A Hardware Implementation of PRAM and its Performance Evaluation *

M. Imai, Y. Hayakawa, H. Kawanaka, W. Chen, K. Wada, C.D. Castanho, Y. Okajima, H. Okamoto

Nagoya Institute of Technology, Gokiso, Showa, Nagoya 466-8555, Japan
** E-mail: (imai,zed,sw20,chen,wada,caca,punio,hiroyuki) @phaser.elcom.nitech.ac.jp

1 Introduction

A PRAM (Parallel Random Access Machine)[4] is the parallel computational model most notable for supporting the parallel algorithmic theory. It consists of a number of processors sharing a common memory. The processors communicate by exchanging data through a shared memory cell. Each processor can access any memory cell at one unit of time and all processors operate synchronously under the control of a common clock. These facts make the model a very advantageous platform for considering the inherent parallelism of problems. However,the development of parallel computers which fit this model has not quite matched the theoretical requests. The researches focusing on the reduction of this gap has been carried out [2, 5, 6, 7, 8, 9]. However, most of them give only theoretical analysis; the implementation of the PRAM on hardware level is seldom seen.

Placing emphasis on the realization of the shared memory, we provide several approaches to implement the PRAM model on hardware level. PRAM algorithms can be executed directly and efficiently in these computers without essential modifications. Variants of the PRAM model differ in their handling of simultaneous reading and writing of the same memory cell. In this paper, our object of study is the CRCW PRAM[4] which is the most powerful variant of the PRAM that allows concurrent reading and concurrent writing to the same cell. In the case of concurrent writing to the same cell, there are several methods to determine the content of writing. Here, our implementation is available for all methods. We present three kinds of PRAM-like computers whose shared memory access mechanisms based on tree structure, tree-bus structure, and bus structure, respectively. We realize the shared memory mechanisms and the synchronous mechanism on hardware level by using VHDL(VHSIC Hardware Description Language)[1] and evaluate their performance. The results obtained from the simulations have shown that our approaches can be realized with a practical amount of hardware, and that the running time decreases in proportional to the amount of hardware.

2 Design of the PRAM-like Computers

2.1 Steps of the PRAM Model and Our PRAM-like Computers

In the PRAM each processor has its own local memory and holds its own ID. All the processors operate synchronously in parallel. Usually, one step of the

* This work is supported by the Grant-in-Aid for Scientific Research (B)(2) 10205209 (1999) from the Ministry of Education, Science, Sports and Culture of Japan.

** Corresponding e-mail address is: wada@elcom.nitech.ac.jp

PRAM model is considered as one instruction of PRAM algorithms, despite containing multiple access to the shared memory. To avoid this ambiguity, we define one step of our PRAM-like computers, called *PRAM step*, as executing only one shared memory access, or arithmetic and logical operations performed in their local memory. The period of one PRAM step lays between two adjacent synchronous events. Thus, an instruction of a PRAM algorithm such as $a(i) := a(i) + b(i)$ would be changed into four PRAM steps: reading $a(i)$ and reading $b(i)$ from the shared memory, evaluating $a(i) + b(i)$, and writing the value to the shared memory cell used for $a(i)$. Obviously, an instruction of a PRAM algorithm can be easily changed into a sequence of PRAM steps without any essential modification.

2.2 Architecture of the PRAM-like Computers

In the PRAM model, let P denote the number of processors and C denote the number of the shared memory cells. The PRAM parallel computer we design consists of N processor units, M memory units, and one synchronous processing unit (Fig. 1(a)), where each processor unit undertakes P/N PRAM processors, each memory unit undertakes C/M PRAM shared memory cells, and the synchronous processing unit takes the responsibility for synchronous processing. The processor units perform arithmetic and logical operations locally, and the shared memory accesses take place through the communication between the processor units and the memory units. At the end of every PRAM step, the synchronous processing proceeds in the order of the following three steps: (1) Each processor unit transmits a PRAM step termination signal to all the memory units. (2) Each memory unit prepares the data to execute the next PRAM step, and sends the memory update termination signal to the synchronous processing unit. (3) When the termination signal has been received from all memory units, the synchronous processing unit sends the synchronous processing termination signal to all the processor units.

The construction of the above units is based on tree structure which facilitates the tasks of sending messages to or receiving messages from multiple units.

(1) Structure of the Processor Units A processor unit consists of $M - 1$ nodes. The leaves are connected with the M memory units such that the processor unit can communicate with each memory unit (see Fig.1 (b)). There are two kinds of nodes: the root and internal nodes. As mentioned above, one processor unit undertakes the work of P/N processors of the PRAM model. The processor-root executes the arithmetic and logical operations, sends memory accessing requests, receives the results from memory reading requests, and sends the synchronous processing request. The processor-nodes route the memory accessing requests to the desired memory unit according to the cell ID, and pass the results of memory reading requests to the processor-root.

(2) Structure of the Memory Units We design three different kinds of memory units. *(i)Tree model:* It consists of $N - 1$ nodes which form a binary tree. The leaves are connected with the N processor units such that the memory unit can treat the access requests from each processor unit (Fig. 1(c)(i)). There are three different kinds of nodes: the leaves, the root and the internal nodes which are denoted as *memory-leaves (ML)*, *memory-root (MR)* and *memory-nodes (MN)*, respectively. As stated above, one memory unit corresponds to C/M shared memory cells of the PRAM model. Each memory-leave holds the same data, that is, it holds the data corresponding to C/M PRAM shared memory

cells. Therefore, a memory-leaf can immediately return the result when receiving a memory reading request. The memory-leaves and memory-nodes undertake the work of resolving the competition of concurrent writing requests which have been sent from its children. When receiving a writing request, the memory-root sends the update information to all the memory-leaves since all of them store the same data. *(ii)Tree-bus Model:* This model uses a bus to transmit the updated information directly from the memory-root to the memory-leaves without passing through the memory-nodes (Fig. 1(c)(ii)). *(iii) Bus Model:* The memory unit of the bus model has the same structure of that in the tree-bus model except that the memory-root, memory-nodes, the edges between them, and the edges from them to memory-leaves are removed. Here, the bus is used for updating the data of the memory-leaves (Fig. 1(c)(iii)).

(3) Structure of the Synchronous Processing Unit The synchronous processing unit consists of $M-1$ nodes which form a binary tree. Besides, its leaves are connected with the M memory units, and its root is connected with the N processor units by a bus which is used to send the synchronous processing termination signal.

2.3 Memory Accessing and Synchronous Processing

In one PRAM step, the processing is accomplished in the following order: the shared memory access, the arithmetic and logical operations, and the synchronous processing. In order to decrease the number of the requests, each processor unit executes a preprocessing for all the memory access requests.

Memory Accessing: (1) The preprocessing in each processor unit: (a) Change the sequence of access requests such that all the reading requests are listed before the writing requests. (b) (i) Compress the requests which access the same cell into one request by resolving the competition of concurrent accessing, and (ii) compress the reading requests which access the contiguous memory cells into one request such that it holds the information of the number of the contiguous cells and also the smallest address of the cells. The rates of compressibility are different from each algorithm, therefore, considering a trade-off between the preprocessing time and the memory accessing time, we may not execute (b). (c) In order to avoid too many processor units accessing the same memory unit at the same time, rearrange the order of access requests as follows: assuming that the processor ID is i $(0 \leq i \leq N)$, revise the kth request into the $((k+i-1)\ mod(M))$th request. (2) Processing flow between processor and memory units: The processor units send access requests to and receive the results of the accesses from the memory units. The memory units process the access requests from the processor units and send the access results to them. After finishing the access requests of one PRAM step, the processor units send the access request termination signal to all the memory units.

Synchronous Processing: After receiving the access request termination signal from all processor units, each memory unit updates the cells of the writing access requests, and prepares to process the next PRAM step. Then they send the synchronous processing request to the synchronous processing unit. After receiving the synchronous processing request from all the memory units, the synchronous processing unit sends the command to start the next PRAM step.

2.4 The Internal Processing in the Nodes of the Units

The nodes of each kind of units consist of basic elements which are: the sending buffer, the receiving buffer, and the processing part. The processing part executes

the processes like routing, and eliminating the competition of concurrent access requests, which are different for each kind of node. Each node communicates with its two children and one parent, however, since the communication between nodes takes more time than the internal processing, in the implementation we allow one node to communicate with two children in parallel.

(1) The Internal Processing in the Nodes of the Processor Unit The internal structures of the processor-root and processor-nodes are shown in Fig. 1(d), where PR-S and PR-R are the processing parts which correspond to the sending and receiving parts. The processor-root has a special part called CPU which undertakes the work of its corresponding P/N processors of the PRAM model.

(2) The Internal Processing in the Nodes of the Memory Unit *(i)Tree Model:* We let TL, TN and TR represent the leaves, internal nodes and the root of the tree. Each of TL-S, TN-S and TR-S executes the process of eliminating the competition of concurrent writing requests, therefore, they have their own memory in order to check the sequences writing requests. One memory unit corresponds to C/M memory cells of the PRAM model. When receiving a reading request, TL-R immediately returns the data to corresponding processor unit. Therefore, each TL-R holds the data of the C/M PRAM memory cells. The other nodes are used for routing (Fig. 1(e)). *(ii)Tree-Bus Model:* It has the same structure as the tree model memory unit, except that a bus is used to transmit the update information directly from the memory-root to memory-leaves. *(iii)Bus Model:* In this case, each node has the same structure shown as (Fig.1(f)), where a part called BN-PROC has a memory used for recording the history of writing requests and a memory used for the shared memory which corresponds to the C/M memory cells of the PRAM model.

(3)The Internal Processing in the Nodes of the Synchronous Processing Unit Each node outputs a signal after receiving the signals from two inputs, therefore, it consists of an AND-gate.

Readers can find the internal processing algorithms of each kind of the nodes and more details in [3].

2.5 Amount of Hardware and Theoretical Processing Time

The processor-root of a processor unit and the memory-leaves of a memory unit perform much more work than the other nodes, therefore, they are considered to take the same amount of hardware as an usual physical processor. The other nodes are mainly used for routing, therefore we call them routing nodes. We treat them differently. We show the required amount of hardware and the theoretical processing time for the memory in Fig 1(g).

3 Evaluation of the Implementation Method

We developed a simulator which measures the time of shared memory accessing and has the following input, output and parameters: [Input:] algorithms of the PRAM model, the number of shared memory cells (C) and the number of processors (P) which are necessary for the algorithms in the PRAM model, [Output:] the time for the shared memory accessing when the algorithms are executed in the PRAM-like computers, and [Parameter:] number of memory units (M) and number of processor units (N).

The PRAM algorithms we use for evaluation are prefix sum, list ranking, and matrix multiplication. This simulator is composed of two parts. One generates

all the memory access patterns caused by each PRAM step which is determined by the input algorithm, and then preprocesses the access requests to decrease the number of the access requests. The other part evaluates the processing time of memory access requests derived by the first part. We used VHDL for devising the simulator. VHDL is a hardware description language used to describe the hardware behavior, as well as to design, analyze and simulate complex digital systems [1]. The part of the simulator that generates the access pattern was entirely written in C++. The following is a summary of our experiment of result.

(1) The effect of the number of processor units: Whether there is a compression of access requests or not, the memory access processing time is almost inverse proportional to the number of processor units. Especially, when x = 2, 4, 8, 16, 32, 64 and 128, the rate of speed-up are 1.25, 1.5, 2, 3.1, 6.9, 13.7, 24.4, respectively. Therefore, we have obtained an ideal effect.

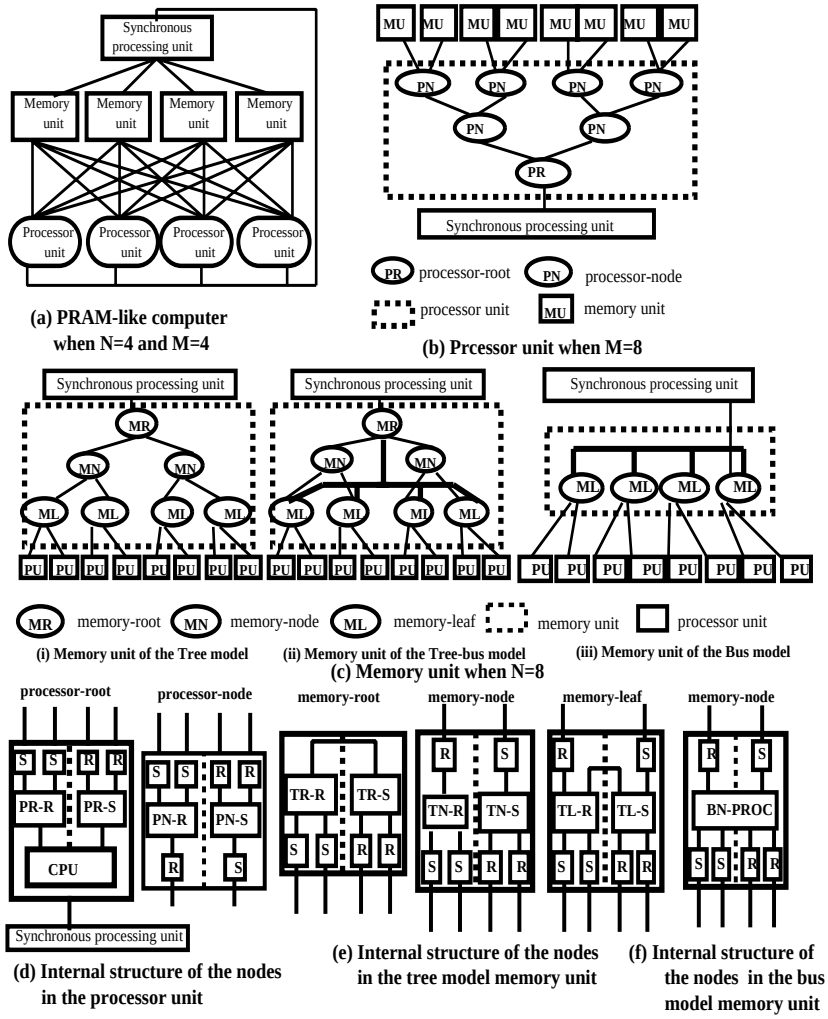
(2) The effect of the number of memory units: The effect of the number of memory units is very small. In the algorithms we have used for evaluation, the frequency of memory reading accesses is larger than that of memory writing accesses. This is the reason why the effect of the number of memory units is small in our experiment. We believe that regardless the number of processor units, a small number of memory units is sufficient.

(3) The effect of compression of access requests: The effect of the compression of access requests is very small for the list ranking and the prefix sum algorithms if the number of processor units is large; but it is large for the matrix multiplication algorithm. The reason is because the matrix multiplication algorithm performs many reading requests to contiguous memory cells.

(4) The comparison of memory models: Tree model and tree-bus model have almost the same memory access processing time, thus, tree-bus model is better because the structures of the memory-root and the memory-nodes are simpler. The processing time of the bus model is the same as the other two models when the algorithms have few memory writing accesses, however for those ones that perform frequent writing accesses bus model showed to be inefficient.

References

1. K. C. Chang: "Digital Design and Modeling with VHDL and Synthesis", IEEE Computer Society Press (1996).
2. D. Culler, et al: "Towards a realistic model of parallel computation", in Proc.4th ACM SIGPLAN Sym.on PPPP (1993).
3. M. Imai, Y. Hayakawa, H. Kawanaka, W. Chen, K. Wada:" Design and implementation of PRAM on Hardware Level", Technical report TR-99-01 in Wada Lab. of Dept. of Elec. & Comput. Eng., Nagoya Institute of Technology, Japan (1999).
4. Joseph JáJá: "An Introduction to Parallel Algorithms", Addison-Wesley (1992).
5. V. Leppänen, M. Penttonen: "Work-Optimal Simulation of PRAM Models on Meshes", Nordic Journal on Computing, 2(1):51-69(1995).
6. F. Luccio, A. Pietracaprina, G. Pucci: "A Probabilistic Simulation of PRAMs on a Bounded Degree Networks", Information Processing Letters, 28(3):141-147 (1988).
7. F. Luccio, A. Pietracaprina, G. Pucciappanen: "A New Scheme for the Deterministic Simulation of PRAMs in VLSI", Algorithmica, 5(4):529-544 (1990).
8. K.Sato, T. Kurozawa, K. Honda, K. Nakano, T. Hayashi: "Implementing the PRAM Algorithms in the Multithread Architecture and Evaluating the Performance", IPSJ AL, 61(6):39-46 (1988).
9. L. G. Valiant: "A bridging model for parallel computation", CACM 33,8 (1990).



Model	Amount of hardware		Theoretical processing time for memory accessing		
	Number of necessary physical processors	Number of necessary routing nodes	Reading access (in all situations)	Writing access	
				All the processor unitswriting into the same cell	All the processor units writing into different cells
T-Model	MN/2 + N	3MN/2 - 2N - 1	O(2logM+(P/N-1)/2)	O(logM+logN)	O(logM+logN+P/M-1)
TB-Model	MN/2 + N	3MN/2 - 2N - 1	O(2logM+(P/N-1)/2)	O(logM+logN)	O(logM+logN+P/M-1)
B-Model	MN/2 + N	MN + M - N - 1	O(2logM+(P/N-1)/2)	O(logM+P-1)	O(logM+P/M-1)

(g) Amount of hardware and theoretical processing time for memory accessing

Fig. 1. PRAM-like computer