Reducing Communication Overhead in Parallel Logic Simulation

Koichi Wada, Tetsuya Murakami and Yukari Hamada

May 23, 1996

ISE-TR-96-131

Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba, Ibaraki 305, JAPAN

This research was supported in part by TARA(Tsukuba Advanced Research Alliance) project, and by the Tien Lo Fund for International Academic Research, University of Tsukuba. This research was also supported by Grant-in-Aid for Scientific Research(C) from the Ministry of Education, Japan.

Reducing Communication Overhead in Parallel Logic Simulation

Koichi Wada, Tetsuya Murakami and Yukari Hamada

Institute of Information Sciences and Electronics,
University of Tsukuba,
Tsukuba, Ibaraki 305, JAPAN
TEL:+81-298-53-5512 FAX:+81-298-53-5206
e-mail: wada@is.tsukuba.ac.jp

Abstract
This paper presents a high performance parallel logic simulator on a PC(Personal Computer) cluster, which is based on a new communication algorithm called event clumping to reduce the number of communications. A logic circuit is partitioned into sub-circuits and assigned to parallel processors. The event clumping stores events to the buffers located at the outputs of the sub-circuits. Then, the events in the buffers are sorted by the addressee and packed into a message. The null message method is an algorithm to avoid deadlocks that can utilize inherent parallelism. However, it has a problem where a large number of null messages degrade a system performance. As well as the event clumping can reduce the number of communications, it can eliminate unnecessary null messages when it is applied to the null message method. In this paper, an implementation of the event clumping and a performance evaluation are described. The results of the evaluation shows that the event clumping can reduce the communication overhead to less then 2%, while the overhead by the conventional commnication is 40%. It is also described that the event clumping accelerate the simulation speed drastically since it can also improve the processor efficiency.

Keywords:
Parallel Logic Simulation, Event Clumping, Reducing Communication Overhead, Null Message, Performance Evaluation

1. Introduction

Logic simulation is a very important process for verifying correctness of logic circuits and their propagation delays in the design of LSI (Large Scale Integration). However, simulation of large scale circuits takes much time and costs. A high speed logic simulator is a requisite tool in designing recent VLSIs.

A parallel logic simulation is an attractive way to realize a high performance simulator[1,2,3]. However, it is not always easy to get an extensive performance, because:

(a) it is difficult to extract inherent parallelism in logic circuits fully while avoiding deadlocks efficiently.

To avoid deadlocks, several algorithms that use additional messages have been proposed so far[5,6]. For example, the null message method uses additional message to propagate logical time to succeeding processes. However, it yields a flood of messages among parallel processes. This results in serious performance degradation. Another Algorithm for solving deadlocks is known as a query computation. In this case, the process that issued a query has to wait for the reply before resuming gate evaluations. This delimits the parallelism.

(b) small grain size of events causes computation/communication imbalance.

In logic simulation, the computational size of gate evaluation is very small, and one or more events are possible to be generated by one evaluation. Because invoking communication is generally a time-consuming procedure, the above situation restricts the performance due to the imbalance between computations and communications.

Several studies have been done so far. The parallel logic simulator presented in [2], which is based on the time warp algorithm proposed by Jefferson[4], has been developed on a dedicated parallel machine. The reference[3] proposes an efficient algorithm for deadlock avoidance on a shared memory parallel computer. Recently, performance of a network of computer has been improved drastically. The parallel logic simulator on such a computer cluster will be an useful tool that has high-performance and good interface to current CAD systems.

This paper presents a high performance parallel logic simulator on a PC(Personal Computer) cluster, which is based on a new algorithm called event clumping to reduce the number of communications. The event clumping stores events to the buffers located at the outputs of the processes. Then, the events in the buffers are sorted by the addressee and packed into a message. As well as the event clumping can reduce the number of communications, it can eliminate unnecessary null messages that are used for deadlock avoidance.

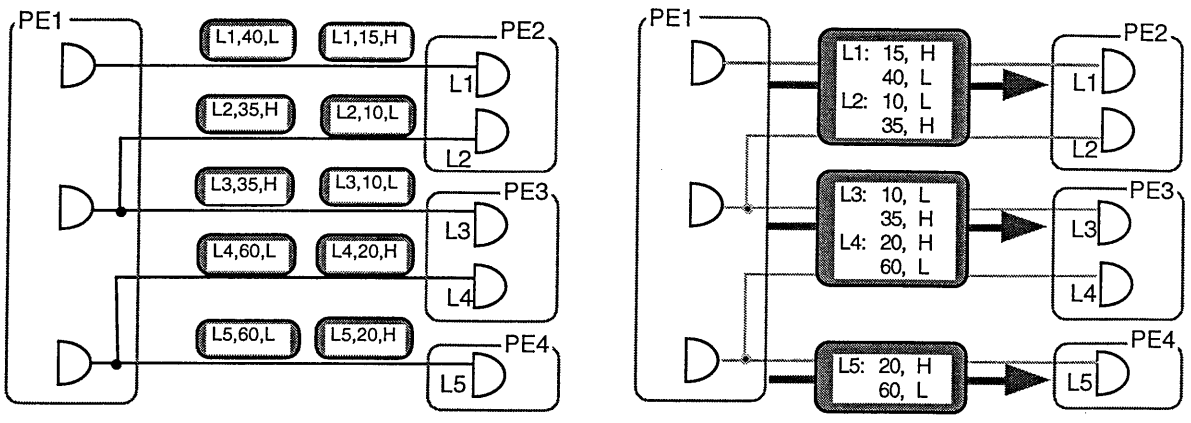
This paper is organized as follows: In section 2, brief overview of parallel logic simulation algorithms are shown. The event clumping method for reducing communication overhead is proposed in section 3. In Section 4, the implementation of the parallel logic simulator is described. Performance evaluation is shown in section 5. Section 6 concludes this paper.

2. Parallel Logic Simulation

In logic simulation, an event consists of the time and the logical value. Basically, in the sequential simulator, events are registered to a time wheel that manages logical time. In parallel logic simulation, the logical clocks are distributed to processors. Then, events are sent as messages among the processors.

Conservative method[5,6] is the one of the well-known algorithm for discrete event simulation that can be applied to logic simulation. In conservative algorithm, the oldest time-stamped event of all input events is evaluated first. This guarantees that events are evaluated and generated in order. However, it is possible to cause deadlocks since no events are evaluated before all input lines have received events.

A null message algorithm and a query computation algorithm have been proposed so far to avoid deadlock. In the null message algorithm, a message called a null message is



(a) Conventional event transmission.

(b) Event clumping.

Fig.1 Event messages.

issued whenever a logical clock is advanced. The null message guarantees that the output retains the previous value until the time contained in the null message.

In the query computation algorithm, when a deadlock is detected, a query message is issued from the input line that has no events. The query message is the message to inquire whether the clock can be advanced or not. The query message is composed of the time and the identifier of the sender gate. When a gate receives a query message, the time in the query message and the time of the receiver are compared. Then if the receiver is ahead of the sender, the receiver issues a reply message. The sender of the query can advance its clock to the time included in the reply message.

These algorithms guarantee the deadlocks are avoided or solved. However, they have some problems; The null message algorithm produces a flood of messages and restricts the performance. In query computation algorithm, parallelism might be reduced because processes have to be blocked until the reply is returned.

Our simulator is based on the null message algorithm for avoiding deadlocks since it can utilize inherent parallelism fully. In parallel processing, the number of communication is a primary factor that determines overall performance. This paper proposes an event clumping that can avoid a flood of null messages.

3. Event Clumping

In communication among parallel processors, there exist several overheads. For example, the data to be transferred have to be packed into message at a sender node. A receiver has to unpack the message to restore the data. There is also protocol overhead for ensuring reliable communication. Reducing these communication overheads is a crucial issue to utilize potential system performance.

Our simulator reduces communication overhead by decreasing the number of messages.

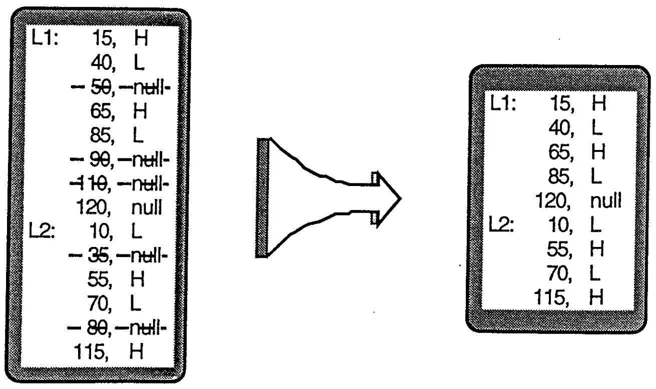


Fig. 2 Elimination of null events in a message buffer.

Fig. 1 shows a conventional event transmission and the proposed event clumping. A given logic circuit is divided into sub-circuits and assigned to processor elements(PEs). The outputs of the sub-circuit in PE1 are connected with the inputs of the sub-circuit in PE2, PE3, and PE4. From L1 to L5 denote line identifiers. The output event consists of the line identifier, the time-stamp, and the logical value. Fig. 1(a) shows a conventional event transmission where each message includes only one event. In this example, PE1 issues ten messages to transmit ten events. Fig. 1(b) shows an event clumping. When the output events are generated at PE1, events are sorted on its line identifier and the time-stamp. The sorted events are stored to the buffer located at the output line of the PE1. The events in the buffer are clumped to form a message and sent to the succeeding PEs, when one of the following condition is satisfied:

(1) When the total number of events in the buffers comes to the specific number.

(2) When the processor has no internal events to be evaluated.

In Fig. 1(b), ten events are clumped into three messages. The receivers unpack the message and place the events to the corresponding input lines.

Thus, the event clumping can reduce the number of messages drastically. However, it is not easy to determine the optimal number of events to be clumped. If the succeeding processors have no active gates, it would be profitable to send events as early as possible instead of storing to buffers. The optimal number of events depends on a performance balance of computation and communication. We experimentally determined the number of events to be clumped. In the current implementation, 1000 events are clumped into one message.

The event clumping is more effective when applying to the null message algorithm. A flood of null messages can be avoided since unnecessary null events are eliminated while being sorted at the message buffer. The null event may be appended only at the end of the clump. Fig. 2 depicts how the null events are eliminated in the message buffer. In the figure, five null events are deleted. For the line L1, all null messages except the last one are eliminated. The last null message should not be deleted since it may be useful to advance the clock of the receiver. Thus, by eliminating unnecessary null events, both the size of the message and the receiver's task in updating the clock can be reduced.

4. Organization of Parallel Logic Simulator

Our parallel logic simulator is running on a PC(Pentium 90MHz) cluster where the PCs are connected by ethernet. The simulator has been developed using C++ language and PVM(Parallel Virtual Machine)[7] running on the BSD/OS operating system. PVM is the software system that permits a network of computers to be used as a single parallel computer. Under PVM, collection of computers appears as one large distributed-memory computer called virtual machine.

A logic circuit at gate level (synchronous, asynchronous, and combination circuit) can be simulated. The signal value may be High, Low, and Unknown. The specific gate delays can be assigned to individual gates.

A logic circuit is partitioned into sub-circuits before simulation by following steps.

(1) Look for loops in a given circuit. The gates in the found loop are colored to form a group called a cluster. Repeat this step until no further loops is found.

(2) The next step is to make clusters of gates that are not included in the loops. Pick up an uncolored gate and trace the signal lines extend from the gate. If another uncolored gate is encountered, it is brought into the cluster. Repeat this step to make as big cluster as possible.

(3) These clusters are assigned to the processors with balancing the number of the gates. At this point, as far as the number of the gates is balanced, try to merge a loop cluster with the adjacent clusters.

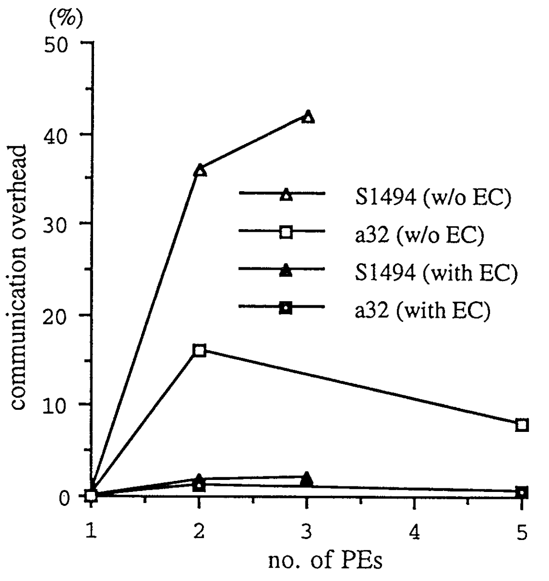


Fig.2 Communication overhead in the total execution time.

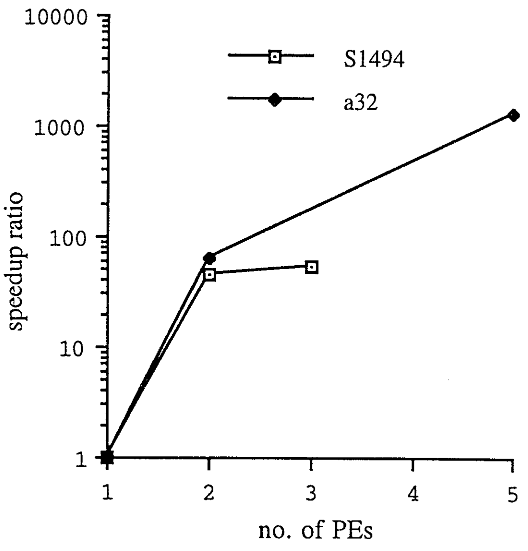


Fig.3 Performance gain by event clumping.

This partitioning algorithm is to make clusters of gates that have cyclic causal relations. The gates in the clusters tend to exchange event messages very frequently. By assigning those gates to one processor, the number of communication can be reduced.

5. Experimental Results

To evaluate the effectiveness of the event clumping, The 32-bits ALU(Arithmetic Logic Unit) called a32 and the sequential circuit S1494 proposed in ISCAS'89 has been simulated. The a32 and S1494 consist of 943 and 683 gates respectively. The 15000 input signals were generated randomly. The number of total events evaluated were approximately 1600k for a32 and 496k for S1494. The S1494 cannot be divided to more than three sub-circuits, since it includes a large cluster of gates. The simulation has been performed with event clumping and without event clumping for comparison.

Fig.2 shows a communication overhead in the total execution time. In the figure, EC denotes event clumping. Without event clumping, 10% to 40% of the execution time are consumed by communication. The event clumping can reduce this overhead to 1% to 2% for both circuits.

The performance gain obtained by event clumping is shown in Fig.3. In the case of a32, using five processors, the simulator with event clumping can execute about 1300 times faster than the simulator without event clumping. For S1494 on three processors, the event clumping can accelerate 55 times. Fig.3 shows that the performance gain increases as the number of processors increases. By clumping events, the sender processor can produce enough load to keep the receiver processor busy before falling into idle waiting for events to be evaluated. Thus, the event clumping reduces the processors' idle time and improves the processor efficiency. This results in the high speedup ratios shown in Fig.3.

6. Conclusions

In this paper, a high performance parallel logic simulator on a PC(Personal Computer) cluster was presented. In parallel processing, the number of communication is a primary factor that determines the overall performance. The simulator is based on a new algorithm called event clumping to reduce the number of communications.

To evaluate the effectiveness of the event clumping, the simulations were performed for combinatorial and sequential circuits. The results of the evaluation showed that the event clumping could reduce communication overhead to less than 2%, while overhead without the event clumping was 10% to 40%. It is also confirmed that the event clumping accelerated the simulation speed by a factor of 55 to 1300.

Our circuit partitioning algorithm finds loops in a circuit and assigns the loop to one processor. This partitioning algorithm is beneficial to reduce the number of communications. However, it sometimes has a problem in balancing the loads among processors since the scale of the loop depends on a topology of the circuit. The partitioning algorithm that performs better with event clumping is required to be developed. The detailed evaluation of our simulator is also necessary by applying to wider range of circuits.

References

[1] T. Murakami, K. Wada, and S. Okano,"Parallel Logic Simulator on a Workstation Cluster," Proc. of the IEEE Pacific Rim Conf., 1995, 268-271.

[2] Y. Matsumoto,"Applications of the time warp mechanism to LSI-CAD," Int. Symp. on Fifth Generation Computer Systems'94, 1994, 117-124.

[3] T. Kudou, T. Kimura, T. Terasawa, and H. Amano,"Parallel logic simulator for shared memory multiprocessors," CPSY 91-23, 1993, 151-158[in Japanese].

[4] D.R. Jefferson,"Virtual Time," ACM Trans. on Prog. Lang. and Sys., Vol.7, No.3, 1985, 404-425.

[5] J. Misra,"Distributed discrete-event simulation," Computing Surveys, Vol.18, No.1, 1986, 39-65.

[6] K.M. Chandy, J. Misra, and L.M. Hass,"Distributed deadlock detection," ACM Trans. on Computer Systems, Vol.1, No.2, 1983, 144-156.

[7] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam,"PVM 3 User's Guide and Reference Manual," 1993.