TM-0140

Current Research Status of PIM: Parallel Inference Machine

by
A. Goto and S. Uchida

November, 1985

Current Research Status of PIM: Parallel Inference Machine

Atsuhiro GOTO, Shun-ichi UCHIDA

Fourth Research Laboratory
ICOT Research Center
Institute for New Generation Computer Technology
Tokyo, JAPAN

Abstract *The parallel inference machine (PIM) is the most important hardware research target of the FGCS project. The initial stage mainly aimed to conduct R&D of individual component technologies by studying parallel inference mechanisms from various standpoints. Three basic mechanisms for PIM were studied by software simulators and by developing experimental machines with 8 to 16 modules: the reduction mechanism, the data flow mechanism and the kabu-wake method. PIM R&D in the initial stage showed the interesting structure and characteristics of PIM. It also clarified many problems to develop more practical experimental systems. In the intermediate stage, both parallel hardware mechanisms and parallel software system will be studied based on unique and tentative philosophy. Component hardware modules will be developed with the accumulation of implementation techniques such as appropriate hardware building blocks and common software tools. Realistic software research environment will be provided by connecting PSIs to encourage kernel language implementation and parallel operating system development. Efforts to integrate them into a total PIM system will start around the middle of the intermediate stage.*

1. INTRODUCTION

The parallel inference machine (PIM) is the most important hardware research target of the FGCS project. Its ultimate aim is to develop a machine enabling the execution of parallel inferences, the central concept of the fifth-generation computer. This report describes the research and development (R&D) overview of PIM in the initial stage[1], its current status, and tentative plans for the intermediate stage.

2. INITIAL STAGE

The initial stage mainly aimed to conduct R&D of individual component technologies to establish the base for the PIM hardware architecture to be built in the intermediate stage. (see Figure 1)

2.1. Research Themes in the Initial Stage

(1) Analysis of kernel language characteristics

The first important issue is to analyze the behavior of logic programs precisely. Thus, PIM R&D began with static and dynamic analysis of several sample programs written in Prolog or Concurrent Prolog [2]. Prolog was selected to describe don't-know-nondeterminism (OR parallelism) and Concurrent Prolog to describe concurrent processes or stream processing (AND parallelism). Both languages were used as examination bases for PIM in the initial stage. The results of this analysis were incorporated in the architectural design of PIM.

(2) Approaching basic mechanisms of PIM from various standpoints

At first, R&D of PIM architectures included many unsolved problems. So the PIM project began by studying various mechanisms of parallel inference from various standpoints. These approaches were evaluated by software simulation. Three of them were also tested by developing experimental machines with 8 to 16 modules.

They are summarized as follows.

Approach	Mechanism	Experimental system
Basic inference mechanism	Reduction	PIM-R
Basic hardware mechanism	Data Flow	PIM-D
Architectural support for load allocation	Kabu-wake method	Kabu-wake system

2.2. Reduction Mechanism (PIM-R)

Logic programs written in Prolog or Concurrent Prolog generate several pieces of resolvent from a body goal in a clause. This can be regarded as a process in which a goal modifies itself using a clause as a rule. The reduction mechanism can also be viewed as a kind of self-modification. Considered in top-down manner as above, the reduction mechanism can be used as a basis for PIM [3].

(1) PIM-R architecture [4]

The conceptual configuration is shown in Figure 2. The PIM-R consists of two types of modules, inference modules and structure memory modules, with networks connecting them.

An Inference Module (IM) consists of a Process Pool Unit (PPU) and a Unification Unit (UU), as shown in Figure 3. A Process Pool Unit stores and manages reducible processes. A reducible process is sent to a Unification Unit, unified with an appropriate clause and generates reducible goals. Then the results are returned to the the Process Pool Unit. The PIM-R executes Prolog programs in OR parallel and Concurrent Prolog programs in AND parallel.

(2) PIM-R basic software simulator

A basic software simulator was developed to confirm the fundamental validity of PIM-R mechanisms; written in Prolog/C-Prolog, it runs on DEC2060 or VAX-11.

(3) PIM-R detailed software simulation

A detailed software simulator was developed using Occam. This simulator precisely reflects the detailed structure of PIM-R, such as internal data formats. It also handles more than 64 IMs.

(4) PIM-R experimental machine

The PIM-R experimental system was built to examine the reduction mechanism. The hardware configuration is shown in Figure 4. This system consists of 8 PEs (m68000 boards) connected by a common bus with a shared memory. The shared memory works as various connection networks to be tested.

2.3. Data Flow Mechanism (PIM-D)

In the data flow concept, each execution of instructions starts when all necessary data become ready, resulting in parallelism regardless of whether it is explicitly indicated in the program. Therefore the data flow mechanism is expected to be a low level parallel hardware mechanism in PIM [5][6][7].

The PIM project selected the data flow mechanism as a candidate for approaching the parallel inference machine that capable of exploiting the parallelism in logic programs naturally.

(1) PIM-D architecture

The PIM-D executes logic programs in a goal-driven manner: the execution of a clause is initiated when a goal is given and it returns the results (solutions) to the goal. In this execution, the PIM-D can exploit OR and AND parallelism as well as a low level parallelism in unification.

The machine is constructed from multiple processing element modules (PEMs) and multiple structure memory modules (SMMs) interconnected by networks as shown in Figure 5. Each PEM consists of several APUs as execution units and an ICU, a data-driven mechanism.

(2) PIM-D Software Simulation

A software simulator was developed in C on the VAX to confirm the detailed structure of PIM-D. A Prolog or Concurrent Prolog program is compiled into a data flow code, as shown in Figure 6, and runs on this simulator as well as on the following experimental machine.

(3) PIM-D Hardware Simulator

The hardware simulator of PIM-D was developed and is now being finally checked. This machine consists of 8 PEMs and 8 SMMs, connected by a hierarchical bus network, as shown in Figure 7 and Figure 8. Each ICU, APU, and SMM is made of bit-sliced microprogrammable processors (Am2900 series) and TTL ICs.

2.4. Kabu-wake Method

It is an important problem to examine how to divide a job, or how to distribute each piece of a job among PEs. The kabu-wake method is one of the hardware supported mechanisms for job division and allocation in the multi-inference processor environment. The kabu-wake method uses an effective job allocation mechanism for getting all solutions in a large tree search [8].

(1) The Kabu-wake method

In the kabu-wake method, each inference processor, having a job, searches solutions in a depth first manner. On the other hand, idle processors issue requests for jobs to the busy processors. If one processor requests a job from another processor, it splits up its own job and passes search of the remaining branches of the tree to the other processor, so that they perform OR-parallel inference. This execution feature is expected to minimize job allocation overhead among inference processors.

(2) Experimental system

The experimental system was built to test the effectiveness of the kabu-wake method quantitatively. The hardware configuration is shown in Figure 9. The system consists of 16 PEs (one PE for input/output), connected by two kinds of exclusive networks; CONT-network and DATA-network. The CONT-network is a ring network for job requesting packets. The DATA-network is a high throughput switching network for transferring a split job (kabu).

2.5. Conclusion of PIM R&D in the Initial Stage

The software simulation results for the PIM-R and the PIM-D are partially shown in Figure 10 and Figure 11. These results show that both the PIM-R and the PIM-D can extract the parallelism in Prolog and Concurrent Prolog, and that execution speed can be increased nearly in proportion to the number of processors with up to or more than 64 PEs in the system.

The 4-queens program, shown in Figure 10, increases in performance as the number of processing units increases; the performance hits a ceiling around six or seven units. This roughly corresponds to the average level of OR parallelism, which is 6.2, as found in the dynamic analysis.

The quicksort program (written in Concurrent Prolog and containing 10 elements) increases in performance as the number of processing units increases; the performance peaks at around six and seven units. Quicksort has a parallelism level of about seven. In other words, logic programs can be executed in parallel if the programs naturally have parallelism.

Evaluation results of the kabu-wake experiment system are shown in Figure 12 and Figure 13. This indicates the kabu-wake method is effective for finding all solutions in a large search tree. In particular, the overhead for parallel processing in PEs is low, so the amount of communication between PEs is relatively low, as Figure 13 shows.

As described above, PIM R&D in the initial stage showed the PIM structure and its characteristics. The quantitative evaluation of the PIM hardware simulators is now in progress. However the absolute performance of each processing element is relative low. In other words, it is the time to start developing next version experimental machines with realistic processing speed.

3. INTERMEDIATE STAGE

PIM R&D in the intermediate stage, starting this year, will be performed according to the following unique and tentative plans.

3.1. Basic Philosophy and Research Subjects

PIM R&D in the initial stage has clarified many problems to develop more practical experimental systems.

In the design of a total system architecture, functions of the hardware part and the software part of the system must be efficiently divided so that the hardware part can be more optimized, and thus simpler and faster. This requires the study on the software system for static and dynamic resource allocation, parallel job monitoring, and also the implementation of parallel language interpreter for KL1 (Kernel Language version 1). Thus, the intermediate stage plan includes these software research aiming at the development of parallel operating system (PIMOS). To encourage this research activity, the development of multi-PSI system is planed to provide software researchers with more realistic research environment.

In the design of component hardware modules, accumulation of implementation techniques must be more emphasized to build faster and smaller hardware components and also to reduce labor for hardware debugging and maintenance. Large scale PIM systems require stable and easy-to-handle hardware elements with software tools for simulator and debugging tools. Thus, the intermediate plan includes an effort to find out appropriate hardware building blocks and common software tools to make the hardware developments a little more comfortable.

(1) Building Blocks for PIM

The basic inference mechanisms were studied from both top-down approach (PIM-R) and bottom-up approach (PIM-D). These studies revealed clearly that commonly used hardware elements should be developed to enhance and to ease the R&D of PIM. Multi-port memory, packet send/receive hardware modules, and tag handling hardware are some of these commonly used elements.

(2) Developing tools for PIM

Research environment plays an important role in PIM R&D. Researchers usually develop their own tools, such as software simulators, for their own objects. Although these tools are similar functions and structures, these tools are not always inherited among researchers. PIM R&D will study several alternatives. Therefore it seems valuable to develop commonly used software tools as developing base.

(3) Granularity of parallel processing

Initial stage PIM R&D pursued the parallel processing in which rather small sized granules are handled. In the intermediate stage, larger sized granules should be also treated in cooperation with the modularity of programs.

(4) Static and Dynamic resource allocation

The behavior of AI application programs is expected to be dynamic. Therefore, load balancing among the processing elements, and program code allocation are important. The kabu-wake method was studied as a resource allocation mechanism specialized for OR-parallel tree search. However, it is necessary to extend it for more general and complicated inferences.

In addition to the above, the following subjects related to the parallel programming must be solved.

(5) How can programmers write large parallel programs ?

(6) How can programs be tested and debugged ?

3.2. PIM R&D Targets for the Intermediate Stage

In the intermediate stage, both hardware architecture and software systems will be developed for the PIM. The research subjects in the intermediate stage are summarized below.

- Pursuing Large Scale PIM Architecture

(1) R&D on system-oriented PIM architecture

Inter-PEs parallel processing mechanism, especially, a highly parallel connection network and its control mechanism, will be studied in cooperation with PIMOS R&D. The intermediate research goal is an experimental machine consisting of about 100 PEs on which PIMOS will run.

(2) High performance elementary processor for PIM

High performance elementary processors for PIM will be developed by integrating the data flow mechanism and the reduction mechanism. First we will begin by developing hardware building blocks. Next, several experimental processors will be designed in detail, pursuing low level parallelism.

- Pursuing PIM Software System

(3) R&D of software development pilot machines

In order to study parallel software systems for PIM, the following workbenches will be developed in an early stage. They are called multi-PSI systems. The PSI is a personal inference machine developed by ICOT. Therefore it is expected to be a prime candidate processor for parallel software development systems at ICOT.

- Pseudo-multi PSI : a simulator on a PSI machine,
- multi PSI v.1 : 4 - 6 PSI's system,
- multi PSI v.2 : 16-20 new PSI's system,
  - interconnected by 2-D array network.

PIM applications as well as the following language system and operating system will be developed first on these workbenches step by step. Then they will be integrated on PIM.

(4) R&D of PIM language systems

Kernel language systems for PIM will be hierarchically developed extending GHC [9]. GHC is a logic programming language enabling parallel programming. The high level language for system programming, called KL1-u, will have parallel object concepts for system programming. The PIM kernel language, called KL1-c(p), will be a machine independent low-level language, enabling pragmatic control.

(5) R&D of operating system for PIM (PIMOS)

PIMOS, the operating system for PIM, will be developed to facilitate resource allocation and management from the software. First, it will be tried to describe stream-based input/output facilities and goal scheduling based on the locality of multi PSI system configuration.

4. CONCLUSION

This report is a research and development overview of PIM in the initial stage, the current status, and tentative plans for the intermediate stage. In the initial stage, three basic mechanisms for PIM were studied with software simulators and experimental machines. In the intermediate stage, we will study from both parallel hardware mechanisms and parallel software system. Efforts to integrate them into a total PIM system will start around the middle of the intermediate stage.

5. REFERENCES

[1] Murakami, K., Kakuta, T., Onai, R., and Ito, N., "Research on Parallel Machine Architecture for Fifth-Generation Computer Systems", IEEE Computer, Vol.18, No.6, June 1985.

[2] Onai, R., Shimizu, H., Masuda, K. and Aso, M., "Analysis of Sequential Prolog Programs", TR-048, ICOT, May 1984.

[3] Onai, R., Aso, M., Shimizu, H., Masuda, M. and Matsumoto, A., "Architecture of a Reduction-Based Parallel inference Machine:PIM-R", New Generation Computing, Vol.3, No.2, pp.197-228, June 1985.

[4] Onai, R., Shimizu, H., Masuda, M., Matsumoto, A. and Aso, M., "Architecture and Evaluation of a Reduction-Based Parallel Inference Machine: PIM-R" Lecture Note in Computer Science (to appear).

[5] Ito, N. and Masuda, K., "Parallel Inference Machine Based on the Data Flow Model", Proceedings of International Workshop on High-Level Computer Architecture, Los Angels, pp.10, May 1984.

[6] Ito, N., Shimizu, H., Kishi, A., Kuno, E., Rokusawa, K., "Data-flow Based Execution Mechanisms of Parallel and Concurrent Prolog", New Generation Computing, Vol.3,No.1, pp.15-41, Feb. 1985.

[7] Ito, N., Kishi, A., Kuno, E. and Rokusawa, K., "The Dataflow-Based Parallel Inference Machine to Support Two Basic Languages in KL1", IFIP TC-10 Working Conference on Fifth Generation Computer Architecture, July 1985.

[8] Sohma, Y., Satoh, K., Kumon, K., Masuzawa, H. and Itashiki, A., "A New Parallel Inference Mechanism Based on Sequential Processing", IFIP TC-10 Working Conference on Fifth Generation Computer Architecture, July 1985.

[9] Ueda, K., "Guarded Horn Clauses," TR-103, Institute for New Generation Technology, Tokyo, Japan, 1985. (Also in this Workshop)

```
                         :  1982      :      1983      :      1984      :      1985
-------------------------------------------------------------------------------------
                         :
Analysis of              :          Analysis
 kernel languages        :  ------------------------->
                         :
                         :
Data Flow mechanism      :  Basic Design
                         :          Software Simulation ---->
      PIM-D              :                  Hardware Simulator Development ------>
                         :                                          & Evaluation
                         :
Reduction mechanism      :  Basic Design
                         :          Software Simulation (V.1)
                         :                  Software Simulation (V.2)
      PIM-R              :                          Experimental System --->
                         :                          Development and Evaluation
                         :
Kabu-wake method         :          Basic Design
                         :                  Experimental System ----------->
                         :                          V.1             V.2
                         :
-------------------------------------------------------------------------------------
```

Figure 1 The Overview of PIM R&D in the initial stage

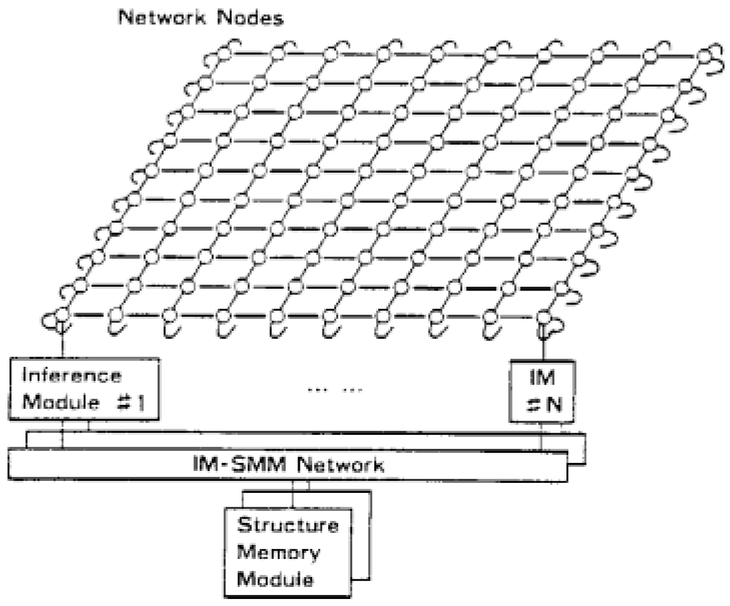


Figure 2 A Conceptual Configuration of the PIM-R

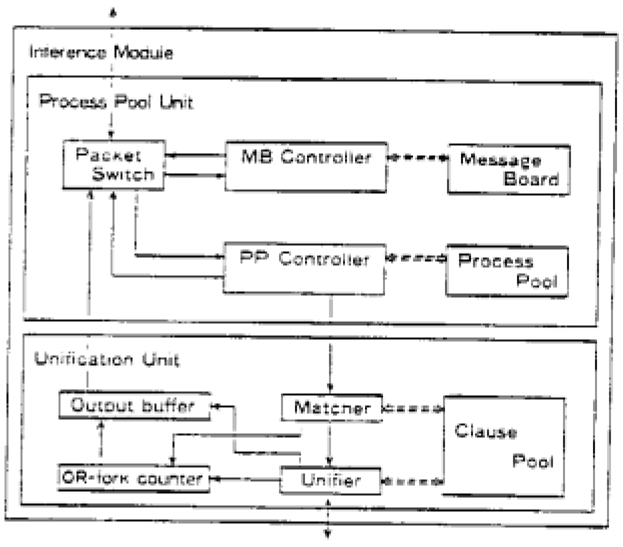


Figure 3 An Inference Module Configuration of the PIM-R

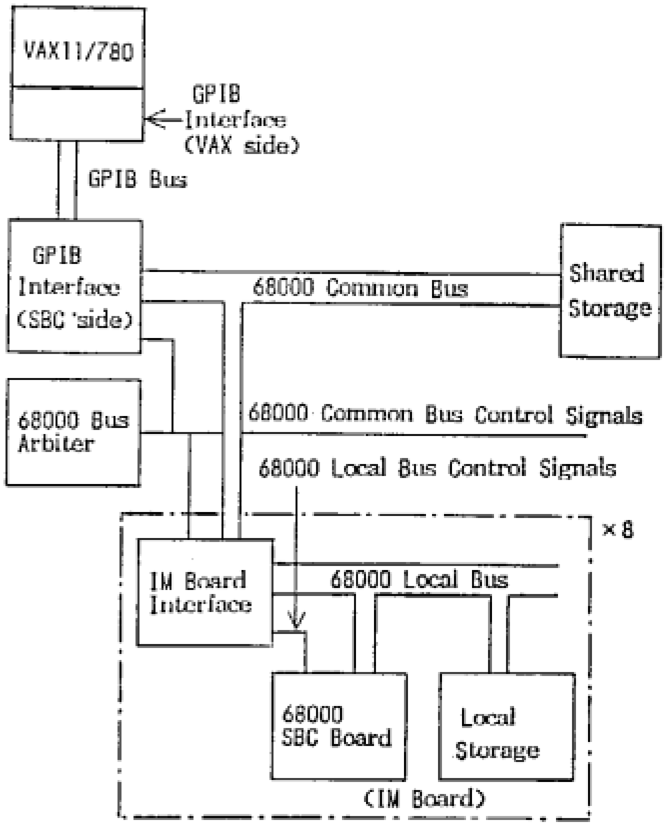
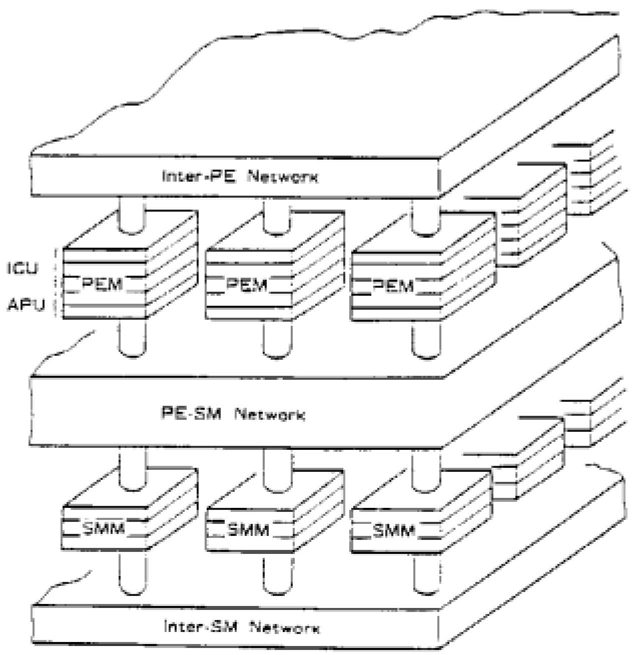


Figure 4 The PIM-R Hardware Simulator Configuration



PEM : Processing Element Module
SMM : Structure Memory Module
ICU : Instruction Control Unit
APU : Atomic Processing Unit

Figure 5 A Conceptual Configuration of the PIM-D

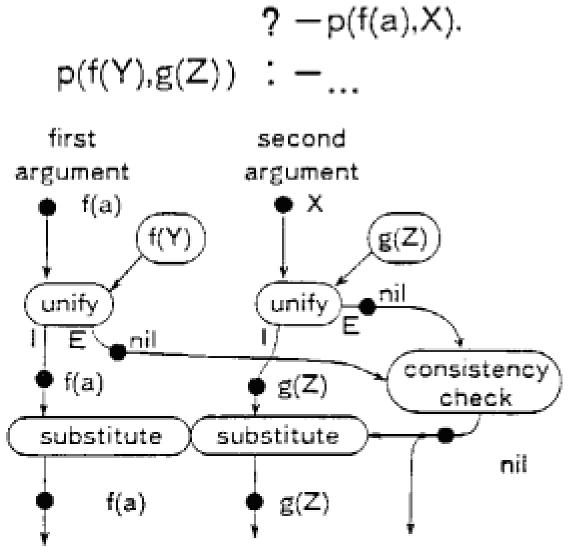
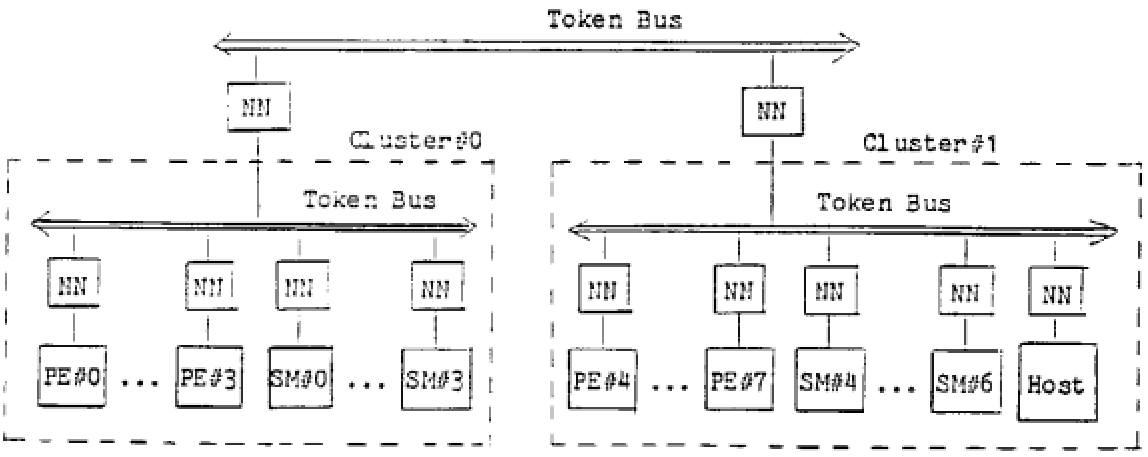
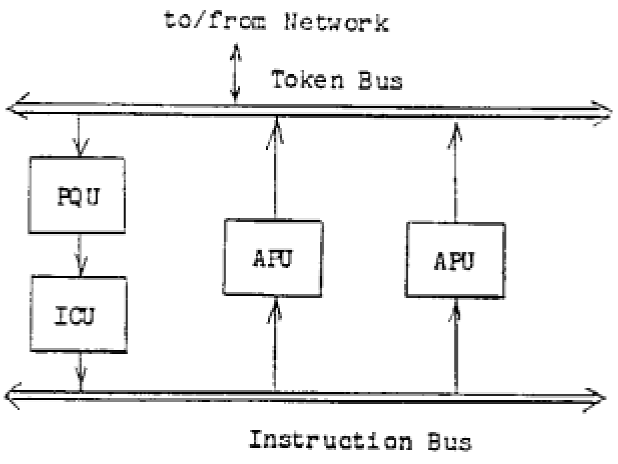


Figure 6 A Data Flow Graph for Head Unification



NN: Network Node
PE: Processing Element
SM: Structure Memory

Figure 7 A Configuration of the PIM-D Hardware Simulator



PQU: Packet Queue Unit
ICU: Instruction Control Unit
APU: Atomic Processing Unit

Figure 8 A Processing Element Module of the PIM-D

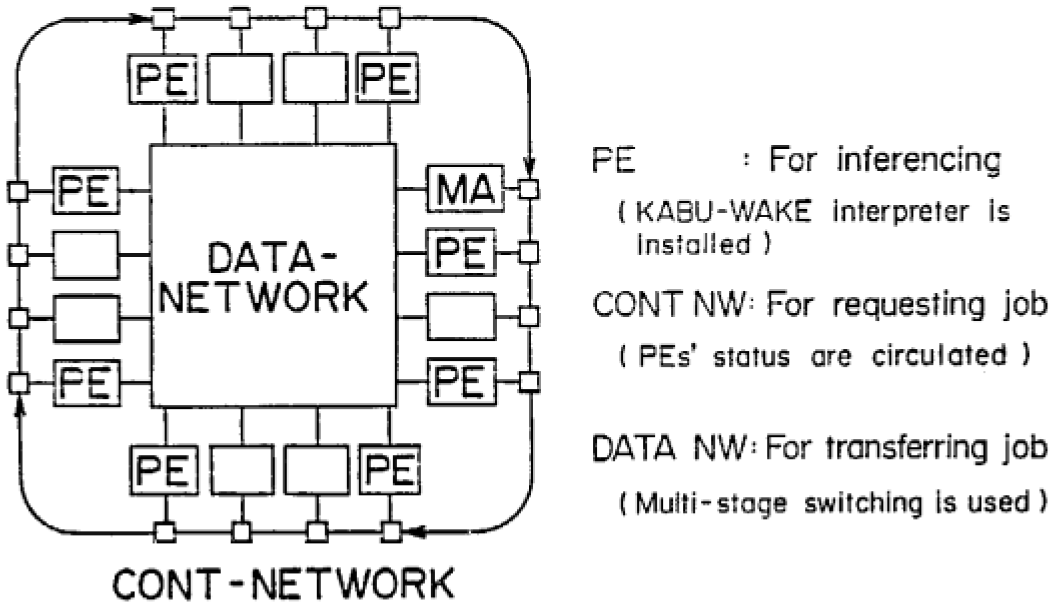

PE
DATA-NETWORK
MA
CONT-NETWORK
PE : For inferencing
(KABU-WAKE interpreter is installed)
CONT NW: For requesting job
(PEs' status are circulated)
DATA NW: For transferring job
(Multi-stage switching is used)

Figure 9 The Kabu-wake Method Experimental System

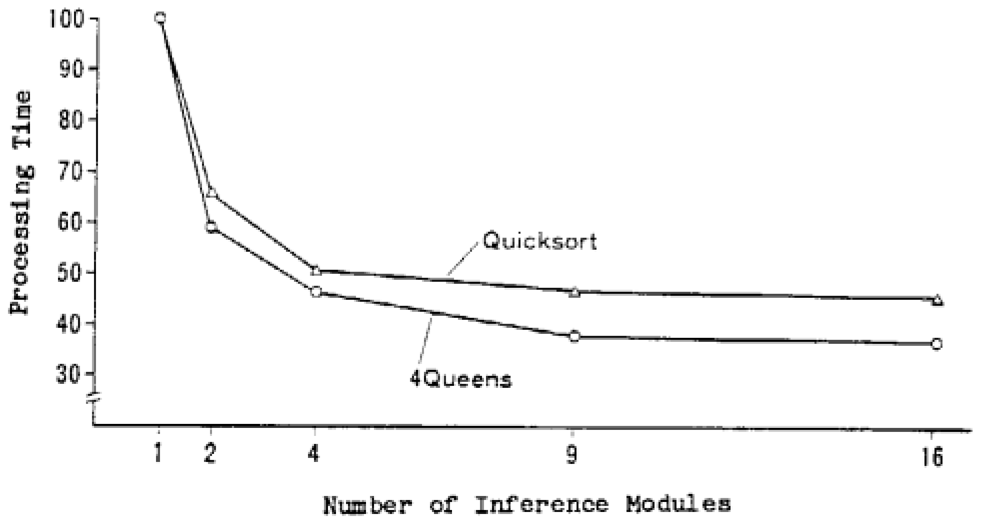


Figure 10 Performance Improvement Ratio in PIM-R

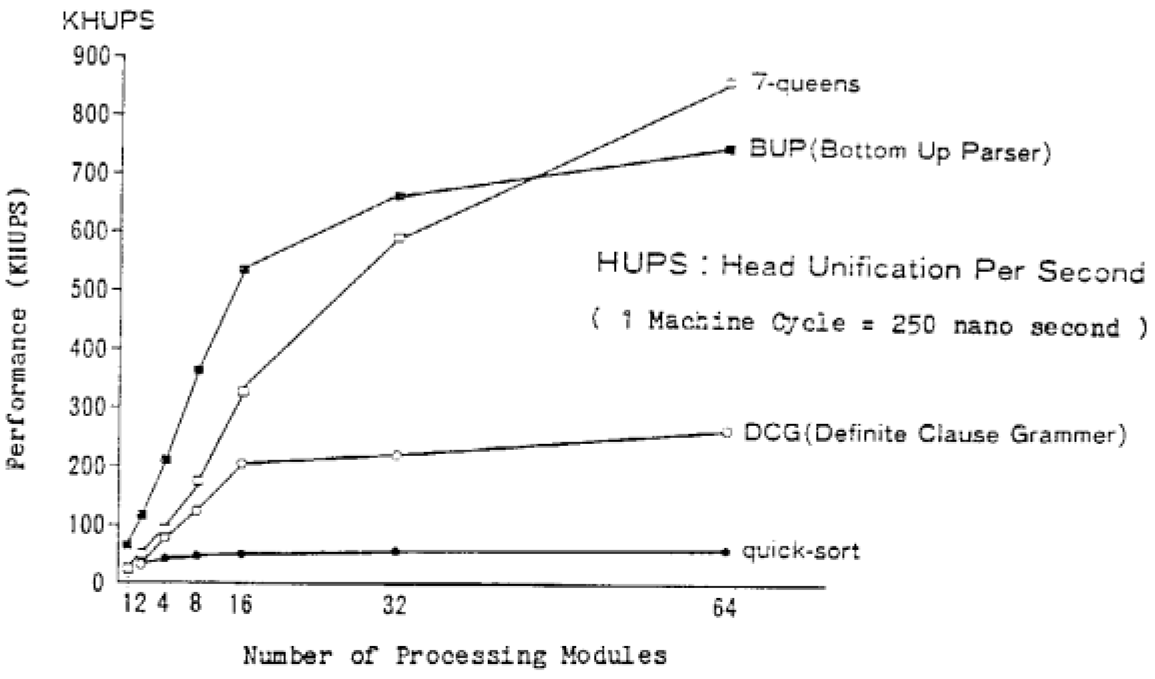


Figure 11 Performance Improvement Ratio in PIM-D

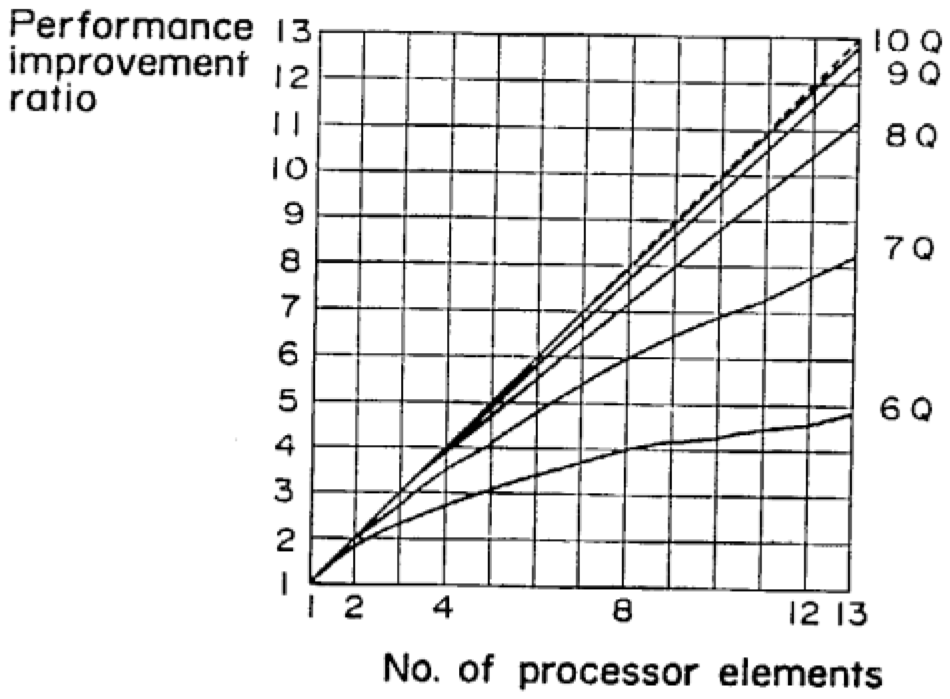


Figure 12 Performance Improvements Ratio in Kabu-wake Method

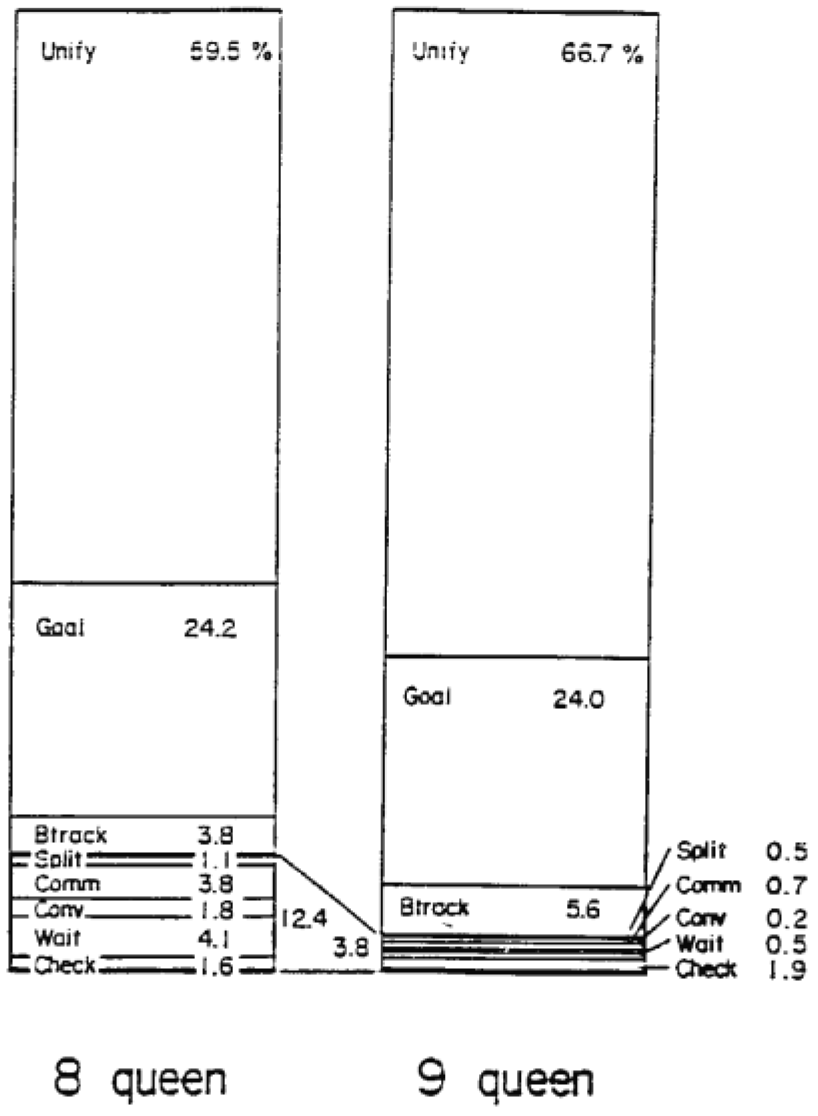


Figure 13 Details of Execution Time in an Inference Processor Element