TR-278

Inference Machines in FGCS Project

by

S. Uchida

June, 1987

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Inference Machines in FGCS Project

Shunichi UCHIDA

Institute for New Generation Computer Technology (ICOT)
Mita-Kokusai Building 21F., 4-28, Mita 1, Minato-ku, Tokyo

Abstract

In Japan's fifth generation computer systems (FGCS) project, logic programming is adopted for the base for software and hardware systems to be developed. As a primitive operation of logic programming is syllogistic inference, machines studied and built in the project are called **inference machines**.

One of the project's target machines is a parallel inference machine (PIM) having about 1000 processing elements. Smaller scale PIMs are also planned as intermediate targets. In addition to PIMs, sequential inference machines (SIMs) have been developed for a software development tool. A personal type SIM is called PSI which is a logic programming workstaion. For parallel software development, a multi-PSI system which consists of several CPUs of PSI connected with a high-speed network, is also under development.

In this paper, these sequential and parallel inference machines, some of which have already been developed, some are planned, shall be described with their languages and operating systems.

1 Introduction

Japan's fifth generation computer systems(FGCS) project aims at the research and development of new computer technology for knowledge information processing systems (KIPS) that will be required in 1990's. The project started from April 1982 as a ten year project. It is divided into three stages, namely, the initial stage (1982–1984), the intermediate stage (1985–1988) and the final stage (1989–1991).

The knowledge information processing systems are considered to have logical inference mechanisms using knowledge bases as their central functions. Logic programming is adopted as the base for software and hardware systems developed in this project. The final target computer system is generally considered to have two types of functions, namely, a high-speed inference function and a knowledge base management function. Two types of the machines have been studied, namely, parallel inference machines (PIM) and knowledge base machines (KBM). The target of PIM research and development is a highly parallel machine having about 1000 processing elements. In addition to these research and development items, software development support systems are also being developed. One of these systems is a personal sequential inference machine (PSI), a logic programming workstation.

The machine languages of all these inference machines are based on logic programming. These languages are called kernel languages (KLn, n=0,1 or 2). KL0 is a sequential language for PSI. KL1 is a parallel language for PIM. KL2 is for the final target machine.

In the initial stage, main research efforts of the inference machines were devoted more to the design of machine architectures than to the fast and compact implementation of machine hardware. From the intermediate stage, the implementation efforts using custom LSIs began with the development of the smaller version of PSI (PSI-II). Now, we plan to implement the intermediate stage PIM (PIM-I) containing about 100 processing elements. We have to use some custom VLSIs for this implementation.

In this paper, the inference machines, some of which have already been developed and others which are planned, shall be described with their languages and operating systems.

2 Sequential Inference Machine (SIM)

In the initial stage, we developed two types of sequential inference machine (SIM) as a software development tool. One was named PSI and the other was named the Cooperative High-speed Inference machine (CHI).

PSI was intended to be used as a common software development tool in the project. It was designed as a logic programming workstation having its programming and operating sytem on it. The development must be completed in a year and a half. Stable operation and smooth delivery were considered to be important. PSI adopted a rather conservative implementation technology using fast TTLs.

On the other hand, CHI could be designed more freely. We intended to design the fastest possible Prolog machine. It was designed as a backend high-speed processor and implemented using CML (Current Mode Logic). Its machine language was designed based on the abstract machine instruction set proposed by Tick and Warren [4] which is now widely known as WAM (Warren's Abstract Machine instruction set).

The SIM programming and operating system (SIMPOS) was developed on PSI. To describe SIMPOS, a new system description language, ESP (Extended Self-contained Prolog) was developed. SIMPOS is a personal OS having a multi-window based human interface and a programming environment for ESP like the LISP machine OS. SIMPOS has been continuously improved and extended. Its size is about 370K lines in ESP consisting of about 2100 class modules.

ESP is a new type of language combining object oriented language features and logic programming language features. It can be used not only for system description but also for knowledge representation. SIMPOS is fully written in ESP and its module structure is completely based on the object oriented concept. Thus, SIMPOS modules (class definitions) can be easily inherited and customarized by the user like the flavor system of the LISP machine OS. ESP is a higher level language than LISP, Prolog and Smalltalk. The productivity of software is very much improved, however, a variety of firmware and hardware support and compiling techniques are necessary to make execution speed satisfactory.

About 130 PSI machines have been distributed to ICOT, our cooperative companies, some universities and research institutes. All these PSI machines are connected by LAN and the public packet switching network.

From the beginning of the intermediate stage, we started the effort to develop a smaller version of PSI (PSI-II) and CHI (CHI-II) using custom LSIs. The CPU of the PSI-II is also intended to be used for the multi-PSI system which will be used for prallel software

Table 1: Main features of PSI-I and PSI-II

	PSI-I	PSI-II
Device	TTL (Fast)	CMOS-G.A., TTL
Cycle time	200 ns	200 ns
Word width	40 bits	40 bits
WCS	64b x 16KW	53b x 16KW
Cache memory	4KW x 2	4KW x 1
Main memory	16MW (Max)	64MW (MAX)
Memory chip	256 Kbit	1 Mbit
Max. No. of Process	64	S/W defined
Machine code	Table type	WAM type
Structure data	Sharing	Copying
Exe. speed(Average)	30 KLIPS	150 KLIPS
Exe. speed(Append)	35 KLIPS	333 KLIPS

Table 2: Performace of PSI-I and PSI-II

	PSI-I (KLIPS)	PSI-II (KLIPS)
Append	35	333
Naive Reverse	34	271
Quick Sort	40	132
Tree Traverse	41	100
8 Queens	60	162

development. We have almost completed the hardware development of PSI-II and CHI-II. They will be used from this fall in this project. Main features of these machines are shown in Tables 1, 2 and 3.

In the implementaion of these hardware and firmware systems, compilers and interpreters, multi-window systems and other parts of the operating systems, valuable experience and design criteria have been accumlated at ICOT. They are now being used for the development of PIM-I and its OS, PIMOS.

3 Parallel Inference Machine (PIM)

3.1 Outline of PIM research plan

The research target of PIM in the final stage was roughly described in the project plan that it would have about 1000 processing elements and attain 100 M to 1 G LIPS. We now feel that this target will be feasible in five years. As an intermediate stage target, we now plan to develop PIM-I which will have about 100 processing elements and attain about 10 to 20 MLIPS including some overhead caused by the operating system.

In the initial stage, we studied several PIM models such as reduction and dataflow for the parallel execution mechanisms of logic programming languages. In the beginning, we used pure Prolog and examind both OR-parallel and AND-parallel execution mechanisms. We built several software and hardware simulators. For dataflow, we designed a machine

Table 3: Main features of CHI-I and CHI-II

	CHI-I	CHI-II
Device	CML	CMOS-G.A., TTL
Cycle time	100 ns	170 ns
Word width	32 bits	40 bits
WCS	78b x 16KW	same
Cache memory	16KW x 2	same
Main memory	64MW (Max)	128MW (MAX)
Memory chip	256 Kbit	1 Mbit
Machine code	WA type	extended WAM
Structure data	Copying	same
Exe. speed(Append)	280 KLIPS	about 400 KLIPS

model and implemented a hardware simulator which was named PIM-D. It executed logic programs in a goal-diven manner. The hardware simulator consisted of 16 PEs and 15 structure memory modules. They were connected by a hierarchical bus network. Each PE was built using the AMD's 2900 series bit-sliced microprocesors and TTLs.

For the reduction, we designed and implemented a simulator, PIM-R. It consisted of 16 MC68K processors connected by a common bus and a shared memory. For the study of the hardware support of job division and allocation in a multiprocessor environment, we designed the Kabuwake-method and implemented it by software on a hardware simulator which consisted of 16 MC68K based computers connected by a switching network and a ring network.

Through this research, we learned many lessons on the parallel execution of logic programming:

1. We must have a simple and elegant language which enables us to introduce compiler optimization compiler effectively and also make the machine hardware simple and fast.

2. The machine language must have such features as to describe an operating system. Some features like process synchronization have to be included in the language model. Then, AND-parallel-type language features are necessary.

3. The amount of the hardware system can not be large for stable implementation of many PEs. For example, commercially available VLSI technology would not be sufficient to fully implement complex architectures such as a dataflow architecture.

In parallel with the PIM research, the design of the parallel logic programming language for PIM (KL1) was carried out. Considering the requirements imposed by the PIM design, a simple parallel logic programming language, GHC (Guarded Horn Clause) was designed. It is a committed-choice AND-parallel logic programming language. We decided to use it as the base for the PIM and PIMOS research in the intermediate stage.

In the beginning of the intermediate stage, we made the detailed plan for PIM research and development. We had realized the importance of the knowledge about parallel languages and parallel software systems, especially the parallel operating system, to design the efficient parallel hardware systems.

The functions of the software systems, for example, the optimization made by the compiler and the strategy of job division and allocation in the operating system greatly influence the optimal design of the machine language and architecture of PIM.

We decided to adopt a simple and fast architecture for the PE suited to the GHC execution mechanism and the cluster structure for the connection mechanism. In addition, we decided to make the hardware system fast and stable so that we could build larger scale parallel software systems like the experimental operating system.

To be able to design the practical hardware of the PIM, we decided to develop an experimental parallel operating system, PIMOS, in more systematic way. To provide the software researcher of PIMOS with a parallel hardware environment very quickly, we decided to develop the multi-PSI system. The second version of the multi-PSI system will be a multi-processor having up to 64 PEs being connected by two dimensional mesh network. The PE is the same as the CPU of PSI-II.

In the multi-PSI system (V.2), the KL1 interpreter will be implemented in the firmware of the PE. Its execution speed will be sufficient to actually support the large scale parallel software systems including PIMOS.

3.2 PIM-I, Intermediate stage target

As the intermediate stage targert of PIM, we now intend to build the experimental system as follows:

1. The number of PEs is About 100.
2. Target processing speed: For the total system, 10 to 20 MLIPS including some overhead caused by PIMOS. For each PE, 200 to 500 KLIPS for KL1.
3. Connection mechanism: In a cluster, a shared memory with parallel cache. Between clusters, a switching network.
4. Machine language: KL1-B based on GHC.

We decided to use the tag-architecture for the PE optimized to GHC (KL1-B). GHC can handle from very fine granularity to coarse granularity. The key problem of the PE architecture is how to make the process switching faster. The optimization by the compliler to reduce the frequency of process switching is also an important research item. Some support for garbage collection is necessary because GHC tends to consume much heap area.

For the connection mechanism, we introduce the cluster in which around 8 PEs are connected by a shared memory with a parallel cache system [12] [13] [14] to achieve fast communication among PEs. This mechanism is chosen because the language features of GHC require very quick responses for communications which are issued in unification operations. The granularity of GHC programs, namely the size of the processes, could be very small, and the synchronization of the parallel processes is made using shared variables. Then, the lock mechanism is also required. To deal with these difficult conditions, we decided to introduce the rather complex parallel cache system. For the connection among the clusters, we introduce a packet switching network. We have not yet decided its details, however, one example is the connection network of the multi-PSI system. The connection mechanism is very closely related to the strategy of job division and allocation among

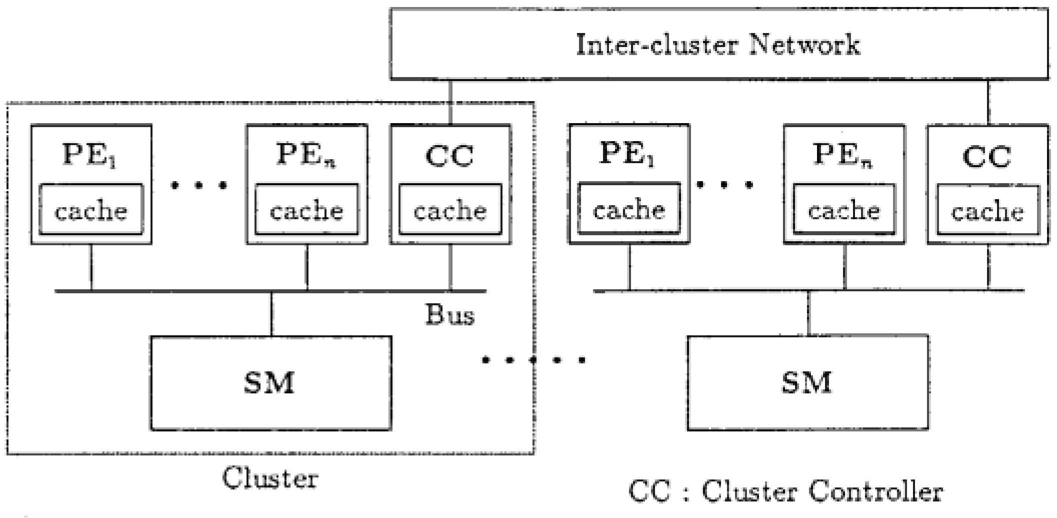

Figure 1: PIM Overview

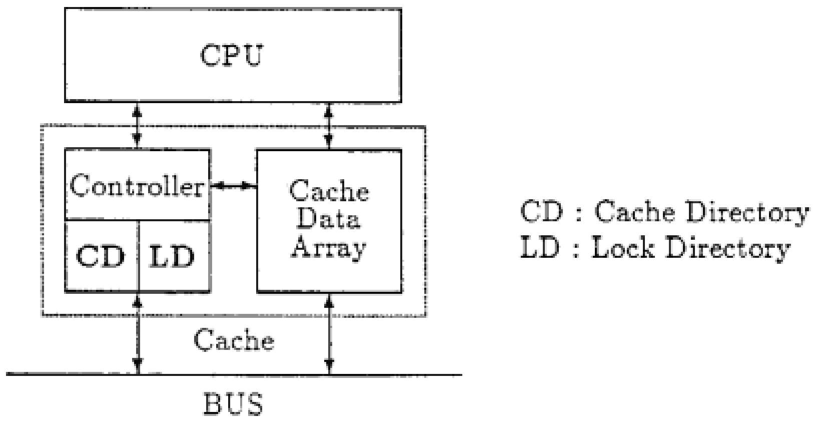

Figure 2: Cache and Lock Mechanism

clusters and PEs. Then we will design its detailed mechanisms considering the requirements imposed by the PIMOS design.

We have not yet completed the implementation details of PIM-I. However, we roughly plan to implement a PE on a printed circuit board the size of A4 paper using standard-cell VLSIs and 2 to 4 clusters in a standard cabinet.

3.3 PIMOS and Multi-PSI system

The research and development of PIMOS started with the design of the KL1 language system. The basic formal model of KL1 is given by GHC just as pure Prolog is a model for practical logic programming languages such as DEC10 Prolog and Prolog-II.

We defined three language layers for a KL1 language system as shown in Fig. 3. The core part of this system is KL1-C, namely GHC. KL1-U is the user language or system description language which is now under design. KL1-P is a notation to permit a programmer to explicitly specify the way of dividing a job into parallel processable subjobs and also assign the amount of computational resources to each subjob. KL1-B is

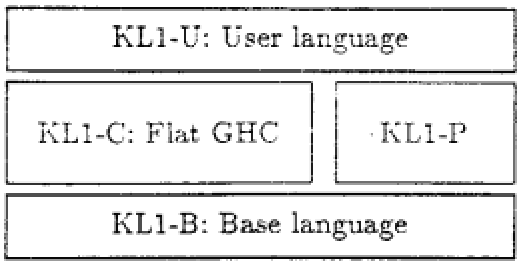

Figure 3: The KL1 Language Systems

the machine language which is directly executed by the PEs.

For KL1-U, we intend to introduce the object oriented concept for its modularization just as we sucessfully introduced it for ESP. KL1-B is now under design in parallel with the design of the architecture of the PE. We are designing KL1-B so that we can make full use of optimization by the compiler.

The experimental versions of the KL1 language sytem include a GHC compiler, and a KL1-B interpreter on such machines as PSI, multi-PSI(V1), VAX, DEC20 and Balance 21000. Several small parallel programs were written in KL1 and run on these machines to evaluate the KL1 execution method and also observe the effect of the job division and allocation strategy.

We have started to design PIMOS. Main design features of PIMOS are summarized as follows:

1. It is designed as a single unified operating system which controls the parallel hardware system just as one system. Main functions are the hardware resource management and the control of load distribution and balancing.

2. Basic functions for the programming environment are implemented on it. Debugging, man-machine interface, measurement and evaluation and exception handling are being considered.

3. It is fully written in KL1 and implemented on the Multi-PSI system. The interpreter for KL1-B is implemented in firmware, and will be moved to PIM-I after the PIM-I hardware system is completed.

PIMOS will be actually built on the multi-PSI system (V2) which will be developed in the end of 1987.

The multi-PSI system was planned to encourage the software researcher to actually make larger scale parallel programs in KL1. Especially, the development of PIMOS needs fast and stable parallel hardware. The multi-PSI is best suited to this purpose at ICOT.

As the key item of parallel software research is the job division and allocation problem and the mapping between software models and machine architectures, we adopted a simple topology for the inter-PE network for the multi-PSI, namely, two dimensional mesh structure.

We built the first version of the multi-PSI using 6 PSI-I systems. We made a specialized network hardware. Each node of the network has a simple routing control mechanism and 5 channels. One channel is connected to the PSI and others are connected to 4 neighbors. Each channel has two 8 bit-buses and two independent FIFO buffers which are used for read and write data transfers. The data transfer rate is 500 Kbytes per second.

The multi-PSI V2 is now under development. One PE consists of a CPU, 16 MW main memory and connection hardware for one node of the network. The connection hardware is improved to achieve faster data transfer and augment the routing function. Some hardware support for load balancing is added to the connection hardware. Two custom LSIs are included in the hardware. As one PE is not so small, 8 PEs will be implemented in one cabinet. Up to 8 cabinets, namely 64 PEs, can be connected.

The KL1-B interpreter will be implemented in the firmware of the PEs. The firmware interpreter is expected to attain 100 to 150 KLIPS for KL1. The communication delay between the two PEs of the multi-PSI is much longer than that of PIM-I which uses the shared memory with the cache system. The job division and allocation strategy must be more optimized on the multi-PSI than PIM. This means that we have to find better method for job division and allocation which makes full use the locality of communication in the given progrms. If this is successful, we can attain a few MLIPS for well organized parallel programs on the multi-PSI. Research results of this kind will be reflected in the network design of PIM.

4 Concluding Remarks

The research and development of inference machines is closely related to many other research fields. Inference machines are considered to be general purpose machines for AI applications. They need more general and powerful functions than conventional machines. Limitless symbol crunching jobs are apparently existing although many of them are not well formalized to be suited to existing parallel architectures. The implementation of sophisticated parallel architectures will require more advanced VLSI technology.

Considering this background, we have been making the best effort to combine current advanced knowledge in software, language, architecture, hardware, and VLSI. Our first effort was the development of PSI and SIMPOS where we combined a multi-window based personal operating system, a logic programming language and a tag architecture. We did not combine the most advanced VLSI technology because our project was planned to use commercially available LSI technology.

Our next effort is much more difficult than the first one. We are trying to combine a parallel software system, a parallel logic programming language and a parallel architecture. Naturally, we have to use the most advanced VLSI technology availabe to us.

We decided to combine the above elements in a step by step manner. The first step is to combine KL1 and PSI as the multi-PSI system, the PE of PIM-I and VLSI switch in the cluster, and, KL1 and a parallel OS as PIMOS. PIMOS is probably the most difficult job among the three because we have very little knowledge about parallelism in programming languages, operating systems, paradigms, algorithms and applications.

Finally, we intend to build fast, simple and stable hardware systems to allow software systems to be as large as possible. Advances in VLSI technology will enable us to make more sophisticated PEs and network systems in the near future. This will again enable us to build larger scale software systems. This bootstrapping development will be the most natural strategy for inference machines research.

Acknowledgment

The author would like to express his gratitude to the researchers of his laboratory, especially, Dr. Atsuhiro Goto, Dr. Kazuo Taki and Dr. Takashi Chikayama. He would like to thank Dr. Kazuhiro Fuchi, the director of the ICOT research center, for his encouragement.

References

[1] A. Goto and S. Uchida. *Toward a High Performance Parallel Inference Machine -The Intermediate Stage Plan of PIM-*. TR 201, ICOT, 1986.

[2] S. Uchida. *Toward a Parallel Inference Machine* TR 196, ICOT, 1986. Proc. of COMPAR 86, Sept. 1986.

[3] H. Nakashima, K. Taki and K. Nakajima *Performance and Architecture Evaluation of the PSI Machine* To appear in Proc. of ASPLOS-II, Oct. 1987.

[4] David H.D. Warren. *An Abstract Prolog Instruction Set.* Technical Note 309, Artificial Intelligence Center, SRI, 1983.

[5] K. Ueda. *Guarded Horn Clauses.* TR 103, ICOT, 1985.

[6] K. Taki. *The parallel software research and development tool : Multi-PSI system.* Technical Report No. 237, ICOT, 1986. Proc. of France-Japan Artificial Intelligence and Computer Science Symposium 86, October 1986.

[7] H. Nakashima, K. Nakajima *Hardware Architecture of the Sequential Inference Machine PSI-II* , To appear in Proc. of 4th Sympo. on Logic Programming, Aug. 1987.

[8] N. Ichiyoshi, T. Miyazaki and K. Taki. *A Distributed Implementation of Flat GHC on the Multi-PSI.* Technical Report No. 230, ICOT, 1986. To appear in Proc. of 4th International Conference on Logic Programming.

[9] S. Habata, et al *Co-operative High Performance Sequential Inference Machine: CHI* Proc. of IEEE International Conference on Computer Design: VLSI in Computer and Processors, Oct. 1987.

[10] Y. Kimura and T. Chikayama *An Abstract KL1 Machine and Its Instruction Set* Technical Report No. 246, ICOT, 1987. To appear in Proc. of 4th Sympo. on Logic Programming, Aug. 1987.

[11] T. Chikayama and Y. Kimura *Multiple Reference Management in Flat GHC.* Technical Report No. 248, ICOT, 1986. To appear in Proc. of 4th Internaltional Conference on Logic Programming, May, 1987.

[12] M. Sato et al. *KL1 Execution Model for PIM Cluster with Shared Memory.* Technical Report No. 250, ICOT, 1986. To appear in Proc. of 4th Internaltional Conference on Logic Programming, May, 1987.

[13] R. H. Katz et al. Implementing a cache consistency protocol. In *Proc. of the 12th Annual International Symposium on Computer Architecture*, June 1985.

[14] J. R. Goodman. Using cache memory to reduce processor-memory traffic. In *Proc. of the 10th Annual International Symposium on Computer Architecture*, 1983.

[15] P. Bitar and A. M. Despain. Multiprocessor cache sychronization. In *Proc. of the 13th Annual International Symposium on Computer Architecture*, June 1986.