# Parallel Logic Simulator

## ABSTRACT

Logic simulation, one of the most time-consuming stages in LSI design, is used to verify the logical and timing specifications of designed circuits. We built a high-performance parallel logic simulator capable of deriving a lot of parallelism from the target circuits so as to exploit the entire potential of large-scale MIMD machines, such as the PIM machine.

## KEY FEATURES

### Time Warp mechanism (TW)

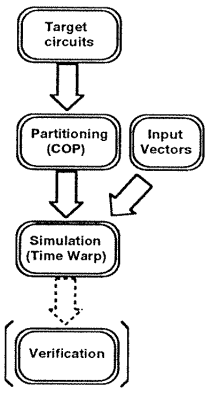
TW asynchronously controls the order in which messages should be evaluated. It tries to exploit complete parallelism with speculative computation, while the rollback process cancels speculation errors.

### Reduction of rollback overhead

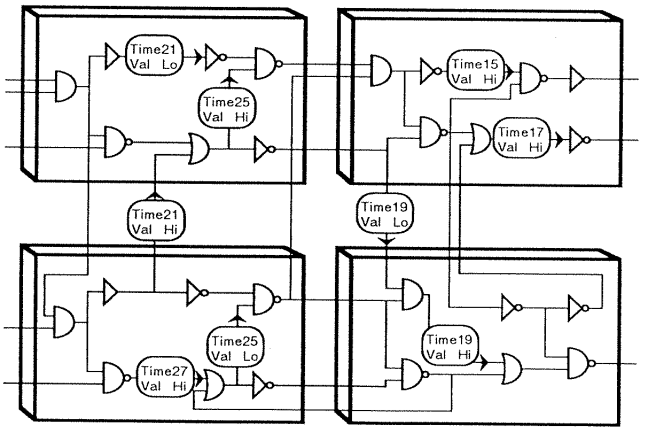
The antimessage reduction mechanism, adequate message scheduling, and the Adaptively Moving Time-Ceiling are used to reduce the cost and frequency of rollback.

### Cascading-Oriented Partitioning (COP)

COP provides high-quality solutions for circuit partitioning, achieving low inter-PE communication, high parallelism extraction and load balancing.




Process Flow




Parallel Logic Simulation

### Background

Logic simulators are used in order to verify the logical and timing specification of designed circuits. Since logic simulation is one of the most time-consuming stages in LSI design, faster simulators are required. In addition, flexibility is also needed.

A parallel logic simulator is one likely way of producing quick and flexible simulation. We built a parallel logic simulator on the PIM machine, as a first but significant step to the realization of super high performance simulators on future hyper-parallel machines.

### Specification of the Simulator

The simulator 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 treats gates only, flip-flops and other functional blocks should be completely decomposed into gates.

### Parallel Discrete Event Simulation and Time Warp

Parallel logic simulation is treated as a typical application of parallel discrete event simulation (PDES). PDES can be modeled so that several objects (corresponding to gates) change their states by communicating with each other. A message has information of an event whose occurrence time is stamped on the message (time-stamp). Since messages should be received and evaluated in the time-stamp order by their destination objects, a time-keeping mechanism is needed.

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

The rollback process has been suspected to contain large overheads which might affect performance adversely. So, we added several devices to reduce the overheads so that the simulator can run efficiently on the PIM machine.

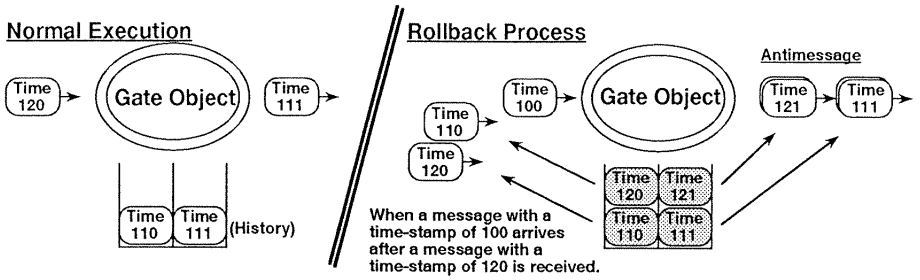



Figure 1: Time Warp mechanism

### Message Scheduling

During simulation, there are usually several messages to be evaluated in a PE, and therefore some scheduling strategy is needed. In our strategy, the message with the smallest time-stamp is evaluated first. This strategy is considered to be adequate for TW because the bigger time-stamp a message has, the more likely the message is to be rolled back.

### Antimessage Reduction

As long as messages are sent through the KL1 stream, the messages arrive at the receiver in the same order as they were transmitted. In this environment, subsequent antimessages can be reduced. We adopted this optimization technique, expecting that it would reduce the rollback cost.

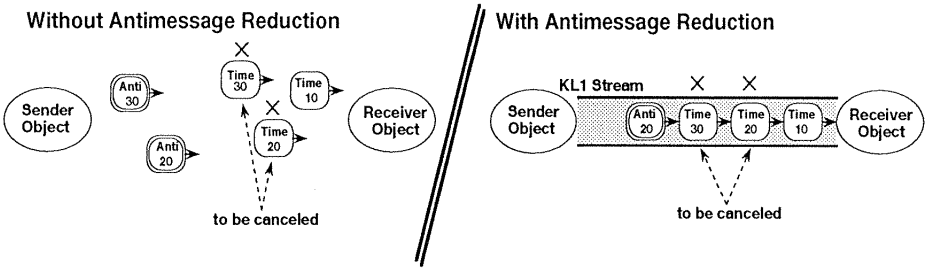



Figure 2: Antimessage Reduction

### Adaptively Moving Time-Ceiling

In a naively implemented TW, a message is evaluated as soon as it arrives at its destination gate, even if its time-stamp is extremely large. Such a message, however, will probably be rolled back later and thus the evaluation may cause other side-effects.

Moving Time-Ceiling (MTC) gives the upper limit of time for suppressing both the evaluation and generation of messages with extremely large time-stamps. MTC contributes to reducing the frequency of rollback.

MTC is sometimes updated into the future. However, it is quite difficult to statically determine the optimal interval between the old MTC position and the new. If the interval is too short, many processors might idle although the rollback frequency would be greatly reduced. Conversely, if MTC jumps to the distant future, the rollback frequency will hardly be reduced at all.

In our simulator, the MTC interval is adaptively determined according to the rollback frequency and the effect of the previous intervals on performance (Adaptively Moving Time-Ceiling). AMTC can exploit the entire potential of parallel machines without depending on target circuits or simulation conditions.

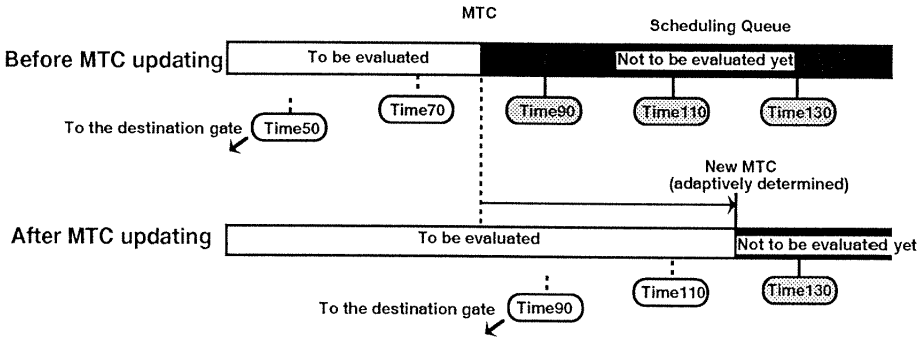



Figure 3: Adaptively Moving Time-Ceiling

### Circuit Partitioning

In our simulator, the "Cascading-Oriented Partitioning" strategy is used for circuit partitioning to attain high-quality load distribution.

Firstly, COP makes several clusters by grouping gates that are connected to each other into a cascade-form. Then, the clusters are cut or merged to adjust their sizes. Finally, the clusters are assigned to processors at random.

This scheme provides adequate partitioning solutions that satisfy these three requirements: load balancing, keeping inter-PE communication frequency low and extracting a lot of parallelism.

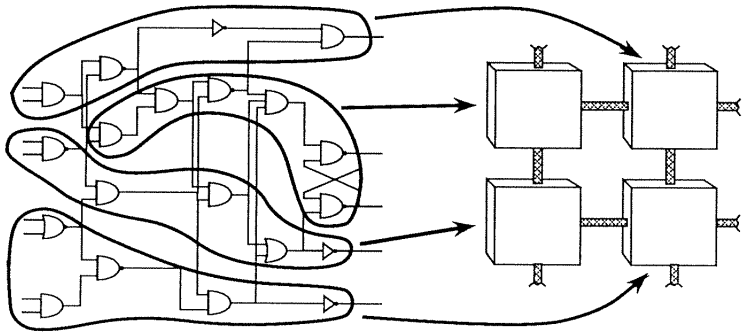


Figure 4: Cascading-Oriented Partitioning

### Performance Measurement

We simulated five sequential circuits on the PIM/m machine. These were benchmark data for ISCAS'89. Table 1 shows the size (# of gates) of the target circuits. Figure 5 indicates the performance of the simulator.

| Circuits | s38584 | s38417 | s35932 | s15850 | s13207 |
|---|---|---|---|---|---|
| # of gates | 27,965 | 31,995 | 26,433 | 13,354 | 11,965 |

Table 1: Size of the circuits

In the best case, 537K events/sec performance and 166-fold speedup were attained using 256 processors.

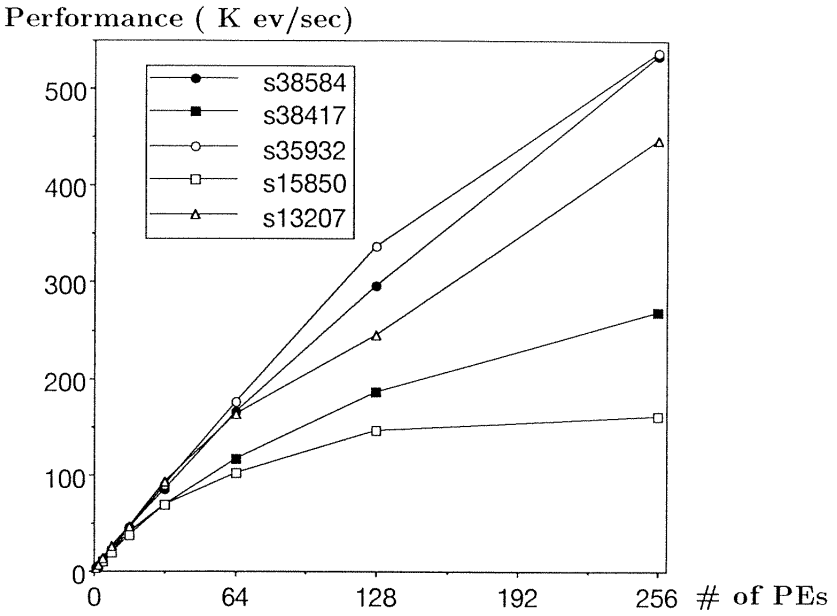



Figure 5: Performance vs. # of Processors

### Outline of Demonstration

In the demonstration, two sequential circuits, s15850 and s38584, will be simulated.

The demonstration will focus on these three points.

- The utilization of processors.
- The adequacy of the intervals in AMTC.
- Total performance and rollback frequency.