TR-711

Parallel Logic Simulator based on Time Warp and its Evaluation

by
Y. Matsumoto & K. Taki

November, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Parallel Logic Simulator based on Time Warp and its Evaluation

Yukinori Matsumoto and Kazuo Taki
Institute for New Generation Computer Technology

Abstract

This paper focuses on parallel logic simulation. An efficient logic simulator on a large-scale multiprocessor is targeted. The Time Warp mechanism, an optimistic method for time keeping, was experimented and evaluated.

Synchronous mechanisms and conservative mechanisms for time keeping have been already examined, and their inefficiency on large-scale distributed memory machines has been pointed out. On the other hand, there have been few reports on evaluation of the Time Warp mechanism although rollback processes have been presumed to be heavy. We aim at evaluating the efficiency of this mechanism. Several devices such as a local message scheduler, an antimessage reduction mechanism and a load distribution scheme are added in order to reduce rollback overhead.

The simulator is implemented on the Multi-PSI, a distributed memory multiprocessor. The simulator is written in concurrent logic language KL1. KL1 is expected to be suitable for parallel programming because it supports data-flow synchronization and global name space across the processor boundary.

In our experiment using 64 processors, 48-fold speedup and 99K events/sec performance was obtained. The result showed that the simulator has fairly good performance as a full-software logic simulator. Also we ascertained that rollback processes slightly affected performance.

1 Introduction

Logic simulation is one of the most important stages in LSI design. It is used in order to verify not only the logic design of circuits but also the timing of signal propagation in designed

circuits. Since the simulation phase consumes massive computation power, faster simulators are urgently needed. A parallel logic simulator is one likely way of producing quick simulation.

Parallel logic simulation is usually treated as a typical application of parallel discrete event simulation (PDES). In PDES, performance depends on the time keeping mechanism.

The mechanisms broadly fall into three categories: synchronous mechanism, conservative mechanisms and optimistic mechanisms. Since synchronous mechanisms require global synchronization, apparently they do not work efficiently on distributed memory multiprocessors[13]. On the other hand, the conservative mechanisms tend to deadlock when circuits have feedback loop paths. So, a lot of computation power is needed to avoid this[7, 9, 12]. On the contrary, optimistic mechanisms cannot deadlock, however, they do expend some computation power on rollback processes[4, 6]. Evaluation was done on only a SIMD machine[2] but has not been reported yet on MIMD machines.

We are targeting an efficient logic simulator on large-scale MIMD multiprocessors, most of which will be distributed memory machines. We adopted the Time Warp mechanism, an optimistic mechanism because overheads of the mechanism were considered to be reduced using some devices such as a local message scheduler, an antimessage reduction mechanism and a load distribution scheme. By adding them to the mechanism, we expected that it would become suitable for logic simulator on large-scale MIMD machines.

We have implemented a parallel logic simulator on the Multi-PSI — an experimental parallel inference machine[14], a distributed memory multiprocessor. Our simulator was written in concurrent logic language KL1. KL1 provides several advantages for quickly programming parallel applications. Data-flow synchronization, global name space and dynamic memory allocation are expected to remove the causes of lots of bugs.

Several benchmark circuits have been simulated on the simulator in order to evaluate the efficiency of the Time Warp mechanism. Performance, speedup, rollback overhead and inter-PE communication overhead have been measured.

This paper firstly overviews our system. Remarkable devices to enhance efficiency, such as a load distribution scheme, a local message scheduler and an antimessage reduction mechanism are mentioned. Secondly, KL1 and the Multi-PSI are briefly introduced. Then, fairly good performance and speedup of the system in actual execution are reported. Finally, we refer to the examination that revealed the main cause of decreasing performance in our simulator.

2 The Time Warp Mechanism

Event simulation can be modeled so that several objects change their states by communicating with each other. An object is a state-automaton. A message has information of an event whose occurrence time is stamped on the message (time-stamp).

Jefferson proposed the Virtual Time paradigm and its implementation, the Time Warp mechanism[6]. He suggested that the Time Warp mechanism would be useful as the time keeping mechanism for PDES.

In the Time Warp mechanism, each object usually acts according to received messages and also records the history of messages and states, assuming that messages arrive chronologically. But when a message arrives at an object out of time-stamp order, the object rewinds its history (this process is called rollback), and makes adjustments as if the message had arrived in correct time-stamp order. After rollback, ordinary computation is resumed. If there are messages which should not have been sent, the object also sends antimessages in order to cancel those messages.

A global control mechanism is also needed for memory management in the Time Warp mechanism. If all objects were to keep their full histories until execution termination, an enormous amount of memory would be needed and the Time Warp mechanism would be infeasible.

The global control mechanism works in order to get GVT (global virtual time). GVT must satisfy the following two conditions.

1. GVT is not greater than the minimum simulation time at any object.
2. GVT is not greater than the minimum time-stamp values in the messages that have been sent but not yet received.

After the global control mechanism updates GVT, it notifies all objects of the new GVT. As no objects rewind their histories before GVT, the memory area occupied by histories before GVT can be released.

3 System Overview

3.1 System Specification

The system simulates combinatorial circuits and sequential circuits that have feedback loops. It handles three values: Hi, Lo, and X (unknown). A different delay time can be assigned to

each gate (non-unit delay model). Since this simulator only treats gates, flip-flops and other functional blocks should be completely decomposed into gates.

The supported functions are the minimum set for experiments, but they can be easily expanded (e.g. to handle more signal values).

3.2 Load Distribution Scheme

For efficient execution of parallel logic simulation on a distributed memory machine, the scheme of load distribution is important at following three points: load balancing, keeping inter-PE communication frequency low and deriving a lot of parallelism.

In our simulator, target circuits are partitioned statically in the preprocessing phase. We propose a new partitioning strategy called "Cascading-Oriented Partitioning" (COP, for short) for high-quality load distribution.

COP makes several clusters by grouping gates that are connected to each other in a cascade form. A grouping operation starts from the primary input of the circuit. By tracing a path of the gate connection straightforward, subsequent gates are included in a cluster. If there are several candidates to be included, only one gate is selected and the others are left to be the starting points of other grouping operations.

After the partitioning process, small clusters that contain very few gates are merged into adjacent large clusters. Conversely, extremely long cascade-formed clusters are cut into several smaller clusters so that they do not cause load imbalancing.

Finally clusters are assigned to PEs randomly; the only constraint is that each processor should contain a roughly equal number of gates.

COP intends to exploit parallelism of the multiple fanouts. COP also guarantees that a gate has at least one adjacent gate in the same cluster. So, COP is effective in keeping the communication locality, that is, redicing inter-PE communication. The random distribution of clusters attains load balancing.

In COP, the smaller each cluster, the better load balancing but the higher inter-PE communication frequency. There is a performance trade-off between good load balancing and the low frequency of inter-PE communication. It is necessary to decide the appropriate size of a cluster according to the number of gates in the target circuit and the number of PEs. For the reference, clusters with 12 to 32 gates are generated for a circuit consisting of 12,000 gates in our simulation with 64 PEs.

3.3 Local Message Scheduler

In the simulation, there are usually several messages to be evaluated in a PE. When the Time Warp mechanism is used, the bigger time-stamp a message has, the more likely the message is to be rolled back. For this reason, message scheduling in each PE is expected to reduce rollback effectively.

In our system, a message scheduler resides in each PE. When a message is spawned, it is first registered in the scheduler in which the destination object belongs. The scheduler picks up the messages with the smallest time-stamp, and sends them to destination objects at the appropriate moment.

Owing to this scheduler, rollback never happens as long as an object is receiving messages from the other objects in the same PE. Only messages sent from other PEs may bring about rollback.

3.4 Reduction of Antimessages

In Jefferson's original Time Warp mechanism, when rollback occurs, as many antimessages must be generated as the number of messages that need be canceled (Figure 1). However, the number of antimessages can be reduced when we assume the next condition: for any objects A and B, messages sent from A to B arrive at B in the same order as they are sent by A (Order-preserved condition)[5].

Assume that $M_1, M_2, .., M_n$ are messages and AM is an antimessage. Also assume $M_1, M_2, .., M_n$ all satisfy the following three conditions:

- $M_1, M_2, .., M_n$ were sent before AM,
- $M_1, M_2, .., M_n$ were sent along the same channel that AM is sent along,
- $M_1, M_2, .., M_n$ have time-stamp values greater than or equal to AM.

Then it is clear that $M_1, M_2, .., M_n$ must be canceled. No other messages must be canceled. Only one antimessage that corresponds to the canceled message with the smallest time-stamp value need be sent (Figure 2).

We advanced this idea one step further. Assume that a sender has to cancel messages $M_1, M_2, .., M_n$ that are already sent in this order, and at the same time the sender knows that it will send a new message M_{new}, whose time stamp is equal to or less than that of M_1.

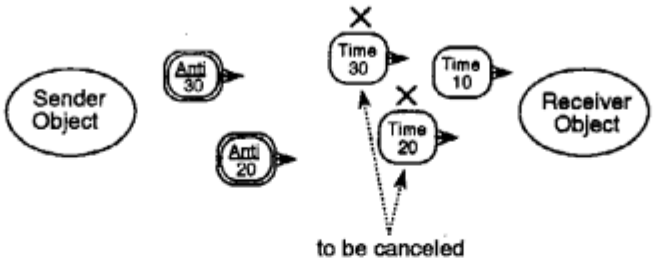

Figure 1: Cancellation with **several** antimessages

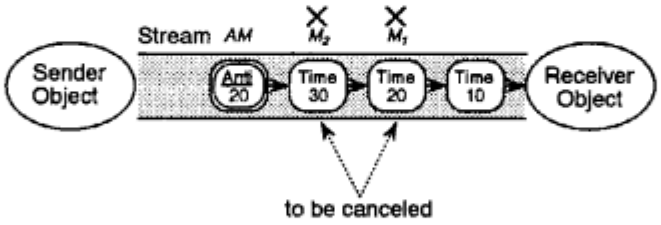

Figure 2: Cancellation with **one** antimessage

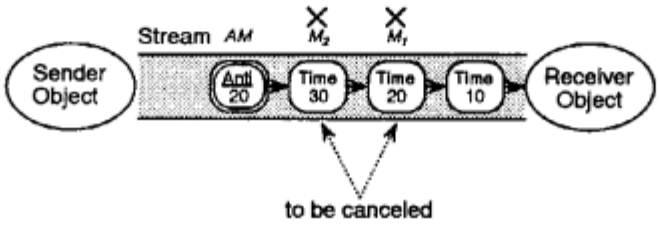

Figure 3: Cancellation with **no** antimessages

In this case, the sender sends M_{new} but no antimessage. When a receiver receives M_{new} with a smaller time-stamp than M_n that the destination object received just before, the receiver can easily notice that an invalid situation occurs, and can cancel $M_1, M_2, .., M_n$ immediately (Figure 3).

In our system, the message streams of KL1 are used for communication between objects. Since KL1 keeps the order of messages in the stream, the Order-preserved Condition is satisfied. So, we adopted the above optimization for reducing antimessages.

4 Hardware and Language

4.1 Hardware

This simulator is implemented on the Multi-PSI[14], a distributed memory MIMD machine. The Multi-PSI consists of 64 processing elements (PEs), which are connected to each other by a 2-dimensional mesh network. A PE is a 40-bit (8 bits for tag and 32 bits for data) CISC

processor controlled by the horizontal micro-instruction. The cycle time is 200 nsec.

A network controller is paired with each PE, supporting message passing communication between PEs. The bandwidth of the controller is 5M bytes/sec. The network has wormhole routing functionality.

Since the Multi-PSI is a distributed memory machine, communication latency between objects in different PEs takes approximately twenty times more than the latency in the same PE. However, the distributed memory architecture is easy to scale up.

4.2 Language and Implementation

This simulator is written in concurrent logic language KL1.

KL1 is a language without destructive value assignment to variables, that is a single assignment language. Due to its nature, data-flow synchronization is realized without significant overheads in the language implementation. Needlessness of explicit description of synchronization eliminates the causes of lots of bugs.

On the other hand, a single assignment language tends to consume storage rapidly. Dynamic memory allocation mechanism and garbage collection mechanisms are supported in the KL1 implementation. So, programmers are free from writing memory management.

The KL1 language assumes a system-wide (global) name space even on a distributed memory machine. In KL1 programming, first, a programmer writes only the logical concurrency, relations among concurrent objects or data-flow. Then he adds the "pragma" to specify an object allocation to a certain processor, as below.

$$\ldots, B@processor(PE), \ldots$$

where B is a "goal" of KL1, which represents an object. Inter-PE reference pointers among objects or variables are maintained automatically by the KL1 language system at run time. So, a programmer need not worry at all about the programming of inter-processor communication. It also reduces the cause of bugs.

Since these characteristics described above are very beneficial for parallel programming, KL1 programming is much easier than using other conventional languages (e.g. FORTRAN and C). In fact, it took one person just three months from the start of designing to the completion of the primary version of the simulator, including the circuit partitioning module! If it had been programmed in another conventional language, it would have taken at least three times as long.

Table 1: Target circuits

Circuits	s1494	s5378	s9234	s13207
No. of gates	683	3,853	6,965	11,965

4.3 Avoiding Asynchronous Copying GC

As mentioned above, garbage collection (GC) is indispensable for KL1. Two kinds of garbage collection (GC) mechanism, a copying GC[1] and the MRB GC[3], are implemented for intra-PE memory management on the Multi-PSI.

For the Time Warp mechanism, the most important point of obtaining good performance is to keep the pace of simulation in each PE equally. However, the copying GC starts at unpredictable time in each PE. It must disturb the pace of simulation.

Fortunately, since the MRB GC collects the data area of single reference without stopping KL1 execution, it is expected to undisturb the pace of simulation.

We took great care to keep the data reference single so that all data areas can be collected by the MRB GC. Thus we succeeded in preventing the copying GC from working. Consequently, when a certain circuit was simulated using 64 PEs, an improvement in performance of approximately 37% was attained compared to the case where copying GC occurred(see appendix).

5 Measurements and Discussions

We executed several experimental simulations on the Multi-PSI. Four sequential circuits, presented in ISCAS'89, were simulated in our experiments. The number of gates in these circuits is shown in Table 1.

Input vectors were generated at random except clock lines. The clock interval was 40 units of time, and simulation was done for 10,000 units of time.

We simulated each circuit 5 times for each execution. We measured the system performance, speedup, inter-PE communication overhead and rollback overhead. Their average values are reported in the following sections. The parallelism of each problem is also measured and discussed.

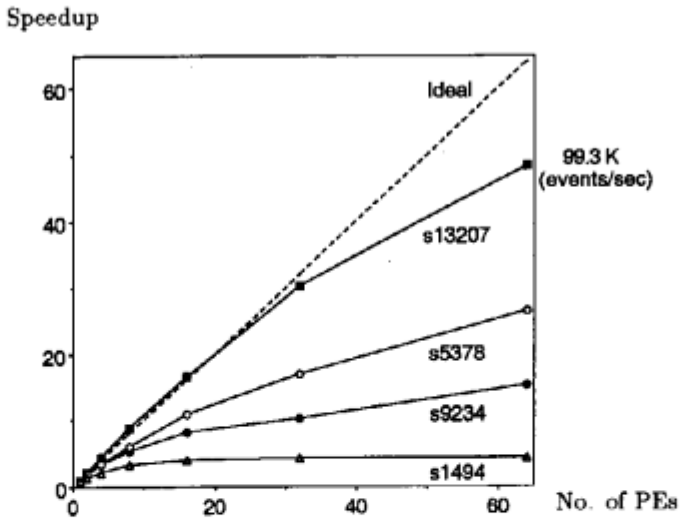

Figure 4: Speedup

Table 2: The frequency of inter-PE communication (64PEs)

Circuits	s1494	s5378	s9234	s13207
f_c	0.402	0.121	0.0846	0.0222

5.1 Performance and Speedup

Figure 4 shows the system performance when the circuits were simulated using various numbers of PEs. Additionally, performance in the best case is also shown there.

Our system showed good speedup and performance in certain cases like s13207 and s5378. In the best case, very good speedup of 48-fold was attained using 64 PEs. Besides, approximately 99K events/sec performance, fairly good for a full-software logic simulator, was attained. On the other hand, comparatively poor performance and speedup were measured in the cases of s9234 and s1494.

5.2 Inter-PE Communication Overhead

We measured f_c, the frequency of inter-PE communication, shown in Table 2. f_c is defined below.

$$f_c = M_c/M_{all} \tag{1}$$

where M_c is the number of messages that crossed the processor boundaries and M_{all} is the total number of messages generated in the simulation.

Table 3: The frequency and the depth of rollback (64PEs)

Circuits	s1494	s5378	s9234	s13207
f_r	0.972	0.0793	0.0571	0.0243
d_r	2.77	3.91	11.2	7.96

We also measured the cost for inter-PE communication per message. A message is 25-byte data. The cost per message for inter-PE communication was 0.503 msec.

The cost C_c can be decomposed to three factors as follows.

$$C_c = C_{p_s} + C_l + C_{p_r} \tag{2}$$

where C_{p_s} is the time consumed in the sender PE for composing a message packet, C_l is the time from when the message leaves the sender till it arrives at the receiver (latency), and C_{p_r} is the time taken by the receiver to decompose the message packet. On the Multi-PSI, C_{p_s} and C_{p_r} are approximately the same[11], however, the latency between the most distant PEs is only 0.011 msec. This is negligible compared to the other costs C_{p_s} and C_{p_r}.

For the case of s13207, using 64 PEs, f_c was very small such as 0.0222. This means that our load distribution scheme worked efficiently. We can certainly say that the inter-PE communication cost had only a slight effect on system performance in this case. On the contrary, in the case of s1494, the value of f_c was 0.402, such a value can not be ignored. In this case, the inter-PE communication is considered to adversely affect system performance.

Since the average cost of the essential work, that is, evaluating and scheduling a message, is 0.362 msec, the inter-PE communication is not negligible. However, the relative cost of the inter-PE communication is far lower than such systems where inter-PE communication is supported by the operating system.

5.3 Rollback Overhead

We measured the frequency of rollback, f_r, and the average depth of rewound history, d_r. f_r and d_r are defined as follows.

$$f_r = R/E \tag{3}$$

$$d_r = H_r/R \tag{4}$$

where R is the number of rollback occurrences, and E is the number of actual events, that corresponds to the messages which were not rolled back. H_r is the number of messages that

Table 4: Modified speedup and parallelism

Circuits	s1494	s5378	s9234	s13207
Speedup (modified)	3.96	23.62	12.51	35.66
Parallelism	18.88	35.52	17.95	43.24

were rolled back. Table 3 shows the frequency of rollback and average depth of rewound history for each problem.

The cost per rollback also attracts our interest. As the computation time for rollback, t_r, is roughly proportional to the number of messages rewound, t_r is represented by the next equation.

$$t_r = k_r h_r + C_r \tag{5}$$

where h_r is the number of messages rewound in the history and C_r is a constant. Our measurements revealed that k_r was 0.0241 msec and C_r was 0.256 msec.

For the case of s13207, where h_r in the equation (5) was substituted by d_r in Table 3, the average cost for rollback amounts to 0.448 msec. Since the time for essential processes is 0.362 msec, the rollback cost is rather time consuming. However, in this case, the rollback frequency was only 0.0243. So, total rollback cost is not considered to have been seriously high.

5.4 Parallelism

Parallelism suggests the upper limit of speedup. In practice, however, the actual speedup is far different from parallelism because of several overheads such as inter-PE communication and rollback.

The cost of releasing an unnecessary history area is one of overheads that can not be ignored. The cost causes a super-linear speedup[8]. For the purpose of making the effect of inter-PE communication overhead and rollback overhead clear, we measured the cost of releasing an unnecessary history area and removed the effect from the actual speedup in the case of 64PEs.

On the other hand, parallelism of each problem was estimated, as below. We made another simulator for the measurement of parallelism. In that simulator, all PEs work according to the global synchronization. Here, we call an interval between global synchronizations a "time slot". A PE evaluates only one message in a time slot. All the output messages, if any, are also sent and registered to their destination schedulers within the time slot. When there is no message to be evaluated in a PE at a certain time slot, the PE simply idles. Assume that the simulation

finishes after N synchronization is done, and M actual events, which are the messages that are not rolled back, are measured in the simulation. Here, we define the parallelism of the problem as M/N. The parallelism means the speedup in such an environment where the cost for non-essential processes, such as rollback and inter-PE communication, can be ignored. We measured the parallelism of each problem using this simulator .

Table 4 shows modified speedup and the parallelism of each problem using 64 PEs. It turns out that the modified speedup highly correlates with the parallelism. Since the parallelism is the speedup when both inter-PE communication overhead and rollback overhead can be ignored, the gap between them is considered to have been caused by these overheads. In particular, the gap is wide in the case of s1494, where both inter-PE communication frequency and rollback frequency were extremely high.

6 Further Experiments

As either inter-PE communication cost or rollback cost are not negligibly small, both of them are considered to affect performance adversely. However, it is difficult to separate their influence clearly.

In this section, we report on the experiments that aim at clarifying which affects performance more dominantly, the inter-PE communication cost or the rollback cost. We assumed a model described below and made a system for the experiments.

6.1 Model

We assume that all the processes that need costs are rollback, inter-PE communication and an essential process. Here, an essential process consists of a message evaluation work and a scheduling work. Any other processes, such as GVT updating and releasing unnecessary history area, do not need any costs at all. We also assume that essential process cost is equal for any gates.

In our model, the inter-PE communication cost is represented by the equation (2), and the rollback cost by the equation (5).

6.2 Experimental System

The experimental system is based on the simulator presented in the previous sections. By adding several dummy loads to the original simulator, the actual costs for rollback and inter-PE communication become negligible. Thus this system keeps the fidelity to the model as highly as possible.

In the system, the cost for an essential process is fixed to be heavy, whereas rollback cost and inter-PE communication cost are changeable.

6.3 Results

We performed two kinds of comparative examination, as below.

1. Inter-PE communication cost is fixed to keep the relative cost to an essential process the same as in the actual simulation. With respect to rollback, k_r and C_r in the equation (5) are varied but keeping C_r/k_r equal to that in the actual simulation.

2. Rollback cost is fixed to keep the relative cost to an essential process the same as in the actual simulation. Inter-PE communication cost is varied but keeping the equality between C_{p_s} and C_{p_r} in the equation (2) because they were approximately equal in the actual simulation.

We simulated s9234 and s1494. They have approximately the same parallelism, whereas both inter-PE communication frequency and the rollback frequency were much different between them.

Figure 5 shows the results. X axis shows the relative value of $C_{p_s} + C_{p_r}$ and C_r compared to the essential cost respectively. Y axis shows the relative performance against the performance when both inter-PE communication cost and rollback cost are set to zero. The arrows indicate the the actual proportion points between these costs.

For both cases, the higher cost for inter-PE communication, the worse performance. It terned out that inter-PE communication cost affected performance adversely. The difference between the decline curves was apparently caused by the difference in inter-PE communication frequency.

On the contrary, performance did not decrease in both cases, even when the cost of rollback increased. It means that rollback cost did not affect performance dominantly in our system.

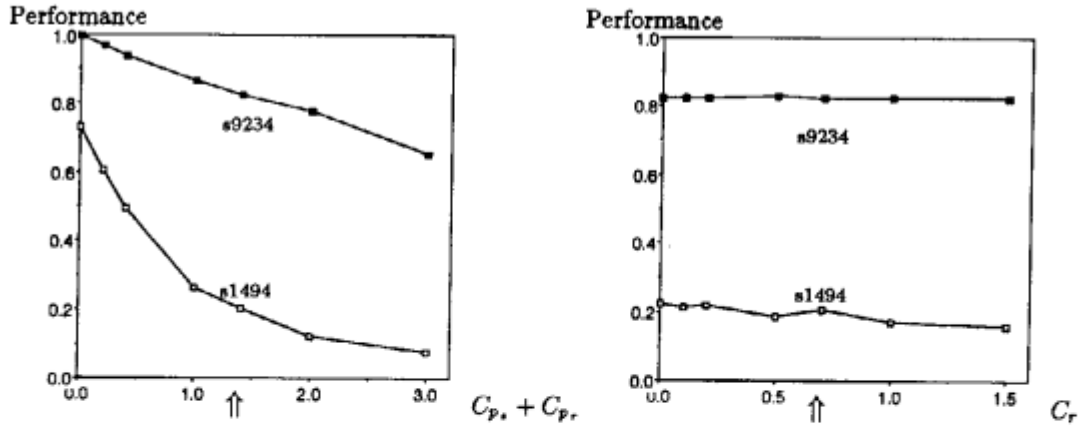

Figure 5: Factors of affecting performance

7 Summary and Conclusion

We constructed a parallel logic simulator on the Multi-PSI, a distributed memory multiprocessor. The simulator was programmed using concurrent logic language KL1. Since the causes of bugs are reduced in KL1, it enabled to program the simulator in only three months by one person.

The Time Warp mechanism was adopted for time keeping in the simulator. Since rollback overhead in a naive Time Warp mechanism was considered heavy, we added several devices such as a local message scheduler, an antimessage reduction mechanism and a load distribution scheme to reduce the overhead.

Several benchmark circuits were simulated on our system. The simulator attained good performance and speedup: about 99K events/sec and 48-fold speedup using 64 PEs. The performance obtained here is fairly good for a software logic simulator. This experiments showed that the Time Warp mechanism worked efficiently in the simulator.

We also examined the factors that are considered to affect performance adversely. The experiments revealed that rollback overhead did not affect performance seriously in our system, while inter-PE communication overhead decrease performance.

References

[1] Baker, H. G. : List Processing in Real Time on a Serial Computer, *Communications of the ACM*, pp. 280–294, (1978)

[2] Chung, M.J. and Chung, Y. : Data Parallel Simulation using Time-Warp on the Connection Machine, *26th ACM/IEEE Design Automation Conference*, pp. 98–103, (1989)

[3] Chikayama, T. and Kimura, Y. : Multiple Reference Management in Flat GHC, *Fourth International Conference on Logic Programming*, pp. 276–293, (1987)

[4] Fujimoto, R. M. : Parallel Discrete Event Simulation, *Communications of the ACM*, Vol.33, No.10, pp. 30–53 (1990)

[5] Fukui, S. : Improvement of the Virtual Time Algorithm, *Transactions of Information Processing Society of Japan*, Vol.30, No.12, pp. 1547–1554 (1989) in Japanese

[6] Jefferson, D. R. : Virtual Time, *ACM Transactions on Programming Languages and Systems*, Vol.7, No.3, pp. 404–425 (1985)

[7] Lubachevsky, B. D. : Efficient Distributed Event-Driven Simulations of Multiple-Loop Networks, *Communications of the ACM*, Vol.32, No.1, pp. 111–131 (1989)

[8] Matsumoto, Y. and Taki, K. : Parallel Logic Simulation based on Virtual Time, *Proceedings of Joint Symposium on Parallel Processing '91*, pp. 365–372 (1991) in Japanese

[9] Misra, J. : Distributed Discrete-Event Simulation, *ACM Computing Surveys*, Vol.18, No.1, pp. 39–64 (1986)

[10] Nakajima, K. et al. : Distributed Implementation of KL1 on the Multi-PSI/V2, *Proceedings of 1989 International Conference on Logic Programming* pp. 436–451, (1989)

[11] Nakajima, K and Ichiyoshi, N. : Evaluation of Inter-processor Communication in the KL1 Implementation on the Multi-PSI, *ICOT Technical Report*, TR-531, (1990)

[12] Shimogori, S. and Kage, T. : Parallel Logic Simulation using A Message-Driven Approach, *IEICE Technical Report*, CAS88-110, pp. 23–30 (1989) in Japanese

[13] Soulé, L. and Blank, T. : Parallel Logic Simulation on General Purpose Machines, *25th ACM/IEEE Design Automation Conference*, pp. 166–170 (1988)

[14] Taki, K. : The parallel software research and development tool: Multi-PSI system, *Programming of Future Generation Computers*, North-Holland, pp. 411–426 (1988)

[15] Ueda, K. and Chikayama, T. : Design of the Kernel Language for the Parallel Inference Machine, *The Computer Journal*, Vol.33, No.6, pp. 494-500 (1990)

Appendix

For the purpose of ascertaining the influence of asynchronous copying GC, we made another system and compared it to the original one. The difference between the system and the original one is that copying GC sometimes happens in the new system, whereas only the incremental GC works in the original one.

Original system: Only the MRB GC works for collecting garbages.

Comparative system: The copying GC happens asynchronously in each PE.

Table 5 compares them when s13207 was simulated using 64 PEs. The result shows that asynchronous outbreaks of copying GC in each PE increase both rollback frequency and the amount of rewound history. It certainly caused the poor performance of the simulator.

Table 5: Influence of asynchronous occurrence of the copying GC

	Performance (K events/sec)	Frequency of rollback	Depth of rollback
Original system	99.299	0.0243	7.96
Comparative system	72.895	0.0261	11.684