Effective Simulation for The Giga-scale Massively Parallel Supercomputer SR2201

Kaoru Suzuki

Software Technology Development Center
Genral Purpose Computer Division, Hitachi, Ltd.
1 Horiyamashita, Hadano-shi, Kanagawa-ken, 259-13 Japan
e-mail: kasuzuki@kanagawa.hitachi.co.jp

Shunsuke Miyamoto

Strategic Products Development Center
Information Systems Division, Hitachi, Ltd.
292 Yoshida-cho, Totsuka-ku, Yokohama, 244 Japan
e-mail: s-miyamo@system.hitachi.co.jp

Masato Kurosaki

Design Automation Development Department
Genral Purpose Computer Division, Hitachi, Ltd.
1 Horiyamashita, Hadano-shi, Kanagawa-ken, 259-13 Japan
e-mail: mkurosa@kanagawa.hitachi.co.jp

Junji Nakagoshi

RISC Development Department
Genral Purpose Computer Division, Hitachi, Ltd.
1 Horiyamashita, Hadano-shi, Kanagawa-ken, 259-13 Japan
e-mail: jnakago@kanagawa.hitachi.co.jp

Abstract - A high performance parallel network simulation environment was developed in the SR2201 project. The SR2201 is one of the highest performance massively parallel supercomputers in the world. The enhanced simulation algorithm achieved a 2.4 times increase in simulation speed compared with conventional simulation methodology. A 98% detection rate for all design errors before physical design contributed to the shortening of development time.

I. Introduction

Design verification strategy is one of the most important issues in making the high quality design possible, especially in mega-gate systems. There are three significant strategic issues that affect the quality of the design of a target system.

These issues are:

(1) The usage of different kind of logic simulators

(2) Design abstraction models

(3) Well qualified test data

Among these, the selection of a logic simulator is the key factor in assuring a high quality of design. High speed and large scale microprocessor design projects usually involve many kinds of logic simulation and emulation systems, with different levels of design description [1] - [2].

The usage of several simulators in a verification process has led to performance deterioration and memory capacity shortages in current simulator implementation. If a simulator has good performance and memory capacity, design result is verified simply using one level abstraction of the design description. However, recent innovation of process technology demands greater logic simulator performance and capacity, especially in high-end microprocessor design, which handles at least one million transistors. The design process of supercomputers and mainframe computers involves the same problem. One solution is to introduce a special purpose logic simulation machine to achieve high performance simulation. We have developed the Vectorized Processing System for Logic Verification (VELVET) using our supercomputer with newly developed vector simulation instructions[3]. The VELVET system usually processes a supercomputer design of several tens of mega-gates using a clock event suppression algorithm and a simulation control language with an interface for using real test-and-maintenance programs. In many former projects, only main storage and cache RAMs were implemented with high level simulation control language. Now, designers are able to verify their intentions without any design abstraction; in other words, there is no reduction in detailed design information.

Test data for design verification is another significant issue. The white box test supported by designers is very efficient for design quality assurance. Usage of test and maintenance programs with randomly generated instructions in the design verification stage also helps to improve verification quality.

In the SR2201 project, the designers suggested a giga-gate scale of integration in one system, instead of several tens of mega-gates. The challenge of huge-scale integration simulation has been met by using a simulation model with a high level of abstraction. Therefore, the objective of the design verification team here in the SR2201 project is to develop a high speed system simulator to achieve a high quality of design in the giga-gate system, using a minimum and proper simulation abstraction model.

In the following section, an overview is given of the simulation target, SR2201. In section 3, our simulation system and simulation models are characterized. Figures which show the resulting verification are included in section 4. We state our conclusion in section 5.

II. Outline of Massively Parallel Processor System SR2201

Performance evaluation through benchmarking proved SR2201 to be the fastest massively parallel machine in the world. This section outlines the complexity of this machine.

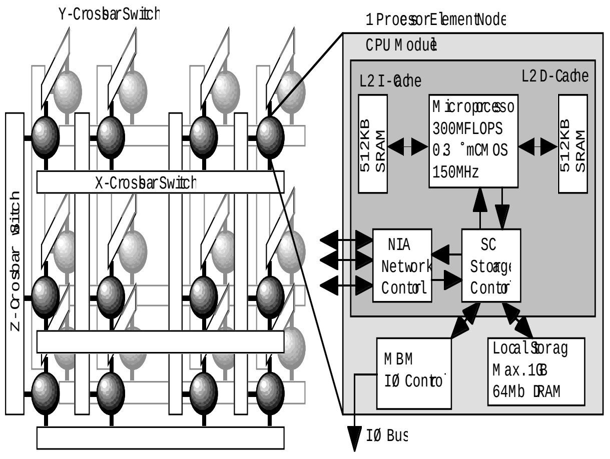
The system consists of three subsystems: a processor element subsystem, a network subsystem (among the processor elements) and an input/output subsystem. The heart of the SR2201 is the processor element subsystem, denoted in Figure 1(a) by black spheres. It includes an originally developed CMOS RISC microprocessor, two level 2 caches which each hold 512K bytes of instructions or data, a network interface adapter to transfer data to and from the network subsystem, a storage controller, which controls high-speed data access to local storage of up to 1GB, and an input/output controller. These elements, with the exception of the input/output controller are clustered on one ceramic module, with eight processor-element modules on one package. The RISC processor runs on a 150 MHz clock, and achieves 0.3 GFLOPS peak performance. It contains multiple 16K byte primary caches for storing instructions and data, and has a pseudo vector processing facility[4]. Maximum capacity of the system is 2,048 processor elements. Figure 1(b) shows block diagram of one processor element.

The second major subsystem is a network subsystem of the two-dimensional or three-dimensional crossbar type. Three-dimensional crossbar networks are used for large- scale systems to meet the needs of elaborate scientific applications. The performance of data transmission among processor elements is 300M bytes per second[5]. Figure 1(a) shows a schematic diagram of a three-dimensional crossbar network among processor elements.

The third subsystem is an input/output subsystem. It interfaces to disk array systems, Fast SCSI, tape drives and networks which are Ethernet, HIPPI, ATM, and FDDI network. The subsystem also includes internal hard disc drives and tape drives.

The operating system of SR2201 is HI-UX/MPP, which is based on the Mach 3.0 micro-kernel. For user convenience, parallel FORTRAN, optimized FORTRAN 90, optimized C and C++ are all supported. The parallel software development environment consists of Parallelware, a performance monitor and a symbolic debugger. Common applications for scientific computing are supported by parallel computing feature.

The system holds up to 2,048 processor elements. The scale of one processor element is more than one million gates. The total scale of the system, then, is more than two giga gates. There is no logic simulator available today that can simulate this huge system using gate-level description. Therefore, designers require the new verification system to validate the SR2201 design in a manner representative of the complete design.



(a) Network subsystem @ @(b) Processor Element

Fig. 1 Structure of SR2201

III. Parallel Network Simulation

In the development of conventional large-scale general purpose computers, we have achieved greater than 99% detectability of design errors before the start of physical design, using designer's checking through system level simulation using VELVET[6]. In addition to the conventional verification methodologies passed to SR2201 from experience in previous models, a new verification method, "critical cluster simulation", is introduced to fully utilize actual test-and-maintenance programs in system simulation. It is necessary to use test-and-maintenance programs to assure quality assurance and precise debugging of design information.

The new critical cluster simulation approach to design verification of huge systems uses a test program which generates network transmission sequences randomly. The critical cluster is based on the regular structure of the massively parallel system. It consists of several processor elements, related logic circuits and the network among them.

In a conventional projects, whole-system simulation is used to find hard-to-detect errors. However, whole SR2201 simulation is beyond the capacity of our simulator. Instead two through twelve processor elements are used to define one critical cluster with behavioral microprocessor model and other gate level description. Though the division process is enormous with the new approach, further simulator performance enhancement and efforts to realize high-speed interface signal exchange simulation among processor elements are required.

The objective of parallel network simulation is to find hard-to-detect errors, such as errors which result from processor element interaction. The most effective approach for detecting these errors is to execute a test program which generates network transmission sequences randomly in a large processor element network. Therefore, a parallel network simulation environment should have the ability to run the test program with large network system integration and high performance.

Figure 2 illustrates an event-driven system, which is basically the result of simple expansion of our traditional two-level simulation system to a parallel type. We expanded the system in such a manner that, when an instruction to start communication is executed, the architecture simulator switches to the logic simulator and that, when the communication is complete or enters a wait state, the logic simulator switches back to the architecture simulator. However, this event-driven system poses the following problems:

(1) As many processor elements as required in a network configuration need to be installed in the logic simulator, so a large-scale logic circuit cannot be simulated because of the limitations on the memory capacity of the logic simulator.

(2) The logic size in the logic simulator tends to increase, resulting in an extreme decline in simulation performance.

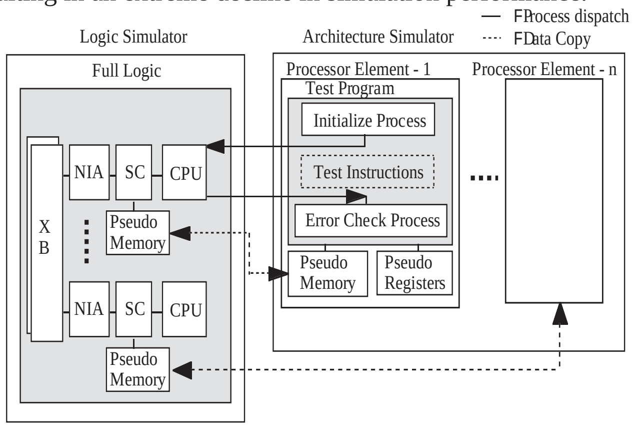


Fig. 2 Parallel Network Simulation (Event Driven)

In order to resolve these problems, our new method no longer simulates processor elements in the logic simulator but instead it simulates them in the architecture simulator. To accomplish this, we designed a batch scheduling system where large-scale parallel simulations can be conducted through a state transition that occurs in a network event[7].

Figure 3 illustrates the batch scheduling parallel network simulation scheme. It consists of an architecture simulator and logic descriptions of network/storage controllers. The architecture simulator simulates multiple microprocessors simultaneously. The number of microprocessors depends on the simulation target network configuration. Intensive chip simulation will verify the correctness of the design in advance. Therefore, in the parallel network simulation, microprocessors act only as request/response queue handlers and program interpreters.

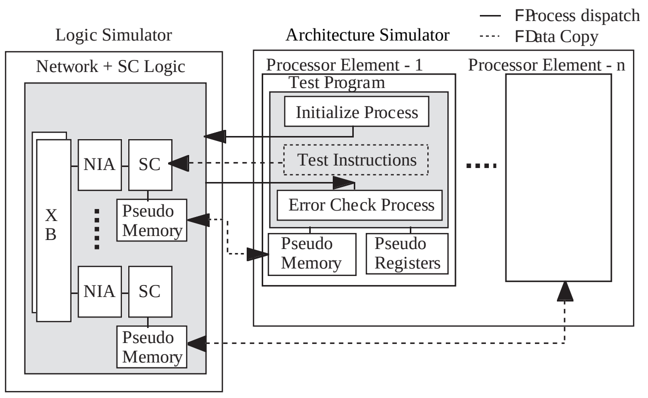


Fig. 3 Parallel Network Simulation (Batch Scheduling)

However, the following problems still remain unsolved in this system:

(1) A state transition takes place every time a network event occurs. Optimizing the amount of data being copied still entails high overhead.

(2) The amount of time consumed by the behavioral model in monitoring network events is excessive.

(3) A time lag occurs between the behavior simulated by an architecture simulator and that simulated by the logic simulator, so the simulated results differ from the actual logic results in timing

We therefore designed a cyclic synchronization system as shown in Fig. 4 to realize high-speed, large-scale parallel network simulation. With this system, we solved the problem in timing control by executing the logic simulator and architecture simulator in a cyclic manner and eliminated the overhead in data copying by forcing the logic simulator to reference the pseudo-memory of the architecture simulator.

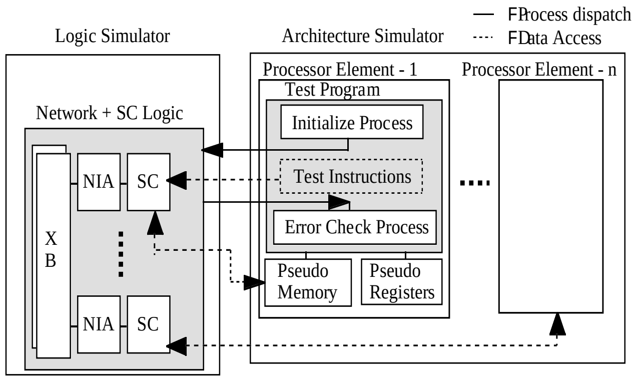


Fig. 4 Parallel Network Simulation (Cyclic Synchronization)

The simulation execution algorithm in the parallel processors′evaluation sequences is critical in accurate, high speed-simulation. Figure 5 illustrates the dispatching algorithm of the simulation execution sequence. The parallel network simulation environment adopts cycle synchronization in a microprocessor execution sequence, basically. The simulation progresses from left to right in the figure. Each box denotes the execution timing of a specified part of the simulation models. Cyclic synchronization means that the architecture simulator transfers control of each microprocessor model after one instruction execution. After the complete microprocessor model execution, network and storage control models are simulated for one cycle using signal values from the architecture simulation results. On the next clock cycle, instruction execution began after evaluation of the gate level logic.

The initialization and error check phases need no synchronization. These phases are executed on the architecture simulator side, and only the network test phase requires rigid synchronization in order to preserve simulation validity, though simple synchronization mechanism increases the overhead of simulation time. The sequence of the simulation execution influences the consistency of memory coherence.

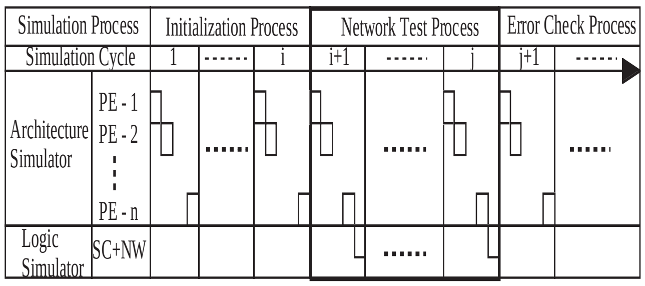


Fig. 5 Simulation Execution Sequence

A. Memory Consistency

Memory modeling is another key issue in improving simulation speed. Figure 6 illustrates a pseudo-memory access method, from the logic simulator to the architecture simulator. Pseudo-memory is implemented only in the architecture simulator. The logic simulator and the architecture simulator access pseudo-memory directly in each execution process. Only upon network communication requests does the architecture simulator access pseudo-memory, via storage control logic on the logic simulator. Only the minimum number of memory requests are executed on the logic simulator, so logic simulator performance is improved because most memory requests are handled on the architecture simulator in one simulation cycle.

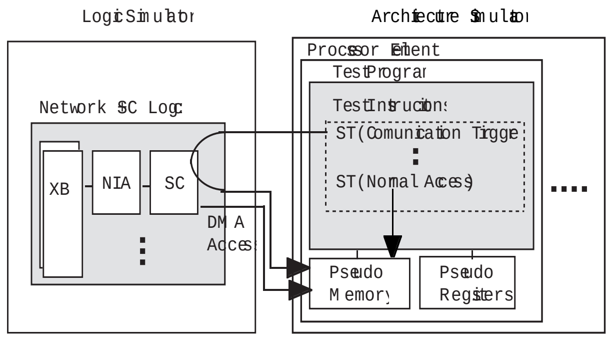


Fig. 6   Pseudo-Memory Access method

However, this technique raises a difficult problem in the ordering of memory requests. Figure 7 shows the timing chart for a typical memory operation. In this case, a test program on the architecture simulator cannot refer to correct status information if interrupted.

If a partial write operation with flag data is finished in the i+1 cycle, the test program in the architecture simulator can read the flag data, but if the flag data write is delayed for 2 more cycles due to the SC being busy, the test program cannot read the flag data. In such cases, then, the interruption report from the logic simulator to the architecture simulator has to be delayed for 2 simulation cycles. We adopted a request scheduler written in simulation control language, which defers an interruption report to the architecture simulator during an SC busy.

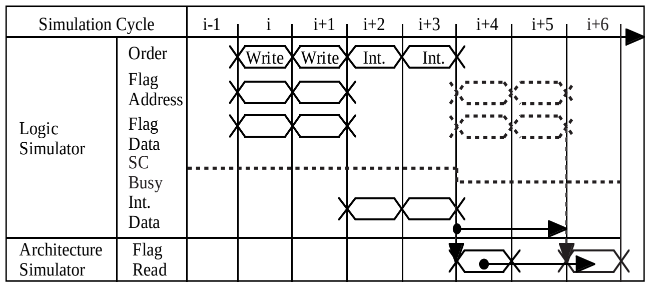


Fig. 7 Execution Sequence of Memory Operation

B. Performance Enhancement

Performance enhancement, i.e. improvement of the percentage of simulation time devoted to simulation and reduction of the percentage of time spent on overhead tasks (copying data, checking network events, running the architecture simulator, and initializing and ending the simulation) has heretofore been achieved by batch scheduling. Now, a new method, cyclic synchronization, has achieved significant improvement over the previous method.

The conventional method: batch scheduling

Where this system is not employed, all logic units composed of parallel processors must be mounted on the logic simulator, so the logic simulator's performance will decline significantly. The use of this system has made it possible to minimize such a decline in the logic simulator performance. Contrary to our first expectations, however, the result is that a total of 58% of the entire simulation time is still seen as overhead. This figure is the sum of 50% for data copying in the event of state transitions between the architecture simulator and the logic simulator every time a network event occurs, and 8% for behavioral model operation for monitoring network events.

The improved method: cyclic synchronization

Cyclic synchronization increases the speed of simulation 2.4 times over the conventional method (batch scheduling). It eliminates 58% of the simulation time applied to overhead tasks in batch scheduling, by adopting the previously discussed cyclic synchronization, and enabling direct memory access from the logic simulator to the architecture simulator.

IV. Results

High performance and effective parallel network simulation environment has been achieved. Figure 8 compares the simulation execution time for the cyclic synchronization method vs. the batch scheduling method with eight processor models.

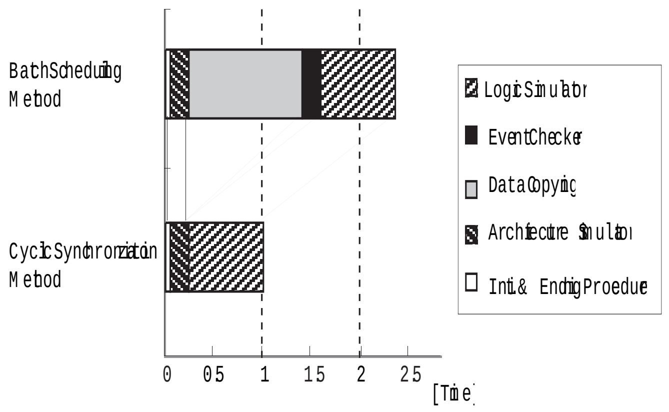


Fig. 8 Results of Parallel Network Simulation

Applying the parallel network simulator and verification methodologies to the SR2201 project, high quality design is achieved at the verification phase. Four percent of hard-to-detect errors are detected using the parallel network simulation, and 98% of all design errors are detected before physical design.

Figure 9 shows the result of the error detection rate at each verification stage.

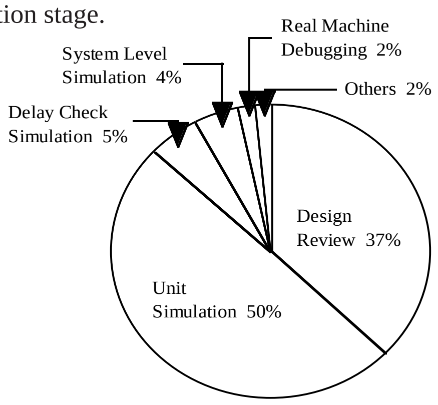


Fig. 9 Error Detection Analysis

Most of the design errors are detected at the machine debugging stage are delay errors. The influence of high temperature and electric noise to most CMOS LSIs was not estimated accurately.

V. Conclusion

Verification of the giga-scale SR2201 massively parallel processor requires a new methodology and performance enhancement of the simulation environment. A system level verification methodology for the giga-scale massively parallel supercomputer have been established. To avoid the huge capacity problem, we adopted the "critical cluster" simulation, which includes gate-level network related LSI descriptions and up to 12 behavioral microprocessor simulation models. Cycle rotation and other simulation scheduling enhancement, based on the execution time analysis of the critical cluster simulation attains more than twice the speed of a conventional simulator. Results show 98% detectability of all the design errors in the SR2201 design before physical design, and contributed to shortening the development time of the huge machine.

VI. References

[1] James Monaco et.al., "Functional Verification Methodology for the PowerPC 604," Proceedings of the 33rd Design Automation Conference, pp.319-324 (June 1996).

[2] C. Montemayor et. al., "Multiprocessor Design Verification for the PowerPC 620 Microprocessor," Proceedings of International Conference on Computer Design '95, pp.188-195 (Oct. 1995).

[3] Masayuki Miyoshi et.al., "An Extensive Logic Simulation Method of Very Large Scale Computer Design, " Proceedings of the 23rd Design Automation Conference, pp.360-365 (June 1986).

[4] Kotaro Shimamura et.al., "A Superscaler RISC Processor with Pseudo Vector Processing Feature, " Proceedings of International Conference on Computer Design '95, pp.102-109 (Oct. 1995).

[5] Kisaburo Nakazawa et.al., "The Architecture of Massively Parallel Processor CP-PACK," Journal of IPSJ, pp.18-27 (Jan. 1996).

[6] Kaoru Suzuki et.al., " System Level Verification of Large Scale Computers," Proceedings of the International Conference on Computer Design '92, pp.149-152 (Oct. 1992).

[7] Kaoru Suzuki et.al., "Logic Verification Technique for the Parallel Processor System," Proceedings of the 49th Information Processing Society Conference, pp.129-130 (Sept. 1994).