Paper

A Test Program Generation for Scalable SIMD Parallel Computer

Member Akira Iwase (Mitsubishi Electric Corp.)

Non-member Tetsuaki Isonishi (Mitsubishi Electric Corp.)

Non-member Hiroyuki Miyata (Mitsubishi Electric Corp.)

Member Hisao Koizumi (Tokyo Denki University)

Computer hardware testing is performed with test programs generated by using machine instructions or high-level language. A parallel computer is characterized by the complexity resulting from its system configuration consisting of an array of a number of processors, and also by the parallel processing unit whose hardware configuration can be varied corresponding to the system objects and the required performance conditions. Because of this, it is required to prepare the test programs corresponding to the respective hardware configurations. This means that the number of test programs required increases with the number of hardware configuration types designed, and this requires a tremendous amount of labor for their generation.

This paper proposes a system for efficiently generating the test programs to be used for hardware testing of the scalable SIMD parallel computer. This system makes the most of the functions and features of the scalable SIMD parallel computer, and generates the test programs without depending on the hardware configuration of the parallel processing unit. By using this system, it is possible to reduce the types and number of the test programs, and consequently, the period for the development.

Keywords: test program, hardware testing, SIMD, parallel computer

1. Introduction

Along with the enhancement of performance in the general-purpose microprocessor and the progress in the multiprocessor, computers are showing a remarkable improvement in performance. However, in multi-media processing and others, the amount of data to be processed is tremendous, and much greater computing power is required. As a possible solution for this problem, a scalable parallel computer based on VLSI technology is being looked for. In particular, for high speed and efficient execution of various routine processing of two-dimensional image data, application of the SIMD (Single Instruction Multiple Data stream) system is effective as a parallel processing system making most of the parallelism included in such image processing [1].

This paper discusses the test program to be used for the hardware testing of this SIMD parallel computer. Hardware testing means detecting, pinpointing the location of and finding out the cause of any fault by operating the system hardware under the conditions equivalent to those in the actual operating environment. The hardware testing is executed by using test programs generated by human effort by using machine instruction or high-level language. When generating such test programs, it must be taken into consideration that the specifications of the parallel processing unit and hardware configuration can vary according to the system objects and required performance. If the test programs are to be created for all of required hardware configurations, it would require a large amount of manpower and much longer development period. The authors have been concerned in the development of the SIMD scalable parallel computer (hereinafter called CAP in this paper) [2][3] and found that this problem was significant in the generation of the test programs for prototype machine which was developed as an architecture model for the first time.

It can therefore be said that generating the test programs independently without being affected by hardware configuration of the parallel processing unit will lead to a reduction in the test program development period.

This paper proposes a method for generating the test programs, independent from the hardware configuration of the parallel processing unit, for the scalable SIMD parallel computer. It also reports that application of this method resulted in a reduction not only in the number and types of the test programs but also in the period required for generating the test programs compared to the conventional method.

Chapter 2 describes the test program development procedures for the scalable SIMD parallel computer and remaining problems, and chapter 3 proposes a test program generating method independent from the hardware configuration of the parallel processing unit. Chapter 4 evaluates the application results.

2. Test program generating procedure for scalable SIMD parallel computer and remaining

problems

2.1 Configuration and features of SIMD parallel computer

Typical configuration of the SIMD parallel computer is shown in Fig. 1. As shown here, this computer is composed of a parallel processing unit (AU) and a control unit (CU) that controls the parallel computer as a whole, and AU is further composed of a number of basic processor elements (PE) and a network that interconnects the PEs. A shared memory (SM) is used to store a large quantity of data. This AU features that it can be so configured as to meet the objective and required performance by making the number and connection of PEs variable. Each PE is basically designed to perform data processing operation simultaneously in synchronization with the clock according to the instruction given by CU.

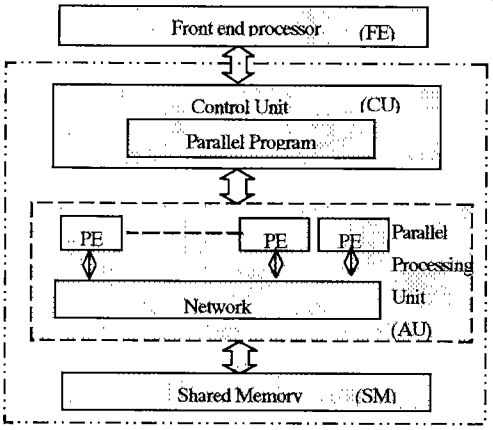


Fig.1 Structure of SIMD parallel computer

The parallel programs and test programs of the SIMD parallel computer are stored in CU, where the parallel program execution sequence control and scalar operation, input/output control, and the control of the array operation executed by AU are carried out.

2.2 Conventional test program generating procedure

The test program to be used for hardware testing of the SIMD parallel computer has been generated by human effort by using machine instruction or high order language, or by using RIT (Random Instruction Testing) where the machine instruction is executed random [4].

The procedures for test program generation and test program execution are shown in Fig. 2. Based on the test specifications, the test programs are generated using the machine instruction or high-level language. A software simulator is then used to debug the programs before they are executed on the actual machine. These test programs are then loaded to the CU of the actual machine together with the test program monitor for controlling test program execution. These programs are then verified by executing them in sequence. The test program operation is shown in Fig. 3. The basic procedure consists of initialization, setting of expected data, setting of input data, execution of test instruction, comparison of execution result and expected data, and judgment for error. If any error is detected, its cause is traced.

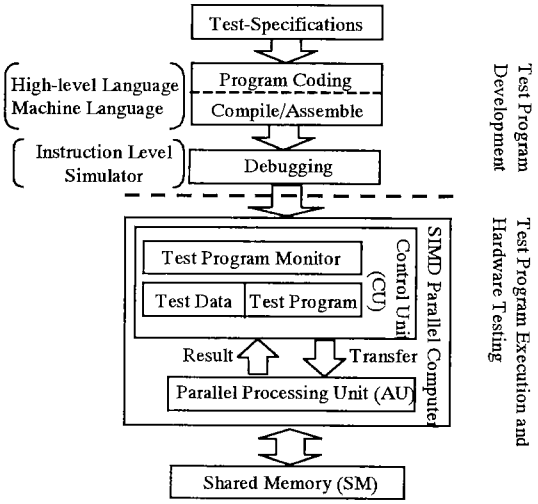


Fig.2 Test program design and test program execution

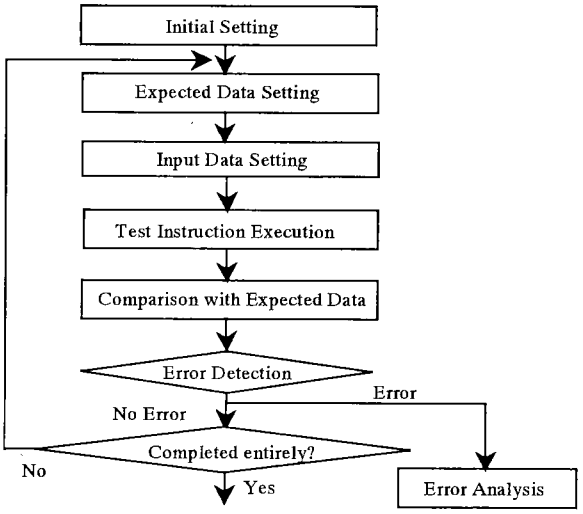


Fig.3 Operational flow of test program

Since the test programs are generated by taking the verification procedures of the objective hardware into consideration, the number of test programs required is obtained by multiplying the number of functions to be tested by the number of hardware configurations.

The types and contents of the machine instructions for a typical SIMD parallel computer classified by functions are shown below:

(1) Array instruction

All PEs perform array operations simultaneously for the array data stored in the register file and local memory in each PE of AU. The array instructions include LOAD-STORE, MOVE, SHIFT, ARITHMETIC-LOGICAL OPERATION, and TRANSFER AMONG PEs.

(2) Scalar instructions

The scalar instructions include LOAD-STORE, MOVE, SHIFT, ARITHMETIC-LOGICAL OPERATION, and BRANCH, which perform parallel program execution sequence control and operation for scalar data.

(3) Array→scalar instruction

This instruction is for converting the array data of all the PEs to the scalar data, and transferring it to CU.

(4) Scalar→array instruction

In contrast to the above, this instruction is for broadcasting the data inside CU commonly to all the PEs.

(5) I/O instruction

This instruction is for data transfer between hardware blocks and outside.

In the SIMD parallel computer, these machine instructions are used to generate test program set classified by functions. Table 1 shows the test program set. Accordingly, hardware testing of the SIMD parallel computer requires the test program sets composed of these five functions, with the maximum number of those test program sets being equal to the number of the hardware configurations of the required AUs.

Table.1 Types of test program

	Test Program	Machine Instruction to be Tested
a)	Interface Test between Hardware Blocks	Input–Output Instruction
b)	Memory Test for CU,LM	Input/Output Instruction
c)	Test of Scalar–Array Data Translation Function between CU &AU	Scalar→Array Instruction Array→Scalar Instruction
d)	Test of Processing Function in AU	Array Instruction
e)	Test of Processing Function in AU involving PE to PE Transfer	Array Instruction

2.3 Problems related to test program generation corresponding to AU hardware configuration

When performing hardware testing of the SIMD parallel computer, the hardware configuration of AU cannot be specified. The reasons are that the SIMD parallel computer itself is able to adopt a scalable configuration with variable AU configuration, and that, at the stage of hardware testing, the objective AU hardware is not always complete. It is also not practical to have enough number of individual test programs generated in advance corresponding to the required hardware configurations. It is necessary to arrange the test program to be used for hardware testing so that it can correspond to any change in the hardware configuration. This point will be further discussed below:

The scalable SIMD parallel computer performs equal operation to all the PEs, hence it is generally considered that, as a general rule, only one test program will do without paying attention to hardware configuration. However, this can apply only to the so-called program code, and still we have to take the test data into consideration. The test data mentioned here refer to both the input data and expected data. If only the identical test data is to be set for all the PEs in executing the test program, then we need not take care of the hardware configuration. However, if we want to improve the quality of testing, it is then necessary to set to each PE different data patterns such as periodic pattern and random pattern as test patterns. In such a case, the hardware configuration must be taken into consideration, and a plural number of test data groups must be prepared corresponding to the hardware configuration.

Besides, testing of array operating function that involves data transfer among PEs will result in an operation over a plural number of PEs within an AU. Hence, the position of data transfer destination PE would vary with the hardware configuration and connection among PEs, and the result of the operation may also vary accordingly. This means that the test data groups must be prepared in the number equal to that of combinations of connections among PEs.

As mentioned above, the test program sets must be generated in the same number as that of the required hardware configurations. This will result in a large increase in the amount of manpower for program generation as well as for program maintenance afterward. A higher test program generating efficiency is therefore being looked for.

Many studies have been conducted concerning automatic test program generation with a view to increase the test program generating efficiency. The results of such studies placing the emphasis on a particular processor and architecture have been reported [5][6][7]. However, reports of the studies on the parallel computer of SIMD type, etc., or reports on the test program generation by considering the difference in hardware configuration are few.

3. Test program generation independent from AU hardware configuration

In this chapter, we propose a method for setting and generating the test data of the test program independent of the hardware configuration, and further, performing a series of operations consecutively up to the transfer of test data and executing the test program so that the test program is adaptable to any arbitrary hardware configuration of AU.

Since the test program a) shown in Table 1 is independent of the AU hardware configuration, only one type of test program will be enough. As the methods for making the automatic generation of the test data possible, we herein propose a method to be applied to test programs b), c), and d), which do not accompany data transfer between PEs, and a method to be applied to test program e) which accompany data transfer between PEs.

(1) The method in which the test program monitor gives instructions to CU to generate the test data

The test program monitor causes CU to generate the test data at the time of test program execution, and also broadcasts the data corresponding to the testing area of AU.

This is applicable to such a test where different data patterns including increment pattern and decrement pattern are inputted as the test data into each memory address or each PE in cases b), c), and d). Different hardware configuration requires the use of different test data. It is also necessary to change the broadcasting range corresponding to the changes in the hardware configuration. In this method, the test data is automatically generated inside CU corresponding to such changes.

Fig. 4 shows how to realize this method. First, the test data generating routine which generates automatically the test data according to the algorithm and corresponding to the hardware configuration, the test data fetching routine for taking out the required number of data patterns, and the transfer range specifying routine for setting the range of transferring by broadcasting are prepared inside the CU. Next, these routines are arranged so that they can be called out from each test program via the test program monitor.

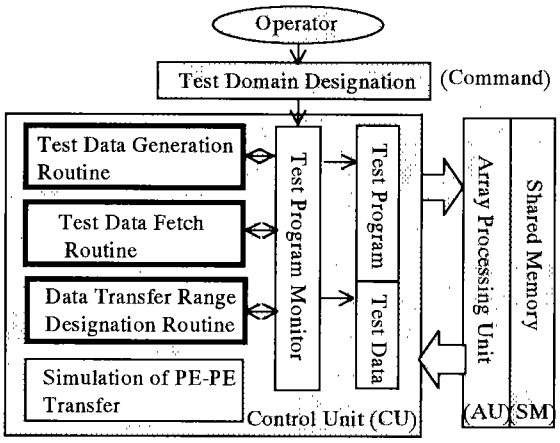


Fig.4 Test data generation and transfer of data

Before executing the test program, the hardware testing personnel or test program operator inputs the value for specifying the AU area to be tested by using commands. The test program then transfers the input value as a parameter to each prepared program routine to execute the routine. By this method, it becomes possible to automatically generate and broadcast the test data that corresponds to the hardware configuration to be tested. After executing the broadcast, test program execution can be started at once under the control of the test program monitor.

(2) The method in which the test program monitor is provided with simulation function

This method uses the test program monitor provided with the simulating function to generate the test data corresponding to the prepared hardware configuration at the time when executing the test program.

This is applied to the case e). Since the position of each PE in AU, that is, the PE address is inseparable from test execution, an appropriate test pattern must be generated each time the hardware configuration changes.

For this purpose, we provided inside CU a PE to PE transfer simulating routine program that can be called from the test program for simulating PE to PE data transfer as shown in Fig. 4.

As an example, a method for realizing a case of SIMD scalable parallel computer whose AU connection network is two-dimensional-torus connection with m x n configuration. Fig. 5 shows its configuration. Fig. 6 shows how data transfer among PEs is operated in such a configuration. If instructions are given to transfer the data within all the PEs into an PE separated by M cells in X-axis direction and by N cells in Y-axis direction, then the position of PE (X, Y) to which the data at PE (x, y) will move after completion of transfer operation can be indicated by the following equation:

$(X, Y) = (X+M, Y+N)$..... if $X+M \leqq m-1$ and $Y+N \leqq n-1$

$=(X+M-m, Y+N-n)$... if $X+M>m-1$ and $Y+N>n-1$

If simulation is executed according to this equation for the input data, and the expected data after execution of the PE to PE transfer instruction is calculated, the test data will be automatically generated without depending upon the hardware configuration. Fig. 7 shows the sequence of the test program execution in this operation. The PE to PE transfer simulation is executed within CU to have the expected data of the test generated, and then the input data is transferred to AU to execute the instructions to be tested. Next, the expected data generated in advance is transferred to AU and compared with the results of executing the instructions to be tested to judge if any error exists. In this case, also, it is possible to perform a series of operations consecutively from test data generation to test program execution.

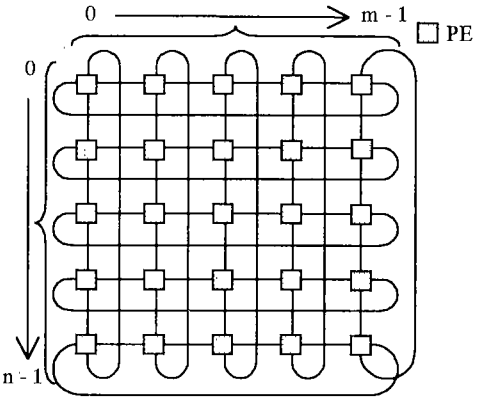
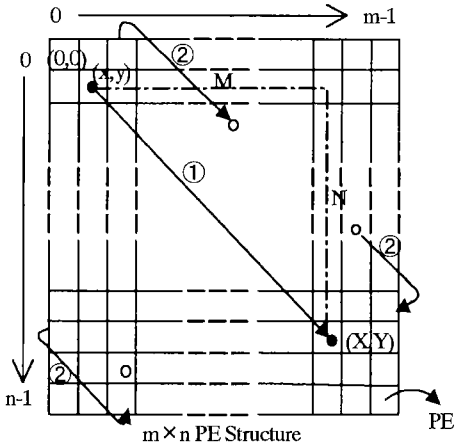
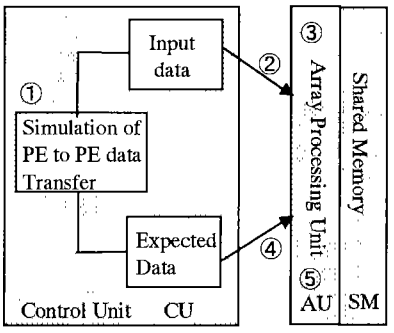


Fig.5 2-dimensional m × n Torus connection of AU



This diagram shows the followingtwo cases.;
①: x+M≦m-1 , y+N≦n-1
②: x+M>m-1 , y+N>n-1
(x,y) ,(X,Y):PE Address (Coordinates)

Fig.6 Data transfer among PEs



①Expected data generation (Simulation)
②Input data transfer
③Execution of instruction to be tested
④Expected data transfer
⑤Comparison of result data with expected data

Fig.7 Test sequence of Array Instruction having PE to PE data Transfer

Similar principle applies also to the machine instructions involving data transfer among PEs. In addition, this idea can be similarly applied not only to the Torus connection, but also to other connecting methods. Therefore, it is sufficient to generate only one type of test program.

4. Evaluation of and consideration on application results

The proposed test program generating system independent from the AU hardware configuration was applied to an actual SIMD scalable parallel computer, and the results were evaluated and examined concerning to what extent the program development efficiency was enhanced. In this study, we performed an evaluation based on the actual results of the manpower and time required for generating the test programs for CAP [8][9] which is a scalable SIMD parallel processor developed by us.

CAP is composed of the parallel processing unit (CAU) and the CAP control unit (GCU) which controls the entire CAP as shown in Fig. 8. This CAP and the shared memory (SM) are connected by the multiplex bus to constitute an SIMD array processor. CAU is further composed of the cellular array unit consisting of a two-dimensional array (m x n PE) of a number of basic processor elements (PE), the processing control unit which controls the cellular array unit, and the data input/output control unit. CAU is variable consisting of m x n elements so that it can be configured to meet the design objects and required performance. CAU and GCU correspond respectively to the typically configured AU and CU of the SIMD parallel computer shown in Fig. 1.

The authors developed two types of machines; a prototype machine and its evaluation machine [8][9] for CAP development. The prototype was an architecture model, and its CAU configuration is maximum 32 x 32 PE. The evaluation machine was developed by correcting the faults of the prototype machine and enlarging its CAU configuration to maximum 64 x 64 PE. The test program generation by the prototype was performed using the conventional method, while the proposed test program generating method independent from the AU configuration was applied to the evaluation machine.

(1) Application results

To show clearly how high the test program generating efficiency has been improved, the test program generating manpower is selected as the evaluation index.

In this evaluation, we assumed a case of four configuration types (8 x 8 PE, 16 x 16 PE, 24 x 24 PE, and 32 x 32 PE), and another case of eight configuration types (8 x 8 PE, 16 x 16 PE, 24 x 24 PE, 32 x 32 PE, 40 x 40 PE, 48 x 48 PE, 56 x 56 PE and 64 x 64 PE), and generated the test programs shown in Table 1 corresponding to the respective configurations. We prepared two types of test data: one for giving the same data to all the PEs or memory addresses, and one for giving different data to each of the PEs or memory addresses. The manpower for generation & debugging of test program was, for the case of test program a), common to both the prototype machine and evaluation machine, and equal to the sum of time required for generating the program code and test data. For the case of test programs b), c), d), and e), the manpower was equal to the case of a) for the prototype machine, but for the evaluation machine, it was the sum of the time for generating the program code and test data routine or PE to PE transfer simulation routine.

Table 2 shows the number of all the test programs generated under the above-mentioned conditions and the manpower required for generating these.

Table.2 Number of test programs and manpower

Test Program \ Number of Programs / Manpower	Prototype 4 types of hardware configuration		Prototype 8 types of hardware configuration		Evaluation Type	
	Number of Programs	Man power (Hour)	Number	Man power	Number	Man power
a) Interface Test between Hardware Blocks	1	250	1	250	1	250
b) Memory Test for CU,LM (Same Data)	1	210	1	210	1	240
(Different Data)	4	1040	8	2080	1	290
c) Test of Scalar-Array Data Translation Function between CU & AU. (Same Data)	1	210	1	210	1	250
(Different Data)	4	1080	8	2160	1	290
d) Test of Processing Function in AU (Same Data)	1	230	1	230	1	280
(Different Data)	4	1120	8	2240	1	310
e) Test of Processing Function in AU involving PE to PE Transfer	4	1200	8	2400	1	650
Total	20	5340	36	9780	8	2560

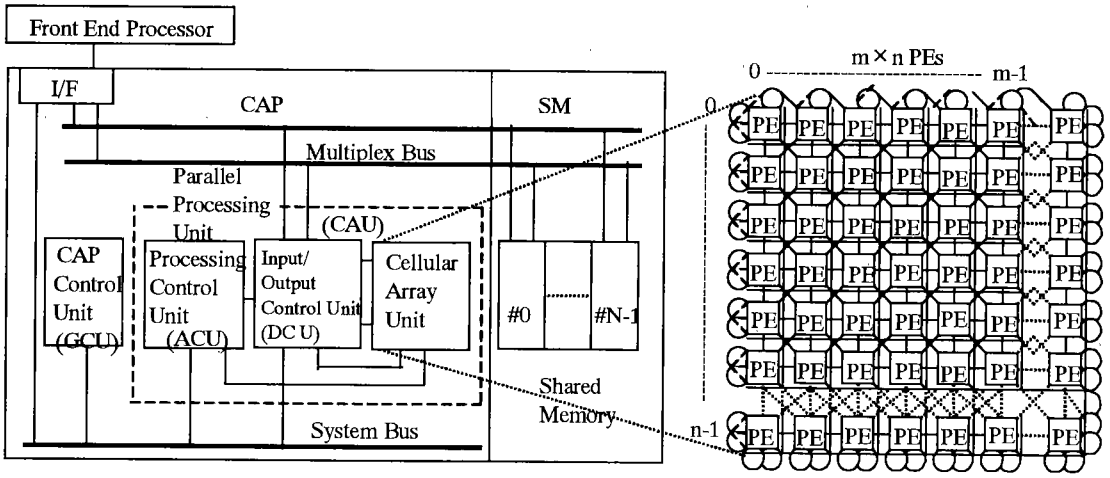


Fig.8 Structure of CAP

(2) Evaluation and consideration

As can be seen from Table 2, by applying the proposed test program generation system independent of the hardware configuration of AU, the number of test programs, compared to that of the prototype machine was reduced down to 2/5 for the case of four hardware configuration types, and 2/9 for the case of eight hardware configuration types, and the generating manpower down to about 1/2 and 1/4 respectively.

This is because that: in the case of the conventional system used for the prototype machine, the test programs must be generated in the number capable of coping with all the hardware configurations and all the test data, whereas in the case of the system proposed for the evaluation machine, generation of the test programs only in the number corresponding to the number of types of the test data is required.

In the evaluation shown in Table 2, when using the same test data for each PE or each memory address in test programs a), b), c), and d), there is no difference in the number of test programs to be generated and the generating manpower between the prototype machine using the conventional system and the evaluating machine using the proposed system. However, in the case of setting different test data to each PE or each memory address in test programs b), c), and d), and also in the case of performing arithmetic operation test accompanied with PE to PE transfer in test program e), it is necessary to generate the test data corresponding to the hardware configurations of AU if such configurations differ. Because of this reason, the prototype machine using the conventional system requires four or eight test programs for each of b), c), d) and e), corresponding to the hardware configurations. On the other hand, for the evaluation machine to which the proposed system was applied, it is necessary to generate only one test program for each. Accordingly, the number of entire test programs required when the proposed system was applied was reduced to 2/5 and 2/9 respectively for four types and eight types of AU hardware configurations, compared to the cases where the conventional system was used.

As for the generating manpower for all the test programs, the conventional system applied to prototype machine required 24 man-months and 40 man-months, for four types and eight types of AU hardware configurations respectively. Contrary, the manpower for the proposed system applied to the evaluation machine was 12 man-months, which means a large reduction to about 1/2 and 1/4 respectively. It is clear that the larger the number of hardware configurations of AU, the greater these ratios will become.

As explained above, by making it possible to generate the test data for the test programs automatically in accordance with AU configuration, it was demonstrated that the efficiency of test program development can be increased largely.

Next, test program maintenance, test results, test program execution time, test program operation and the test coverage are evaluated and considered in turn.

• The burden for maintenance and control of the test programs was reduced corresponding to the reduction in the number of programs and program types.

• The test programs generated by the conventional system and proposed system were executed respectively under the condition where a failure was generated forcibly in the actual machine. As a result, the same error was detected correctly by each program. There is no difference in the execution result of both programs, and the quality of them is equivalent.

• Table 3 shows the test program execution time for the case of test program d). With the conventional system, the execution time means the time when both the program code and test data are set on the main memory of CU which is the scalar processor. The difference in time shown in the table can be regarded as the time for automatic generation of test data, and it became clear that the execution time of the proposed system is longer by approximately 8%. In the case of the conventional system, however, it is necessary to place the test data in the external storage of CU and load the data into the main memory as necessary when executing a plural number of test programs consecutively. The execution time, when this access time to the external storage is taken into consideration, may become longer for the conventional system, and when executing a plural number of test programs in succession, it is expected that the total execution time of the conventional system may be longer.

Table.3 Execution time of test program

Test Program	Conventional System	Proposed System
d) Test of Processing Function in AU	75 sec	81 sec

• As to the operation for executing the test program, the proposed system requires operation for specifying the AU hardware configuration necessary for automatic generation of test data and the types of generating patterns as parameters, and this increases the number of items to be indicated by the operator. However, when executing a plural number of test programs, the conventional system requires loading of test data from the external storage each time a test program is executed. Compared to this, the operation of the proposed system would be easier and save the operator's labor.

• It can generally be considered that the test coverage would improve as the number of combinations of test data increases. The proposed system permits automatic generation of test data conforming to the arbitrary algorithm by using test data generating program and simulation. So, much more test data can be generated easily than the conventional system. This also serves to increase the number of combinations of test data. Accordingly, the proposed system is also effective for improving the test coverage.

5. Conclusion

Both the variety and number of test programs required for the hardware testing of the SIMD parallel computer have increased, requiring a huge amount of time for generating them, because the AU hardware configuration as the specification was variable and the AU hardware was not fixed

at the time of hardware testing. With a view of solving this problem, the authors developed a system for automatically generating the test data for the test program corresponding to the AU hardware configuration, and applied this system to the scalable SIMD parallel computer CAP (evaluation machine). As the result, it was possible to reduce the manpower for generating the test program significantly. This leads to the reduction in both the development period and development cost. Thus, it was shown that this system is quite effective in test program generation for the SIMD parallel computer.

For further improving the program generating efficiency and consolidating the test program functions, we will keep studying the following issues:

- Method for automatic generation of test programs for SIMD parallel computer
- Method for evaluating the test coverage of the hardware testing, and thereby realizing the maximum coverage with minimum test program

(Manuscript received Oct. 30, 2000, revised April 4, 2001)

References

(1) Shinji Tomita: Parallel Computer Engineering, Shoko-do, 73 (1996)

(2) Miyata, Isonishi, Kan, Iwase; Cellular Array Processor for High-speed Image Processing, Society of Electronic Communications, EC84-6, 49/60, 1984

(3) Miyata, Isonishi, Kan, Iwase; scalable Parallel Processor CAP, 31st National Convention of Image Processing and Information Processing Society, 103/104, 2D-9, 1985

(4) J. Willson et al: Challenges and Trends in Processor Design, Computer, Vol. 31, No. 1, 39/50, Jan. 1998

(5) T. Hattori et al: Design Method of a 200MHZ super-scalar microprocessor: SH4, Proceeding of the Design Automation Conference, 246/249, 1998

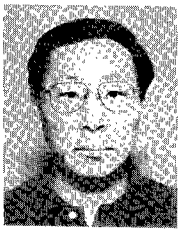
(6) S. Taylor et al: Functional Verification of a Multiple-issue, Out-of-order, Super-scalar Alpha Processor-The DEC Alpha 21164 Microprocessor, Proceeding of the Design Automation Conference, 638/643, 1998

(7) Nakano, Kimura, Nonaka: Test Case Automatic Generating Tool, Mirage, 46th National Convention of Information Processing Society, 5N-5, 1993

(8) Isonishi, Miyata, Iwase: Architecture of Cellular-Array Type Scalable Parallel Processor, Information Processing Society, Computer Architecture Study Group, 88-ARC-73, 61/68, Oct. 1988

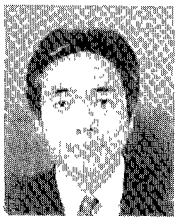
(9) Miyata, Isonishi, Kan, Iwase: Distributed Parallel Processor for Satellite Image Processing (1), Society of Electronic Information Communications, CPSY90-8, 25/32, 1990

Akira Iwase (Member) received the B.E. degree in electronic engineering from Tohoku University in 1965. Since 1965, he has engaged in Mitsubishi Electric Corporation. His research interests include micro-processor and computer architecture.



Tetsuo Isonishi (Non-member) received the B.E. and M.E. degrees from Niigata University in 1981 and 1983. Since 1983, he has engaged in Mitsubishi Electric Corporation. He worked at MIT Media Laboratory as a research affiliate from 1992 through 1994. His research interests include parallel computer architectures, network-based multimedia systems, and multimedia contents. He is a member of the Information Processing Society of Japan.

Hiroyuki Miyata (Non-member) received the B.E, M.E and ph.D. degrees from Kyoto University in 1980, 1982 and 1997 respectively. Since 1982, he has engaged in Mitsubishi Electric Corporation. His research interests include parallel computer architecture and parallel algorithm. He is a member of the Information Processing Society of Japan.



Hisao Koizumi (Member) received B.E. and ph.D degrees from Tohoku University in 1961 and 1993. He worked at Mitsubishi Electric Corporation from 1961 through 1997. Since 1997, he has been a professor at the Department of Computers and Systems Engineering, Tokyo Denki University. His research interests include information system and software engineering. He is a member of the Information Processing Society of Japan.