Figure 1. A multi-event

Multi-events will be discussed, in detail, which are contained in a single message. A message is an object which transfers information between gates. The information carried by the message represents a set of events as follows.

Definition 1. *A multi-event* contains a sequence of events to be propagated.

A multi-event contains a sequence of timestamps in non-decreasing order to propagate the results of the whole event evaluation within an advancement window. A multi-event has the form as shown in Figure 1(a). In the figure, *ts* represents the timestamp of the first event of a multi-event, and v indicates the signal value of the first event. In addition, d_i contains the timestamp difference between the first event and the $(i+1)$-th event in the multi-event, and corresponding signalv_i follows. In event-driven simulation, only events whose output signal is different from the output signal of the previous event evaluation must be propagated.

Depending on what parallel machine is used, the format of a multi-event may vary. For example, if the computation of the difference in timestamps takes too much time, the timestamp itself is sent instead of the difference. The signal fields(v *or* v_i)may not be necessary in a multi-event if binary signals, either 0 or 1, are used and we can make sure that the signal value of the last event in the most recently sent multi-event is different from the value of the first event in the multi-event to be sent.

Time Warp has an antimessage used to nullify previous incorrect event propagation. In this case, the message contains only a timestamp as its information.

For example, suppose binary signals are used and that there are propagation events $e(10000,1)$,$e(10010,0)$,$e(10020,0)$, $e(10040,1)$, and $e(10050,0)$. The message to be sent will contain the *multi-event* shown in Figure 1(b). In traditional distributed event-driven simulation, at least 6 simulation cycles and 4 send operations for the above evaluation and propagation are required. With the proposed advancement windows and multi-events, however, all the above evaluations may be performed and their results will be propagated to successors during the same simulation cycle.

2. Clock Advancement Windows

In this section, the concepts of clock advancement windows in conservative simulation and optimistic simulation are discussed. Since the advancement window techniques are implemented on a massively parallel SIMD (Single Instruction Multiple Data Stream) machine, terminology needed for implementation on the machine will also be introduced.

Definition 2. Event evaluation at simulation time t is defined to be *safe* only if it is certain that no event with timestamp less than t can arrive in the future. Otherwise, event evaluation is called *unsafe*.

2.1 Advanced Conservative Logic Simulation

Link clocks and minimum link clocks are used. That is, a link clock input port of a process is defined to be the timestamp of the most recently arrived event along the input link. The minimum link clock of a process is the minimum of link clocks of all its input ports.

Based on the concept of safety, we define the following advancement window in conservative simulation.

Definition 3. In conservative simulation, a window at a gate can be defined from LVT($=l$) to minimum link clocks ($=c$) where $l \leqq c$. Such a window, which contains the events to be involved in safe event evaluation, is called a *Maximum Conservative Advancement Window* (MCAW).

On MIMD (Multiple Instructions Multiple Data Stream) machines, all (or some) events within the maximum conservative advancement window can be executed for a simulation iteration (or cycle). All gates are synchronized in SIMD processing environments. Hence, all gates must wait until the slowest (or busiest) gate finishes its event evaluations. To avoid that situation, we limit the number of event evaluations.

Definition 4. A *Synchronized Conservative Advancement Window* (SCAW) is defined as the synchronized window which contains only events to be involved in safe event evaluation. The size of the window is defined as the number of event evaluations from LVT.

Definition 5. If advancement windows are applied in logic simulation, each simulation cycle has a repeated procedure which consists of LVT computation, choosing active gates, and event evaluation. This procedure is called an *advancement cycle*. Only one event evaluation is allowed in an advancement cycle.

A simulation cycle has one or more advancement cycles. For example, Figure 2.(a) shows the state of a gate during simulation. The minimum link clock of the gate is 60. All events from LVT 10 to the minimum link clock can be involved in safe event evaluation, as shown in figure 2.(b). In this case, an event evaluation means that an output value is computed based on all the events with the same timestamp. For example, the event evaluation at 10 in Figure 2.(b) involves two events stored in queues Q1 and Q2, respectively. All the events in the MCAW can be conservatively evaluated without violating event processing precedence. In other words, the MCAW consists of a sequence of safe event evaluations. In the figure, since SCAW size 3 is used, the maximum number of LVT computation and event evaluations at a simulation cycle is three.

Each gate might have a different MCAW at a certain simulation cycle, as shown in Figure 3. Gates 1, 2, and 3 have MCAW of sizes 5, 2, and 4, respectively. We assumed that SCAW size is 4. A gate whose MCAW is not smaller than a SCAW, as shown in gates 1 and 3 of the figure, performs event evaluation at each advancement cycle of the simulation cycle, as shown in gate 3 of the figure. Otherwise, the gate may be idle after the first few advancement cycles. It is very important to choose an appropriate SCAW size for good performance. The performance will be analyzed according to the size in section 3. In advanced conservative logic simulation with advancement windows, the following steps are performed at each gate for each simulation cycle.

[Algorithm ADVANCED CONSERVATIVE]

1. The minimum link clock is computed.
2. The following advancement cycle is repeated as many times as the predetermined advancement window size.

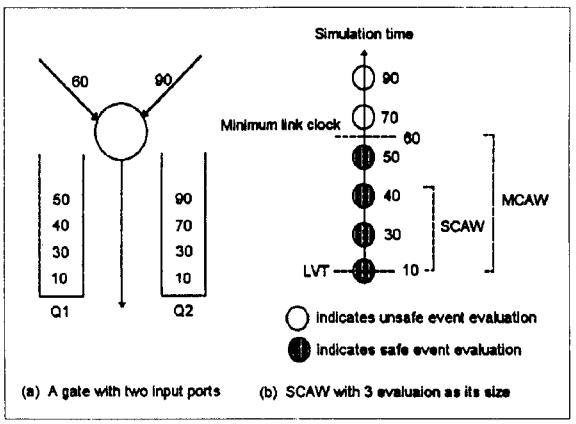

Figure 2. An example of an advancement window

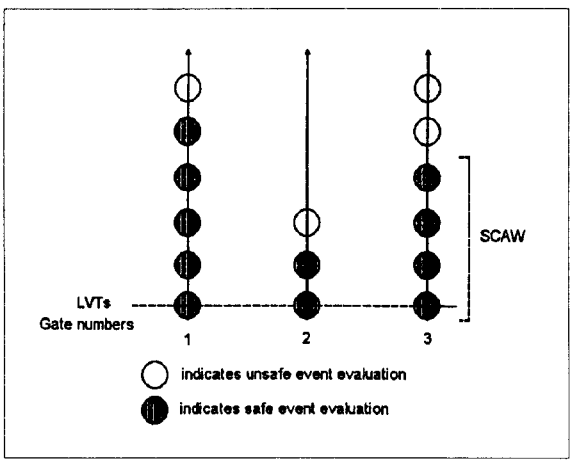

Figure 3. Relationship between MCAWs and a SCAW

(a) LVT is computed.

(b) if LVT is less than or equal to the minimum link clock, event evaluation is performed.

3. Event propagation is performed.

4. Each link clock is set to the last timestamp in the multi-event received on the link.

5. Queue manipulation is performed if necessary.

In addition, we can use a global clock to choose active gates for better performance if the above algorithm is implemented on SIMD machines, as given in [6]. For the above algorithm, a multi-event may also carry a virtual time which is the delay of the gate plus the last LVT. This value is useful in setting the link clocks of successors. Conservative simulation with advancement windows can be applied to the null message strategy as well as the deadlock detection and recovery strategy. It is known that conservative simulation with deadlock detection and recovery is very efficient on MIMD machines [4,13], while the simulation with null messages works efficiently on massively parallel SIMD machines [6]. The techniques proposed in this section can be applied effectively in both cases.

2.2 Advanced Optimistic Logic Simulation

In traditional optimistic simulation such as Time Warp, only one LVT increment is allowed at each simulation cycle. In the proposed approach, each gate can advance its LVT one or more times based on a predetermined advancement window size. We define a term for the window.

Definition 6. A *Synchronized Aggressive Advancement Window* (SAAW) is defined as the synchronized window which contains some events from LVT for each gate as far as there are available events to be executed. The size of the window is defined as the number of event evaluations.

If a MIMD machine is used as a target machine, a predetermined advancement window, which need not be synchronized, is applied to each gate during simulation. Event evaluation in optimistic simulation is allowed even when ordering could be violated. In the proposed technique, event evaluation is performed for all events within a SAAW even if rollback may occur later. In advanced optimistic logic simulation with advancement windows, the following steps are performed at each gate for each simulation cycle.

[Algorithm ADVANCED OPTIMISTIC]

1. The following advancement cycle is repeated as many times as the predetermined advancement window size.
   (a) LVT is computed.
   (b) For the first advancement cycle only, rollback is performed if the LVT is less than or equal to the previous LVT.

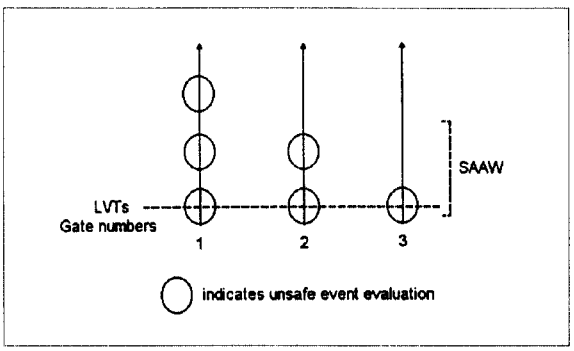

Figure 4. An example of SCAW with size 2

   (c) Event evaluation is performed.
2. Multi-event propagation is performed.
3. Queue manipulation and fossil collection are performed, if necessary.

Since the timestamps of events within a simulation cycle never decrease, only the first advancement of a simulation of a simulation cycle at a gate can cause rollback. Figure 4 shows a simulation example with SAAW size 2. That is, all events within the window are involved in event evaluation at the same simulation cycle. Gates 1 and 2 are busy during the simulation cycle, while gate 3 is idle after one advancement cycle.

When the proposed approach is used, there are some advantages in queue manipulation. All events along the same link will be inserted near each other in the same queue. A disadvantage is that rollback frequency increases.

3. Performance Evaluation

Both conservative and optimistic logic simulation with advancement windows are implemented on the CM-2 with 32K processors. Both simulation protocols were implemented based on the same data structures proposed in [5]. As test input, 1,000 randomly generated input vectors were used. In our simulation, we use the number of event evaluations as an advancement window size.

Let us discuss the performance metrics for the proposed techniques. The performance of the proposed simulation technique will be evaluated by means of the comparison to traditional distributed event-driven simulation techniques. We define the following performance metrics for the evaluation.

- Simulation Cycle Ratio(C_w), which is the ratio of the number of simulation cycles at window size w to the number of simulation cycles at window size 1. That is, simulation at window size 1 indicates the traditional distributed simulation techniques.
- Maximum Queue Ratio (Q_w), which is the ratio of the maximum queue size at window size w to the maximum queue size at window size 1.
- There are two ways of defining parallelism based on how to define an active gate. When an *active* gate is defined as a gate which performs event evaluation at least once during a simulation cycle, *parallelism* is computed as

$$\sum_{i=0}^{i=N} \frac{T_i}{(NS)}$$

where T_i is the total number of simulation cycles for which gate i is active. N is the number of gates, and S is the number of simulation cycles.

Parallelism Ratio(P_w) is defined as the ratio of parallelism at window size w to parallelism at window size 1.

If we define an active gate as a gate which performs event evaluation at an advancement cycle, *adjusted parallelism* at a certain window size is computed as

$$\sum_{i=0}^{i=N} \frac{T_i}{(NSW} \quad)$$

where T_i is the total number of advancement cycles for

which gate i is active and W is the synchronized advancement window size. When the window size is equal to 1, parallelism is the same as adjusted parallelism. The average number of clock advancements (or event evaluation) at window size w for a simulation cycle can be computed by multiplication of adjusted parallelism and the window size.

- Execution Time Ratio (E_w), which is the ration of the execution time at window size w to execution time at window size 1.
- Rollback Frequency Ratio (R_w), which is the ration of the rollback frequency at window size w to rollback frequency at window size 1 in optimistic logic simulation.

In terms of the ratios proposed above, we compare the performance of traditional event-driven simulation and proposed simulation techniques with advancement windows. The maximum number of event evaluations is used as a unit which determines advancement window size. In addition, there are several other ways to define the unit as follows.

- the maximum number of additional fields in a multi-event or
- the maximum number of events to be processed.

3.1 Advanced Conservative Logic Simulation with SCAW

The performance of advanced conservative logic simulation with different SCAW sizes is evaluated in terms of simulation cycle ratio, maximum queue size ratio, parallelism ratio, and execution time ratio. Figure 5 shows the performance comparison with traditional simulation for C1908, and C7552 with 1,000 randomly generated input vectors as a function of SCAW size. The performance in the figure was obtained based on the performance of traditional conservative logic simulation which evaluates gates once per simulation cycle.

As a measurement of how much local simulation clocks advance, the number of required simulation cycle is used. As the amount of clock advancement increases, the number of simulation cycles decreases, since local simulation clock move ahead quickly. As shown in Figure 5.(a), as the SCAW size increases, the number of simulation cycles decreases rapidly and then converges to a certain point.

In Figure 5.(b), as the SCAW size increases, the maximum queue size decreases initially, but then increases. We can explain this as follows. Events in gates are properly processed and consumed as the SCAW size approaches a certain value in the circuit. In this case, the required maximum queue size would become small. But, as the SCAW size increases beyond this value, some gates may have large actually processed advancement windows, while other gates might have small windows. Therefore, some input ports might have a lot of events while other input ports have a small number of events. There might be big differences between link clocks at a gate. Hence, there is a high possibility of having large maximum queue sizes.

In Figure 5.(c), the parallelism increases initially, but then decreases. We can explain this as follows. In the case of a large SCAW size, gates close to primary input ports process many events at each simulation cycle since the gates in general have large SCAW, while gates near primary output evaluate a relatively small number of events due to their small MCAW. That is, if a large SCAW is used, gates in the front part of the circuit complete simulation early, but gates in the rear part may have congestion problems. Therefore, as SCAW size increases beyond a certain point, parallelism might decrease.

In Figure 5.(d), the execution times decrease initially, then increase. The best performance is given when the SCAW size is 4. As the SCAW size increases, the time taken for a simulation cycle becomes longer since we need as many LVT computations and event evaluations as the SCAW size at each simulation cycle. Therefore, the best performance is given at the point where the time spent for a simulation cycle is not long, as well as the number of simulation cycles is not large.

Figure 6 shows the performance with respect to adjusted parallelism, and average number of advancements per simulation cycle. According to the graphs, as SCAW size increases. The average number of advancements increases, and converges to a certain value since there are no more useful advancement cycles if SCAW size exceeds MCAW sizes of all the gates in the circuit being simulated.

Figure 7 shows the performance of advanced conservative

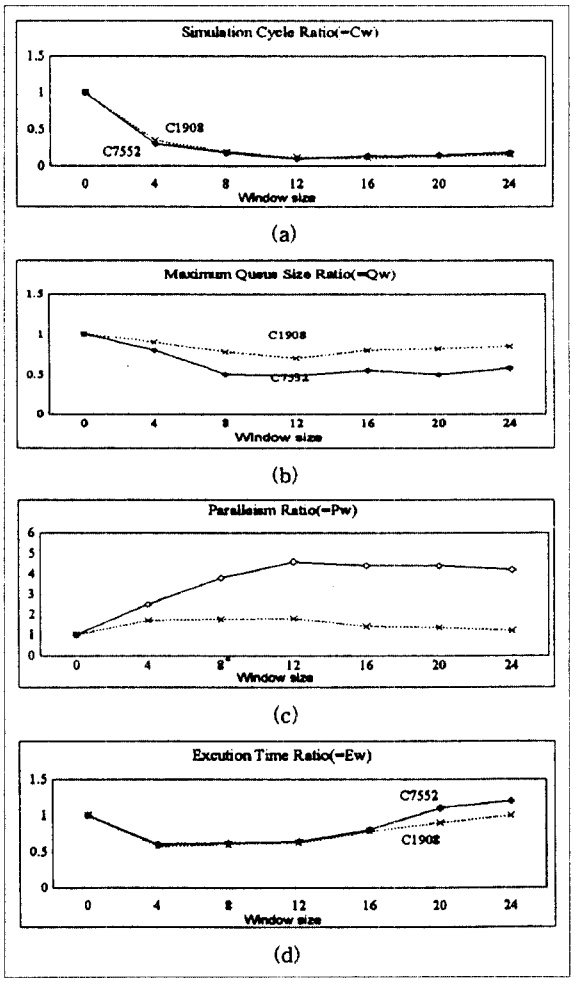

Figure 5. Conservative simulation with different SCAW sizes (1)

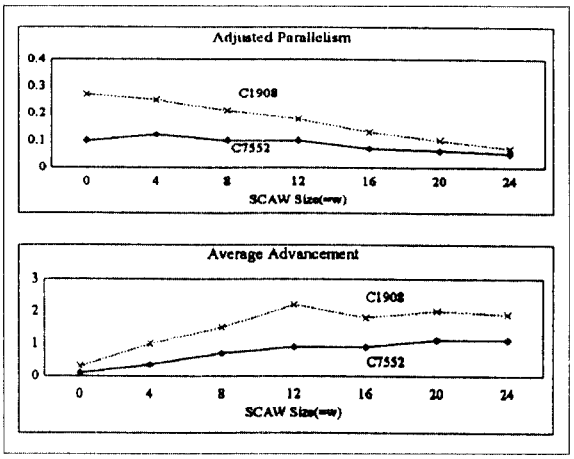

Figure 6. Conservative simulation with different SCAW sizes (2)

logic simulation as a function of the number of input vectors. SCAW size 4 is used since the best performance was given at the SCAW size. As the number of input vectors increases, the number of simulation cycles, the maximum queue size, parallelism, and execution times also increase.

3.2 Advanced Optimistic Logic Simulation with SAAW

Figure 8 shows the performance for C1908, and C7552 as a function of SAAW size. In our simulation, the maximum number of event evaluations was fixed in advance as a synchronized aggressive advancement window. The performance comparison with traditional optimistic simulation was measured in terms of ratios for number of simulation cycles, parallelism, rollback frequency, and execution times.

As the SAAW size increases, the number of simulation cycles decreases and converges to a certain point, as shown in Figure 8.(a). The number cannot decrease further because rollback requires some number of additional simulation cycles even though clock advancements are enhanced by giving a larger window size.

In Figure 8.(b), as SAAW increases, rollback frequency rapidly increases. But, it decreases as the SAAW increases beyond a certain size. The reason is as follows. For a large SAAW, many events are processed in a simulation cycle, even though only the first advancement cycle of each simulation cycle may have rollback and the number of simulation cycles also decreases. At any rate, optimistic simulation with advancement windows has much higher rollback frequency than traditional optimistic simulation.

As shown in Figure 8.(c), parallelism increases, since more events are involved in event evaluation at each simulation cycle as SAAW size increases. But parallelism decreases finally, since the number of active gates will become smaller for a large SAAW. With a large window size, the gate is more

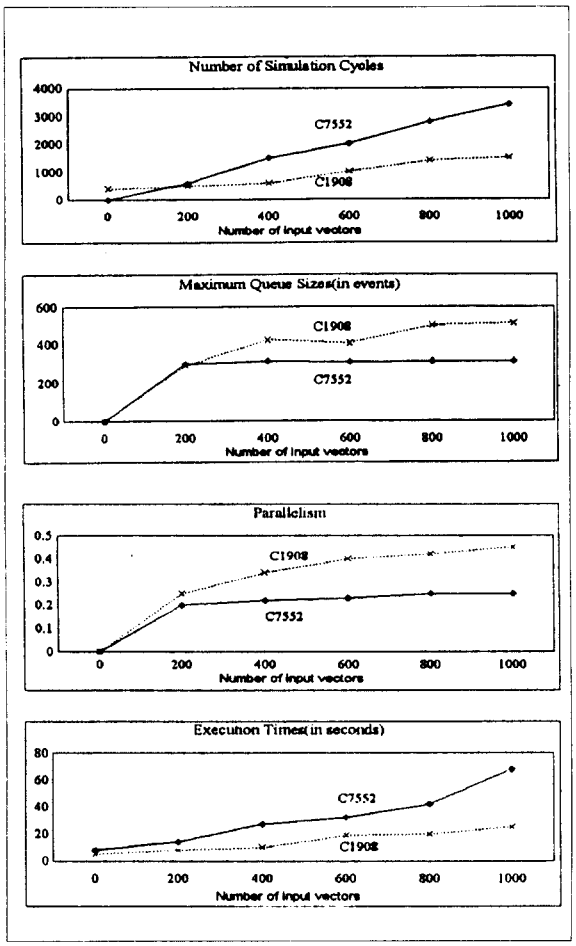

Figure 7. Conservative logic simulation with SCAW=4

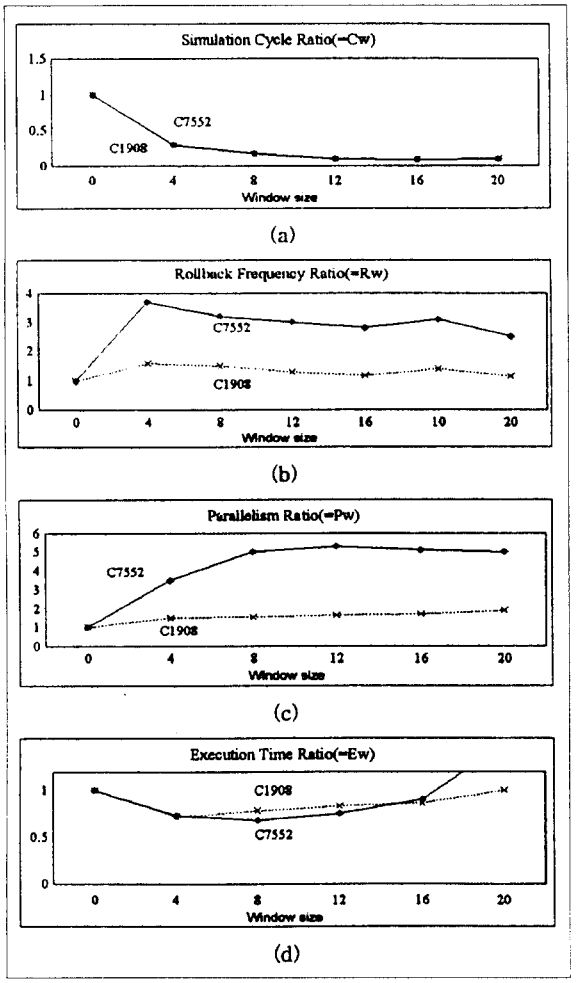

Figure 8. Advanced optimistic simulation with different SAAW(1)

likely to consume its entire queues, and therefore be idle on the next cycle. For example, in the worst case, if the SAAW size is equal to the number of input vectors, parallelism will be extremely small.

In Figure 8.(d), the execution time of optimistic simulation with the proposed technique decreases initially for the following reasons. First, the number of simulation cycles decreases and only one rollback manipulation is required in a simulation cycle even though SAAW size increases. Second, queue manipulations are fast on circular binary search event queues. Finally, rollback manipulation is very simple, therefore high rollback frequency does not affect the execution time that much. But the execution time finally increases as the SAAW size increases, since the number simulation cycle converges to a certain point and each simulation cycle has more advancement cycles.

Figure 9 shows the performance measured in adjusted parallelism, and average number of advancements per simulation cycle with 1,000 randomly generated input vectors. Adjusted parallelism increases until SAAW size reaches 10, but the adjusted parallelism decreases since the number of idle advancement cycles increases beyond that window size. The average number of advancements per simulation cycle increases.

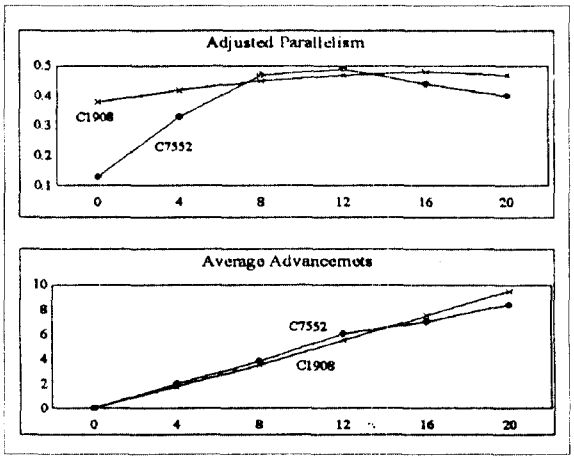

Figure 9. Advanced optimistic simulation with different SCAW sizes(2)

3.3 Communication Costs According to Message Length

The performance of logic simulation using advancement windows will be different depending on the architecture of the parallel machine used for the simulation. Simulation with the windows can be done more efficiently in MIMD than SIMD environments. In SIMD machines, the longest time among all gates for event evaluations and multi-event propagation dominates a simulation cycle time, since all processors are synchronized. In MIMD machines, however, each processor can perform event evaluation and propagate multi-events at its own pace without affecting other processors.

The big advantage of using multi-events is to reduce the communication costs by grouping multiple events into a multi-event. In parallel processing environments, each message between processors contains some additional information, such as packet header and tail.

Figure 10 shows the communication cost increase ratio on several parallel machines as message lengths increase, where the cost increases ratio is defined as the ratio of communication time for a 4n byte message to communication time for a 4 byte message [1,5,9,10]. In the figure, separate sending means that a single event is sent in a message, as in traditional

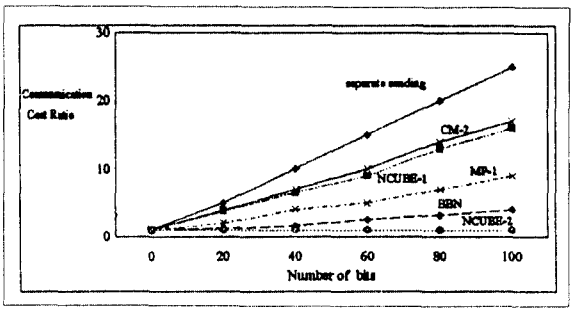

Figure 10. Communication costs according to message lengths

distributed event-driven simulation. In this case, communication cost is exactly obtained by multiplying the number of events sent and time per event. In the figure, the CM-2 and the MP-1 are massively parallel SIMD machines and the others are MIMD machines.

We can achieve good performance in logic simulation with advancement windows on MIMD machines for the following reasons. First, according to Figure 10, communication costs on modern parallel machines such as the NCUBE-2 can be significantly saved by sending long message rarely, rather than sending short messages frequently. Moreover, in SIMD processing environments, there might be many idle processes which have smaller advancement windows than the predetermined maximum advancement window due to synchronization of all processors. In contrast, in MIMD processing environments, there might not be many idle processes for smaller advancement windows.

4. Conclusions

We have proposed new efficient logic simulation approaches with advancement windows to enhance local clock advancement for fast simulation in parallel processing environments. According to experimental results for some combinational benchmark circuits on the CM-2, both conservative and optimistic logic simulation protocols with advancement windows and multi-events require fewer simulation cycles and achieve higher parallelism than traditional conservative and optimistic simulation protocols. In addition, execution times

also are smaller for some window sizes. Simulation with advancement windows is very promising when there are many input vectors, because average events per cycle will be almost at the maximum during steady state.

The good performance would be enhanced even more in MIMD environments than the SIMD environment of the CM-2. The technique can be applied to improve the performance of other simulation schemes, such as YADDES [11].

We can use the proposed technique to enhance local clock advancements, while a global moving time window can be simultaneously applied to prevent queue overflow. That is, advancement windows make local clocks go ahead as fast as possible, but any local clock too far beyond a global clock is not allowed to advance to prevent queue overflow.

As future studies, we aim to implement conservative and optimistic simulation approaches with advancement windows and multi-events on MIMD machines and to compare with the performance obtained on a SIMD machine. In addition, we are going to continue research on how we can efficiently use advancement windows and multi-events for circuits with feedback loops.

References

[1] BBN Advanced Computers Inc. BBN GP1000 Switch Tutorial, March 1989

[2] K. M. Chandy and Misra. Distributed simulation: A case study in design and verification of distributed programs. IEEE transactions on software Engineering, 5(5): 440-452, September 1979

[3] K. M. Chandy and Misra. Asynchronous distributed simulation via a sequence of parallel computations. Communications of the ACM, 24(11):198-206, April 1981

[4] K. M. Chandy and Misra. Parallel Program Design, A foundation. Adison Wesley, 1988

[5] E. M. Choi, M. J. Chung, Y. Chung. Comparisons and analysis of massively parallel SIMD architectures for parallel logic simulation. In Proceedings of the Sixth International Parallel Processing Symposium, March 1992.

[6] Y. Chung and M. J. Chung. Time Warp for efficient parallel logic simulation on a massively parallel SIMD machine. In Proceedings of the Tenth Annual International Phoenix Conference on Computers and Communications, pages 183-189. IEEE, March 1991.

[7] D. R. Jefferson. Virtual time. ACM Trans. Programming Languages and Systems, 7(3):404-425, July 1985.

[8] D. R. Jefferson et al. Implement of Time Warp on the Caltech Hypercube. In the Proceedings of the SCS Multiconference on Distributed Simulation, pages 70-75, January 1985.

[9] Y. Lan. Interprocessor Communication in Distributed Memory Multiprocessors. PhD thesis, Michigan State University, 1988.

[10] H. Xu, P. K. Mckinley, and L. M. Ni. Efficient implementation of barrier synchronization in wromhole routed hypercube multicomputers. Technical report, MSU-CPS-ACS-47, Department of Computer Science, Michigan State University, October 1991.

[11] M. Yu, S. Ghosh, and E. DeBenedicts. A non-deadlocking conservative asynchronous distributed discrete event simulation algorithm. In Proceedings of the SCS Multiconference on Advance in Parallel and Distributed Simulation, pages 39-43. ACM/IEEE/SCS, January 1991.

[12] M. J. Chung and Y. Chung. Data parallel simulation techniques for the Connection Machine. In Proceedings of the Supercomputing'90, pages 606-614. ACM/IEEE, November 1990.

[13] L. Soule and A. Gupta. Characterization of parallelism and deadlocks in distribute event simulation. In Advances in Parallel and Distributed Simulation, volume 23, pages 95-103, January 1991.

● 저자소개 ●

정연모
1980년 경북대학교 졸업 (학사)
1982년 KAIST 졸업 (공학석사)
1982년~1987년 경제기획원 전산처리관
1992년 미국 미시간주립대학교 졸업 (공학박사)
1992년~현재 경희대학교 전자공학과 조교수
관심분야 : VLSI설계 및 CAD, 병렬처리 시뮬레이션, 컴퓨터구조 등