HAL II: A MIXED LEVEL HARDWARE LOGIC SIMULATION SYSTEM

Shigeru Takasaki*, Tohru Sasaki**, Nobuyoshi Nomizu*,
Hiroshi Ishikura* and Nobuhiko Koike***

* NEC Corporation 10, 1-chome, Nisshin-cho, Fuchu City, Tokyo, 183 Japan
** NEC-TOSHIBA Information Systems, Inc. Shiba, Minato-ku, Tokyo, Japan
*** NEC Corporation C&C Systems Labs. Kawasaki, Japan

Abstract

This paper describes a mixed level hardware logic simulation system, called Hardware Logic Simulator II (HAL II). This paper first shows a HAL II simulation method. Then, it overviews HAL II hardware and software system configurations, simulation mechanism and estimates system performance. The HAL II system can handle a maximum of 5.8 million gates and a high level design language FDL (Functional Description Language). Finally, it discusses system applications and results. The paper also indicates that HAL II has been successfully used.

I. INTRODUCTION

In order to manufacture highly reliable and cost effective digital systems, LSIs have played an important role. When the improved LSI technology is used in designing digital systems, network designers have to pay careful attention to differences between a custom made LSI network design method and previous SSI and MSI network design methods. Using the previous methods, it is easy to manufacture a prototype digital system after an original design and to verify the system functions on the prototype. On the other hand, it is not easy to manufacture a prototype digital system using custom LSIs, because design modifications caused by logic errors are quite difficult to make. The reason is that LSI refabrications require a long turn around time and are expensive. Therefore, network designers have to carefully verify custom LSIs and system functions before LSIs are fabricated.

To verify/guarantee system functions/qualifications completely, a test program, which checks system instructions and diagnoses, is carried out. The test program length is more than 10^6 clocks in a large computer system. It is impossible for a software logic simulator to carry out the test program in a reasonable computer run time. Therefore, a high speed hardware logic simulator is an indispensable tool to meet these requirements in the LSI design era.

Up to now, several hardware simulators have been reported [3][4]. The implemented systems are YSE, LSM and ZYCAD simulators. However, they were gate level simulators and used the compiled simulation method. Though gate level simulation is a useful tool, it requires a long simulation model generation time and simulation time, because of the large amount of network connection information and status propagation. To overcome gate level simulation problems, a block level hardware logic simulator, called HAL, was developed [1][2].

Though the HAL system has lots of advantages, it limited the block size from several gates to several hundred gates. As LSIs become larger, blocks generating from LSIs become larger. In order to achieve these technology improvements, a new block hardware logic simulator was developed. The simulation method for blocks is drastically improved. The HAL II system can handle block size ranging from several gates to two hundred thousand gates. In addition, the HAL II system can deal with a high level design language FDL (Functional Description Language) [5]. This expands the HAL II system application field. Such applications are, for example, higher level computer system simulation based on FDL descriptions, and hierarchy verification using FDLs and logic design files. The HAL II system has these features.

II. SIMULATION METHOD

1) Simulation Algorithm

HAL II adopted two logic values, i.e., zero and one, and a zero delay simulation technique. The main HAL II objective is to simulate very large computer system functions. In the logic simulation, especially computer system level simulation, two logic values are sufficient to simulate the system.

From the timing analysis view point, it is necessary to carry out minimum, maximum and nominal delay simulations, in addition to the logic simulation. This delay and logic simulator can simulate asynchronous circuit behaviors. However, in order to implement this kind of simulator, it is necessary to have almost three times the memory capacity of a zero delay simulator. Besides the additional memory capacities, there is a simulation speed problem. This simulator method utilizes a time-wheel technique for simulation. In this technique, simulation events are set on the time-wheel, based on media and circuit delay, and the same time events are only simulated concurrently. Therefore, simulation concurrency, that is, parallel processing possibility, is reduced

0738-100X/86/0000/0581$01.00 ©1986 IEEE

compared to a zero delay simulation.

On the other hand, though a zero delay simulation can only handle a synchronous logic circuit, it has many advantages. From the simulation processes, it can utilize level sorting/ordering technique. This can increase simulation concurrency, i.e., parallel processing ratio in gate evaluations. That is, gates belonging to the same level in the block can be simulated simultaneously. Hardware implementation requires only intermediate gate status memory. A simulation controller is also simpler than a nominal delay simulator.

As most general purpose mainframe computers are made up from synchronous logic circuits, it is sufficient for a zero delay simulator to simulate their logic functions. In large computer systems, there are lots of logic functions to be verified before LSIs/PKGs in the systems are fabricated. If a nominal delay simulator requires a several times longer simulation speed than a zero delay simulator, for logic designers to carry out the test programs using this tool is intolerable.

Though the timing analysis and asynchronous simulation using a nominal delay simulator are also important for digital network designs, they have a lower ratio compared to synchronous logic function verifications, from the current computer system view point.

A unit delay simulation is located in between a zero delay and a nominal delay simulation, and is a very efficient technique. However, this simulation also requires two status memory sets for storing gate input and output values. It has the same problems for logic stabilization as the nominal delay simulator.

2) Block Definition

A block is defined as a set of gates/memories. A set of gates is called a logic block. Similarly, a set of memories is called a memory block. A block is generated from real logic design files or FDL description files.

A) Logic block

A logic block must be a combinational circuit consisting of primitive gates, i.e., AND, OR, EXOR etc. Therefore, a custom LSI, including scan registers, must be partitioned into two or more combinational circuits, i.e., logic blocks. A combinational custom LSI has one-to-one correspondence to a logic block. For example, as ALU, decoder and selector chips are combinational circuits, they correspond to logic blocks. On the other hand, a register chip, including selectors, random logics and registers, must be partitioned into two or more logic blocks. The FDL descriptions are transformed into a logic block consisting of primitive gates by a software program. This will be described in Section III.

B) Memory block

A memory block is used to describe a set of memory chips having an address, read date, write data and control signal. Therefore, many memory chips are modeled as a single memory block. The FDL descriptions are also transformed into a memory block by the software program.

3) Block Level Simulation

HAL II is a block level event driven simulator. Simulation networks, for example, in computer systems are modeled as block-connections. Blocks are arranged in sequence according to block depth from primary inputs/registers to primary outputs/registers. Simulation can be carried out from input side blocks to output side blocks. Blocks belonging to the same level are simulated simultaneously. Block evaluations are carried out based on block models consisting of gates. The gate evaluation also makes use of a level sort technique. Gates are evaluated concurrently in blocks.

III. HAL II SYSTEM OVERVIEW

1) Hardware System Overview

A HAL II system blockdiagram is shown in Fig. 1. HAL II consists of 29 logic processors, two memory processors and one master control processor. It is also possible to have 27 logic and 4 memory processor configuration. Each processor is connected to a router cell network. Logic processors are used to simulate random logic custom LSIs. A logic processor consists of a block processor and a logic block simulator. Memory processors are used to simulate ROMs and static/dynamic RAMs. A memory processor consists of the block processor and a memory block simulator. The master control processor is used to control the overall system, to load initial data and to store simulation results in memories.

A) Logic processor

o Block processor

The block processor performs event processing. It holds input/output pin data and an output pin connection. It receives an event packet, including a block category, and its input pin data from the router cell network. After setting an event in the input status memory, it fetches an event and input pin data, and sends the data to the logic block simulator.

Then, it receives the simulated block outputs from the logic block simulator and compares the block outputs with previous output values. If any change is detected, it searches for input pins connected to the changed output pin. After making event packets to change the input pin value, it sends them to the router cell network.

o Logic block simulator

The logic block simulator simulates block functions. It receives block input pin logic values along with the block category from the block processor. The block simulation is performed through block element evaluations from the inputs to the outputs. The logic block simulator sends the simulated output pin values to the block processor.

B) Memory processor

o Memory block simulator

The memory block simulator carries out the same functions as static/dynamic RAMs. It receives an address, memory control bits and an array of write data as an input and returns an array of read data as an output. It has a direct interface with the main memory for the host processor. It uses a part of the main memory as a storage area for memory LSIs.

C) Master control processor

The master control processor contains a packet transceiver, an execution controller and a DMA (Direct Memory Access) interface with host processor. The packet transceiver communicates with logic processors, in the form of a packet via the router cell network.

The execution controller performs level and clock synchronization among logic processors. All blocks are arranged by signal propagation order and grouped into levels. The logic processors simulate blocks belonging to the same level at a simulation period. When all logic processors have finished simulation for a simulation cycle, the simulated results are saved in the host processor memory and the execution controller broadcasts a level start command to all logic processors to start the next simulation cycle.

D) Router cell network

The router cell network is a multi-stage interconnection network with 32 input ports and 32 output ports. It consists of 80 router cell LSIs, which are a store-and-forward switch with two input ports and two output ports. Packets are transferred through this router cell LSI in a pipeline fashion.

2) Software Supporting System Overview

A software system blockdiagram for HAL II is shown in Fig. 2. It mainly consists of three functions. Individual functions are as follows:

i) Block model generation

Logic and memory models are generated using LSI design files, FDL description files and a LSI component gate library. A random logic LSI, including registers, is partitioned into combinational blocks. Gates in the blocks are arranged, according to logic depth, from block inputs to outputs. When FDL descriptions are used for logic network design, logic/memory blocks are generated from FDL description files by a FDL translator. At this stage, random logic descriptions are transformed into primitive gate networks.

FDL descriptions for an LSI are shown in Fig. 3 (a). In the descriptions, for example, a statement Z (called node Z) is transformed into primitive gates by the FDL translator. That is,

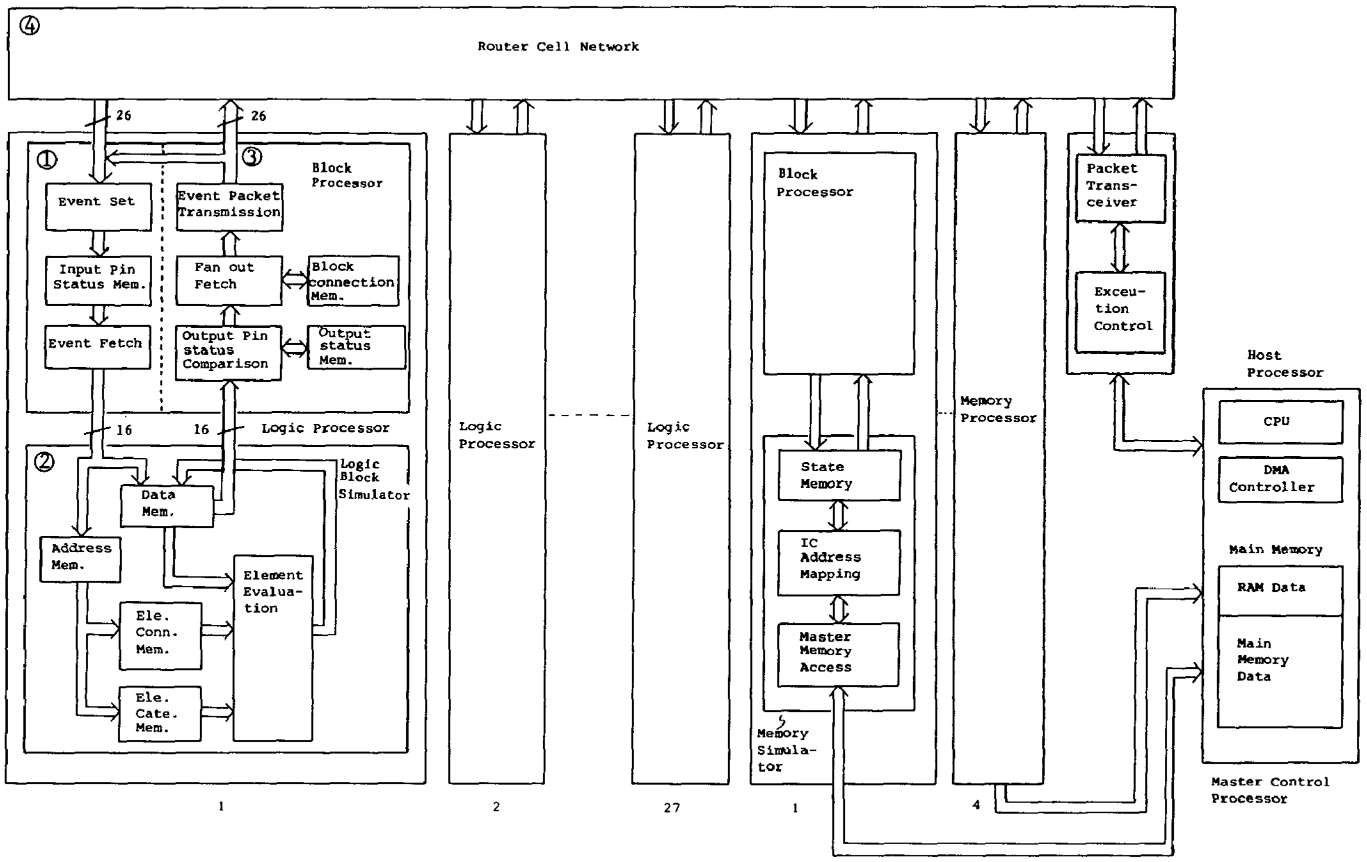


Fig. 1 HAL II System Blockdiagram

the primitive gate syntheses for the node Z are carried out. This is shown in Fig. 3 (b).

ii) Inter-block connection generation

A system design file, consisting of multi-chip-packages and boards, is regenerated as inter-block connections using the block input/output file. Therefore, the system design file becomes a file of block connections. The blocks are also arranged according to block depth from system inputs/registers to registers/outputs. After that, the blocks are optimally allocated to an individual processor. The optimal allocation means that nearly the same block numbers are allocated in an individual processor.

iii) Simulation execution control

This supports functions necessary for the simulation preparation, including loading simulation models, controlling simulation execution, running test programs and firmware, and listing traced signal time charts. As many user's commands are provided, users can easily control the simulation execution.

IV. SIMULATION MECHANISM

A simulation mechanism is shown in Fig. 1 (from ① to ④). Using Fig. 1, HAL II Simulation processes are explained, as follows:

S-1: Block processor (①)

Checks input status memory data, including events, block categories and input pin logic values, in sequence and detect an event. Then, it fetches the event and determines the block category and the input pin logic values. Sends the block category and the input pin logic values to the logic block simulator, and resets the event flag. If a level flag, which is used for a final level simulation flag, is detected during checking the input status memory, it stops detecting an event, and sends a level final signal to a control processor.

S-2: Logic block simulator or memory block simulator (②)

Logic block simulator or memory simulator evaluates a block based on the block category and the input pin logic values, and returns the block output pin logic values to the block processor. A block simulation mechanism in logic block simulator is set up in the following ways:

S-2-1: Sets the input pin logic values to Data Mem, and determines a start address for Element Connection Memory and Element Category Memory from the address memory according to the block

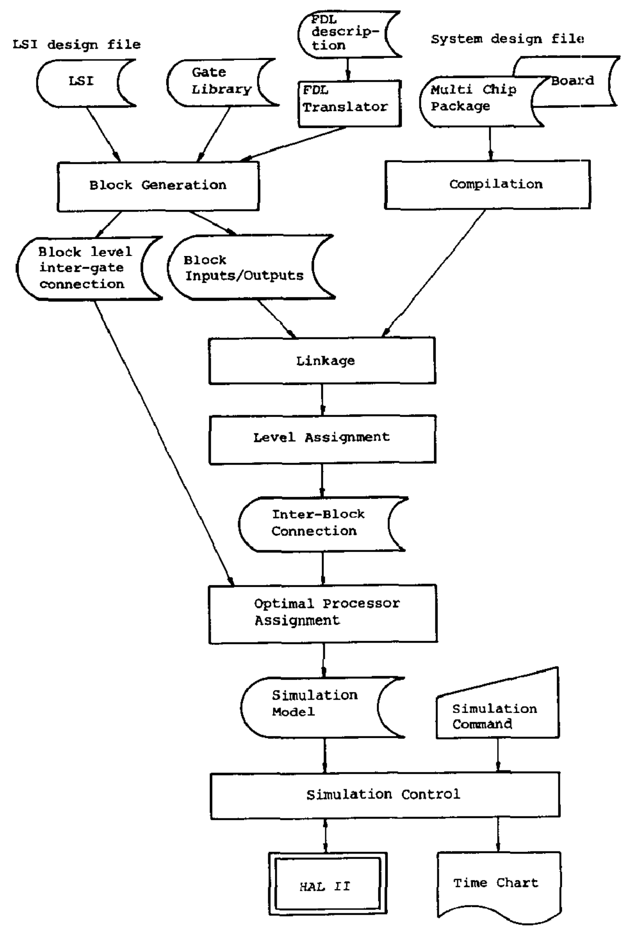
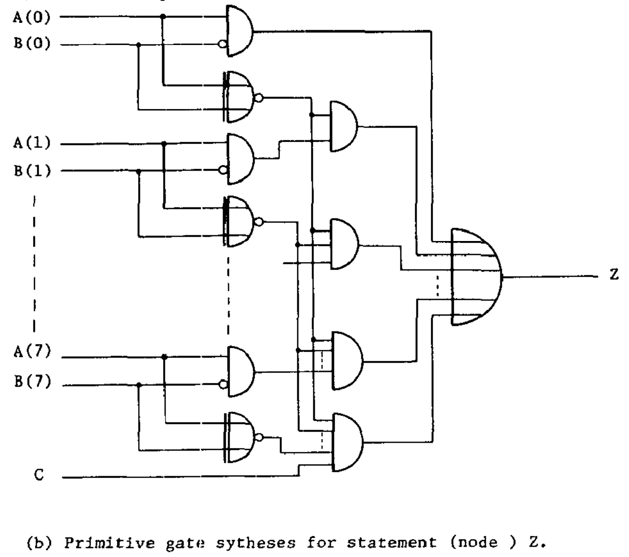


Fig. 2 Software System Blockdiagram

```
INPUT FC(0:8),SQ(0:3), -----,SI(4:4);
OUTPUT SX(0:8),SXP, -----,EIF,SOT;
FCMU = SQ0'*SQ3'*FC(0);
STF = ALM + CTO + CPW + IO1 + CCM + IO2 + DRQ + CSE ;
                 :
                 :
∘ Z = (A(0:8).GT.B(0:8)).OR.((A(0:8).EQ.B(0:8))*C);
                 :
                 :
EIF = QEIF(0:18).ROR.;
SOT = QEIF(0);
```

(a) FDL descriptions for an LSI.



(b) Primitive gate sytheses for statement (node) Z.

Fig. 3 FDL Descriptions and Primitive Gate Syntheses

category. The start address is sent to an address counter.

S-2-2: Reads the previous level logic values from Data Mem, and element categories from Element Category Memory.

S-2-3: Evaluates the current level elements from the read data. For example, eight gates are evaluated simultaneously.

S-2-4: Stores the evaluated element logic values to Data Mem.

S-2-5: Checks the evaluated element level. If the level is final, sends the block output pin logic values to the block processor (S-3). Otherwise, it increases the level and the address counter and goes to S-2-2.

S-3: Block Processor (③)

Compares the new block output pin logic values with the previous block output pin logic values in the output status memory, and sets the new value to the status memory.

S-4: Block processor (③)

If any output pin value change is detected, it determines packets, including processor, block and input pin numbers, to which the changed output pins are connected, by checking the block connection memory.

S-5: Router cell network (④) or block processor

Sends the packets to the processors connecting the changed pins via the router cell network. If the processor is the same processor, i.e., itself, it sends the packets directly to the same processor packet input.

S-6: Block processor (①)

Receives changed input pin values as a packet.

S-7: Block processor (①)

Updates the input status memory by using block and pin numbers, and sets an event flag. Then, it goes to S-1.

Process 1 (S-1, S-6 and S-7), Process 2 (S-2), Process 3 (S-3 and S-4) and Process 4 (S-5) can be operated in pipeline processings.

V. SYSTEM PERFORMANCE

1) System Capacity

The HAL II block, processor and system capacities are as follow:

A) Block capacity

Block	Logic	Memory
Capacity (Max.)	200,000 gates	512 K bytes
Inputs (Max.)	128 pins	Address: 13 pins Write: 64 pins Control: 2 pins
Outputs (Max.)	128 pins	Read: 64 pins

B) Processor capacity

Block	Logic	Memory
Capacity (Max.)	200,000 gates	2 M bytes
NO. of Blocks (Max)	1024	1024
Block category (Max.)	128	128

C) HAL II system capacity

Block	Logic	Memory
Capacity (Max.)	5.8 million gates	4 M bytes
NO. of Blocks (Max.)	29696	2048
Block category (Max.)	3712	256

2) System Performance Estimation

The clock simulation processes were described in Section IV. The simulation processes can be partitioned into four phases. They are Block Processor ① (S-1, S-6 and S-7), Logic Block Simulator or Memory Block Simulator ② (S-2), Block Processor ③ (S-3 and S-4) and Router Cell Network ④ (S-5). They can be operated in pipeline processings. Therefore, system performance can be estimated in examining each phase. The following are assumptions for estimating individual phase performance.

A1. Active total block event ratio per simulation cycle is E_v. For example, E_v is 2/3.

A2. Number of generated events per block evaluation is e_v. For example, e_v is 10.

A3. A block consists of g gates. ex., g=1500.

i) One block simulation cycle time

a) Block Processor ① : Input pin status memory

A block simulation requires e_v accesses, and 1 event fetch access. Therefore, input pin status memory processing time INM_T is computed as follows:

$$INM_T = (e_v + 1) \times M_I \text{ .... } (1)$$

where

M_I: input pin status memory access time, ex., M_I=300ns in HAL II.

b) Logic Block Simulator ② : Logic Block Simulator Processing time

Logic Block Simulator processing time LBS_T is computed by the element connection memory access time, No. of gates, required logic connection bits per gate, No. of parallel evaluation gates at a time and parallel evaluation efficiency ratio.

$$LBS_T = M_A \times m \times g \times \frac{1}{p} \times \mu \text{ .... (2)}$$

where

M_A: the element connection memory access time (This is the longest access time in Logic Block Simulator.), ex., M_A=300ns in HAL II,

m : required connection bit unit per gate, ex., 8 bits when m=1,

g : No. of gates, ex., g=1500,

p : No. of parallel evaluation gates at one time, ex., p=8,

μ : parallel evaluation efficiency rate, ex., μ=1,

c) Block Processor ③ : Block connection memory processing time

In order to send e_V packets, it is necessary to have e_V block connection memory accesses. Therefore, block connection memory processing time BCM_T is computed as follows:

$$BCM_T = M_c \times e_v \text{ .... (3)}$$

where

M_c: block connection memory access time, ex., M_c=450ns.

d) Router Cell Network ④ : Packet transfer processing time

Router cell packet transfer time RTR_T can be computed by number of block events, packet transfer time per event, transfer degradation ratio due to packet collision, and ratio of blocks sent to some other processor.

$$RTR_T = e_v \times e_t \times c_d \times T_r \text{ .... (4)}$$

where

e_t: packet transfer time per event, ex., e_t=400ns in HAL II,

c_d: transfer degradation ratio due to packet collision, ex., c_d=1.4,

T_r: ratio of blocks sent to some other processors, ex., T_r=0.8.

As these phases can be operated in pipeline processings, one block simulation cycle time S_i is obtained as follows:

$$S_i = \text{Max } \{INM_T, LBS_T, BCM_T, RTR_T\} \text{ ... (5)}$$

ii) System performance

System performance S can be computed by number of blocks, one simulation cycle time, number of processors, parallel processor efficiency ratio, block event ratio and number of repeated processing times per simulation cycle.

$$S = \frac{1}{N_p} \times \frac{1}{P_r} \times E_v \times \delta \times \sum_{i=1}^{G_N} Si \text{ .... (6)}$$

where

G_N: number of blocks,

S_i: one simulation cycle time,

N_p: Number of available logic processors, ex., N_p=27,

P_r: parallel processor efficiency ratio, ex., P_r=0.4,

E_v: active block event ratio per cycle, ex., E_v=2/3,

δ : number of repeated processing times simulation cycle, ex., δ=1.

Example:

What is the clock simulation time required to simulate a 1 million gate computer system?

Assumptions: This system is designed with fully customized 1500 gate LSIs, including registers.

INM_T, LBS_T, BCM_T, RTR_T and S_i are computed based on Eqs (1) to (5), as follows:

$$INM_T = (10 + 1) \times 300 = 3.3 \ \mu s$$

$$LBS_T = 300 \times 2.5 \times 1500 \times \frac{1}{8} \times 1 = 141 \ \mu s$$

$$BCM_T = 450 \times 10 = 4.5 \ \mu s$$

$$RTR_T = 10 \times 400 \times 1.4 \times 0.8 = 4.3 \ \mu s$$

$$S_i = \text{Max } \{INT_T, LBS_T, BCM_T, RTR_T\} = 141 \mu s.$$

In order to design this system, required LSIs N_L are

$$N_L = \frac{100 \times 10^4}{1500} = 670.$$

If an LSI is partitioned into 2 blocks, the number of blocks for G_N is

$$G_N = 670 \times 2 = 1340.$$

Therefore, system performance S is

$$S = \frac{1}{27} \times \frac{1}{0.4} \times \frac{2}{3} \times 1 \times 1340 \times 141 = 12 \text{ ms}.$$

This is more than 1000 times faster than a software simulator.

VI. APPLICATIONS

1) Various Simulation Methods

The HAL II system can support not only block level simulation, but also gate level simulation and mixed (gate, block and FDL) level simulation. When a block has only single level primitive gates, HAL II can perform the gate level simulation. Therefore, it has various application methods. For an LSI processor or a small size computer system, the gate level simulation method is useful. For a large computer system, the block level simulation method is more useful than the gate level simulation method. The FDL level simulation has many advantages. For example, when a newly designed part is not fixed as LSI design files, or when designers want to utilize some other maker's LSIs, whose internal detail logics are not clear, the FDL level simulation is quite useful. As designers can describe the newly designed part/other maker's LSI functions by using the FDL, they can simulate the total system.

2) Application Fields

A) LSI design verification

As LSIs become larger, their design verifications become harder and harder. A large number of test patterns are required for verifying LSI logic functions. Furthermore, many custom made LSIs have been developed. Therefore, if these LSIs are verified by a software tool, they require enormous computer resources. HAL II can easily be applicable for LSI design verifications.

B) Computer system design verification

As the HAL II system can perform test programs before building up computer systems, most logic errors are removed. Therefore, at the system test, only assembling faults remain. It is easy to detect them. This design verification method is quite useful to a system consisting of custom made LSIs. This is because the custom made LSIs cannot be refabricated when the design cycle and cost are critical.

HAL II can perform a system test on firmware. This test tool checks system firmware functions. Therefore, HAL II can verify both system logic and firmware functions.

C) System diagnosis evaluation

HAL II is basically a block level simulator. This technique can be used for a system diagnosis evaluation. Block boundaries are real chip terminals or registers. Therefore, HAL II can set up a pseudo-failure on these boundaries and check whether or not a system diagnosis circuit detects this fault. Sometimes, it is quite difficult for a prototype hardware system to set up desired pseudo-failures. On the other hand, HAL II can easily set up the pseudo-failures on any block boundaries.

VII. RESULTS

Both gate and block level simulations were carried out on the hardware simulator. These results are shown in Table 1.
From Table 1, it is clear that the block level simulation is about 10 times faster than the gate level simulation, in regard to both simulation model generation and simulation execution time. The simulation execution efficiency can be considered due to less network status propagation and concurrent evaluation of several gates. The modeling efficiency is also due to the fact that less network connection information is required. That is, the smaller the number of gates/blocks and signal lines among gates/blocks become, the shorter the linkage and level assignment time becomes.

Table I. Comparison between gate and block level simulation results.

Simulation Category	Gate1	Gate2	Block1	Block2
System size (K gates)	130	100	110	105
Model generation time (15 MIPS,sec)	871	685	96	91
No. of total processors	32	28	32	27
No. of simulation clocks (K clocks)	100	100	100	100
Total simulation time (sec)	901	1071	101	150
One-cycle simulation time (mili-sec)	9	10.7	1	1.5

VIII. CONCLUSION

A mixed level hardware simulation machine HAL II simulation method, system overview, and performance have been presented. The mixed level simulation has many effective applications. The simulation results have shown the block level simulation efficiency. The HAL II system can be used as an effective practical design tool in custom LSI design environments.

ACKNOWLEDGEMENT

The authors would like to thank Dr. Saito, Mr. Kuwata, Mr. Sugimoto and Dr. Yamamoto for their encouragement and management efforts. They are also grateful to Mr. Minowa, Mr. Tanaka, Mr. Kurashita and Mr. Kikuchi for developing the HAL II system.

REFERENCES

[1] N. Koike, K. Ohmori, H. Kondo and T. Sasaki, "A High Speed Logic Simulation Machine", Digest of papers COMPCON Spring, 83. pp. 446-451, March, 1983.

[2] T. Sasaki, N. Koike, K. Ohmori and K. Tomita, "HAL: A Block Level Hardware Logic Simulator", Proc. of 20th Design Automation Conference, pp. 150-156, June, 1983.

[3] G. F. Pfister, "The Yorktown Simulation Engine: Introduction:, Proc. of 19th Design Automation Conference, pp. 51-54, June, 1982.

[4] T. Blank, "A Survey of Hardware Accelerators used in Computer-Aided Design", IEEE Design & Test Vol. 1, No. 3, pp. 21-39, Aug. 1984.

[5] S. Kato and T. Sasaki, "FDL: A Structual Behavior Description Language", 6th Int. Symp. CHDL., pp. 137-152, May 1983.