ICOT Technical Report: TR-626

TR-626

Highly Parallel Knowledge Processing and Requirements for Future Hardware Technology

by
S. Uchida

February, 1991

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

(03)3456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Highly Parallel Knowledge Processing and Requirements for Future Hardware Technology

Shunichi UCHIDA
Institute for New Generation Computer Technology (ICOT)
21F Mita-kokusai Bldg., 1-4-28 Mita, Minato-ku, Tokyo 108

1 FGCS Project

Ever since computers became available as tools for symbolic processing, many research projects on computer science have been conducted to develop a variety of technologies for intelligent computers. Technologies that help people do intellectual work.

The fifth generation computer systems (FGCS) project is one such research project. It was planned as a 10-year project and started in June 1982 as a Japanese national project. It is unique in that it intends to use **logic programming** to bridge **knowledge processing** technology and **parallel processing** technology as shown in Fig. 1. It aims to develop a new computer technology that is best suited to knowledge information processing (KIPS), expecting that a KIPS computer will be a new general purpose computer in the next generation.

This paper describes, new computers for highly parallel knowledge processing. It is based on the research results of the FGCS project in terms of software and hardware technologies and on the requirements for future hardware technology.

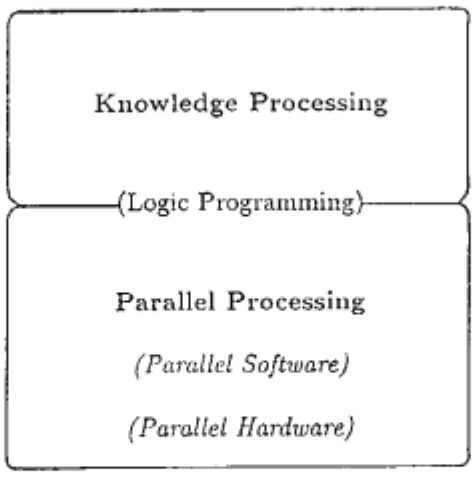


Fig.1 Framework of FGCS

2 Knowledge Processing

In knowledge processing systems, how to describe knowledge is an important problem. Logic-based schemes are often used to describe various fragments of knowledge. To store and retrieve these knowledge fragments, logical inference mechanisms are generally used. Logic programming languages are very suitable for a base for knowledge description and processing.

In the FGCS project, new sequential and parallel logic programming languages, namely ESP and KL1, have been developed and used as bases for processing knowledge such as natural language grammars and dictionaries, expert systems' rules, mathematical theorems, and genome analysis rules. These logic programming languages, ESP and KL1, are higher-level languages than conventional languages such as C and Fortran. However, much-higher-level knowledge programming languages dedicated to specific application fields are developed and used for ease of description and efficient production of knowledge bases or rule bases. So we can say that ESP and KL1 are used as the "assembler" for knowledge processing.

These knowledge programming languages and their sophisticated inference mechanisms are usually provided as knowledge programming environments and enable us to produce complex intelligent systems.

However, these languages require very powerful support of language processors and machine hardware to provide their users with practical response time and performance. Ease of program production burdens the language processors and machine hardware with heavy computational load for their support.

3 Parallel Processing

Parallel processing is a key technology to fulfill the above requirement. This technology has great potential to provide powerful support for knowledge processing. However, parallel processing for logic programming is not well developed yet. Generally, parallel processing technology for symbolic processing is not as mature as the technologies for numerical processing and some specialized processing like image processing.

Parallel processing for symbolic processing, which is the base of logic programming, will be much more general than parallel processing for conventional numerical processing. This is because it should provide users with a general parallel programming environment where they can write large scale parallel application programs, making full use of large-scale parallel hardware systems.

Usually, design of parallel hardware requires technical knowledge on the behavior of parallel software systems run on the hardware, such as how a parallel operating system manages various resources, how parallel application systems behave, and what kind of hardware support is especially necessary.

A new sequential logic programming language, ESP, was designed at the beginning of the FGCS project. After that, a logic programming workstation, PSI was developed to provide a programming environment for ESP. In parallel with this development, a new parallel logic programming language, KL1, was designed.

To provide development teams of the KL1 parallel language processor and the parallel OS with a parallel hardware environment, a Multi-PSI system was developed using PSI CPUs as its processing elements. A Multi-PSI has 64 PEs connected by a 2-dimensional mesh network. On Multi-PSI, a parallel operating system, PIMOS, has been implemented and used to develop various parallel benchmarking programs.

Using the technical knowledge gained by the measurement and evaluation of the behavior of PIMOS and benchmarking programs, a larger-scale parallel inference machine (PIM) was designed. This final version of PIM is also called a prototype of FGCS. It is expected to have 1000 PEs and a processing speed of more than 100M LIPS (Logical Inferences Per Second). Its processing power is expected to support practical knowledge-processing application systems efficiently.

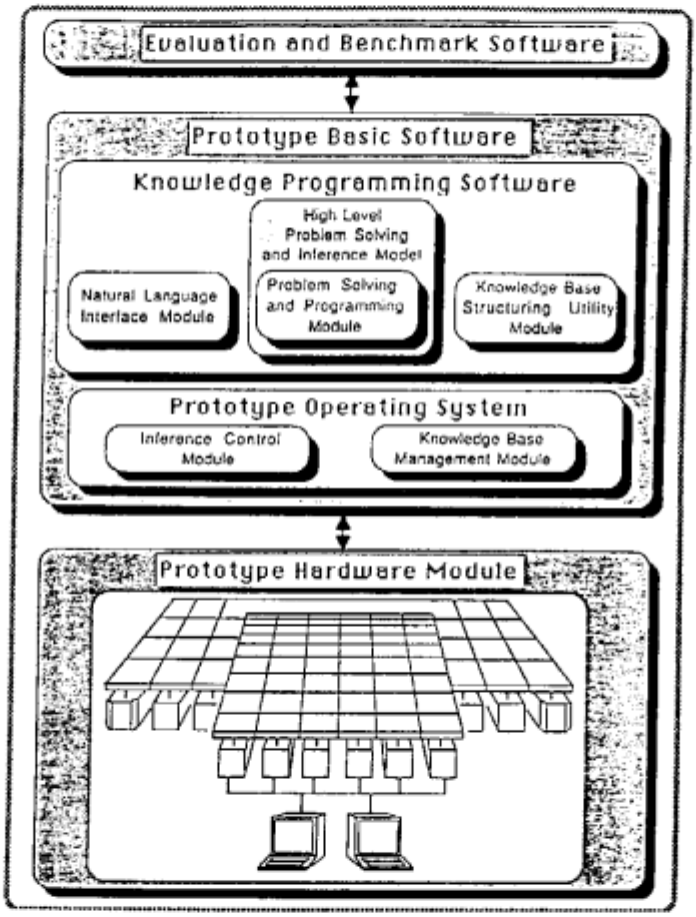


Fig.2 Organization of Prototype FGCS System

4 Organization of FGCS Prototype System

As a main research product, several important components are intended to be integrated into an FGCS prototype system in 1992 as shown in Fig. 2. This system will

have the prototype hardware system at the bottom layer. This first layer contains parallel hardware and the KL1 language processor and is usually called PIM. The second layer is the operating system, PIMOS and knowledge base management system, KBMS. The third layer is knowledge programming software. Most of this layer is still in research phase and includes many experimental program modules such as natural language interface modules and high-level inference modules. The fourth layer consists of various evaluation and benchmarking programs. The first and second layers are integrated well enough to be provided as a practical tool for the development of large-scale knowledge application systems using KL1. This system is shown simply in Fig. 3.

Currently, several application systems are under development such as parallel theorem provers, natural language processing programs, VLSI CAD systems, legal reasoning programs, and genetic information analysis systems. Through the development of these application systems, the KL1 programming environment including PIMOS is proving to be practical and efficient. It enables us to make full use of large-scale parallel hardware systems having hundreds, or thousands, of processing elements.

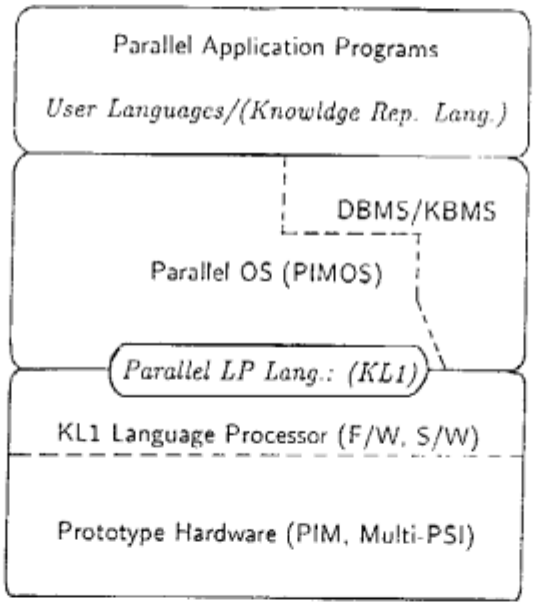


Fig. 3 Structure of Parallel Inference System

5 Research on Parallel Inference Machines

5.1 From Sequential to Parallel

In the FGCS project, research and development of the parallel inference machine was started from the design of a sequential language, ESP, and development of a sequential

logic programming workstation, PSI-I, and its operating system, SIMPOS. About 100 PSI-I workstations were produced and used for research of logic programming and knowledge processing.

History of PSI development is summarized as follows:

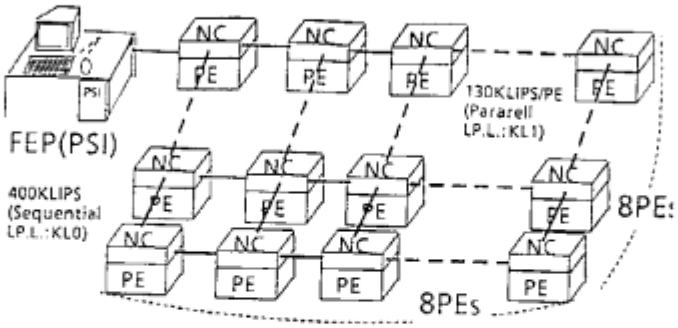
1984 **PSI-I** 37 KLIPS (KL0/V1)
-ESP language and SIMPOS/V1
1986 **PSI-II** 330 KLIPS (KL0/V2)
-SIMPOS/V2 and Kappa-I
1988 -SIMPOS/V5 and **Kappa-II**
-Pseudo Multi-PSI and **PIMOS-S**
**About 300 PSI-II Machines and Network
1991 **PSI-III** ⇒ 1.2 MLIPS (KL0/V2)
-SIMPOS/V7 + UNIX
-Domestic and international network link
to access PIM systems at ICOT

After the PSI-I, an improved version called PSI-II was developed using nine CMOS gate array chips (8K gates/chip). About 300 PSI-IIs were produced and used for research for knowledge processing. Furthermore, PSI-IIs have two independent firmware interpreters for ESP and KL1. Using the KL1 interpreter, a PSI-II can work as a pseudo-parallel machine and can thus be used for tools for parallel software development.

A CPU of a PSI-II was also intended to be used as a processing element (PE) of an experimental parallel inference machine, Multi-PSI, having 64 PEs as shown in Fig.4. Six Multi-PSIs were produced and used to develop a parallel language processor for KL1 and the parallel operating system, PIMOS. Many experimental parallel benchmark programs and application programs have also been developed on this system. Thus, the Multi-PSI is a bridge machine from sequential machines to parallel machines.

While PIMOS and parallel programs were being developed, the design of the final PIM hardware was started. In this design, functional requirements imposed from the characteristics of KL1 and behavior of PIMOS were analyzed to get better hardware design. As a result, several alternative designs came out and a few of them were chosen to be implemented and evaluated. Five modules of PIM are now under development.

All of them are designed to support KL1 and PIMOS efficiently.



- Machine language: KL1-b
- Max. 64PEs and two FEPs (PSI-II) connected to LAN
- Architecture of PE:
  - Microprogram control (64 bits/word)
  - Machine cycle: 200ns, Reg.file: 64W
  - Cache: 4 KW, set associative/write-back
  - Data width: 40 bits/word
  - Memory capacity: 16MW (80MB)
- Network:
  - 2-dimensional mesh
  - 5MB/s x 2 directions/ch with 2 FIFO buffers/ch
  - Packet routing control function

Fig. 4 Structure and specification of Multi-PSI

As KL1 frees its users from explicit management of memory cells and explicit description of process synchronization, the language processor and machine hardware have to dealt with these by complex mechanisms such as garbage collection and dataflow synchronization.

A communication scheme and network structure between PEs is influenced by the structure of application programs. Thus, several different network structures should be evaluated with realistic benchmarking programs of appropriate scales. Then, each of five modules has a different architecture from others in terms of chip design, CPU instruction design, structure of inter-PE network and implementation of parallel language processor as shown in Table 1.

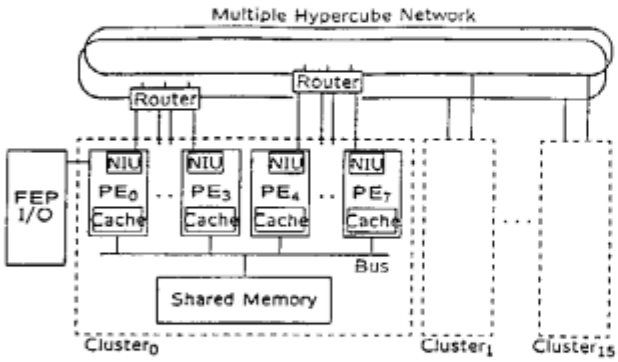
Table 1: Specifications of PIM modules

Item	PIM/p	PIM/c	PIM/m	PIM/i	PIM/K
Machine instructions	RISC-type + macro instructions	Horizontal microinstructions	Horizontal microinstructions	RISC-type	RISC-type
Target cycle time	50 nsec	50 nsec	50-60 nsec	100 nsec	100 nsec
LSI devices	Standard cell	Gate array	Cell base	Standard cell	Custom
Process Technology (line width)	Approx. 1 μm	0.8 μm	0.8 μm	1.2 μm	1.2 μm
Machine configuration	Multicluster connections (8 PEs linked to a shared memory) in a hypercube network	Multicluster connections (8 PEs + CC linked to a shared memory) in a crossbar network	Two-dimensional mesh network connections	Shared memory connections through a parallel cache	Two-level parallel cache connections
Number of PEs connected	512 PEs	256 PEs	256 PEs	8 PEs x 2	16 PEs x 2

History of PIM development is summarized as follows:

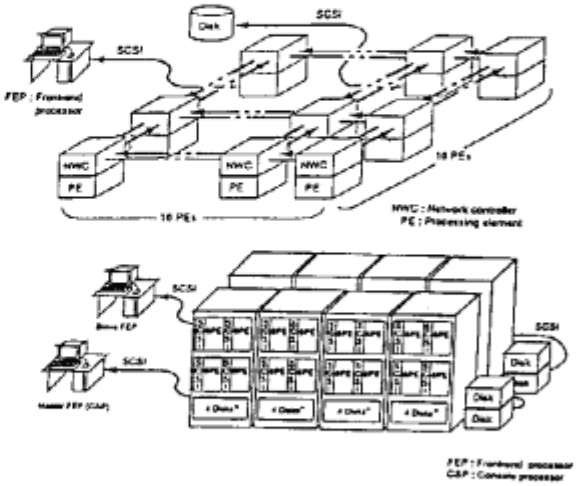
1985 **GHC** Model
1986 **Multi-PSI/V1** 1 KLIPS x 6PE (FGHC)
Parallel interpreter of FGHC
1988 **Multi-PSI/V2** 150 KLIPS x 64PE (KL1)
(2 - 5 MLIPS)
PIMOS/V1 and small benchmark programs
1990 **Final PIM** Chips and CPU Boards
PIMOS/V2 and many application programs
1992 **Final PIM System** 300-600 KLIPS/PE (KL1)
(PIM model/p ⇒ 200 MLIPS/512PE)
5 modules: model-p, m, c, k, i
Total 1072PEs = 512 + 256 + 256 + 32 + 16
PIMOS/V3 + KBMS(Kappa-P)
Parallel Application Systems

PIM model-P Architecture



- Machine language: KL1-b
- Architecture of PE and cluster
  - RISC + HLIC(Microprogrammed)
  - Machine cycle: 60ns, Reg.file: 40bits x 32W
  - 4 stage pipeline for RISC inst.
  - Internal Inst. Mem: 50 bits x 8 KW
  - Cache: 64 KB, 256 column, 4 sets, 32B/block
  - Protocol: Write-back, Invalidation
  - Data width: 40 bits/word
  - Shared Memory capacity: 256 MB
- Max. 512 PEs, 8 PE/cluster and 4 clusters/cabinet
- Network:
  - Double hyper-cube (Max 6 dimensions)
  - Max. 20MB/sec in each link

PIM model-M Architecture



- Machine language: KL1-b
- Architecture of PE:
  - Microprogram control (64 bits/word x 32 KW)
  - Data width: 40 bits/word
  - Machine cycle: 60ns, Reg.file: 40 bits x 64W
  - 5 stage pipeline
  - Cache: 1 KW for Inst., 4 KW for Data
  - Memory capacity: 16MW x 40 bits (80 MB)
- Max. 256 PEs, 32 PE/cabinet
- Network:
  - 2-dimensional mesh
  - 4.2MB/s x 2 directions/ch

Fig.5 Structures and specifications of Pim/p and Pim/m

PIM model P (Pim/p) and PIM model M (Pim/m) are representative of the five modules. Pim/p has a hierarchical network. The first-level network is a common bus with a coherent cache system in a cluster. The second level is a hypercube network. In a cluster, eight PEs communicate with each other using the common bus and coherent cache. Communication delay between PEs is very small and a few machine cycles. This is a great merit of this cluster structure. However, one address space is shared by eight PEs, all of them have to stop while garbage collection is being done. As the network has two levels, the programming for this machine needs a little deeper consideration on static and dynamic structures of programs to get better performance than the one for other machines like Pim/m.

Pim/m has a network structure similar to Multi-PSI, namely, 2-dimensional mesh. Each PE has its own separate memory and makes garbage collection independently. The programming for this machine is simpler than the one for Pim/p and less dependent on the size of the hardware. For instance, the same program structure will

be commonly applicable if the size of the hardware is changed from 64 PEs to 256 PEs. This kind of characteristic is called the "scalability". Pim/m structure has good scalability. However, any communication between PEs needs the usual message handling operations such as address conversion between local and global addresses and composition and decomposition of message contents.

Various comparisons will be done on the design parameters of the five modules with practical large-scale programs. The results of the comparisons will provide us with better criteria for the software and hardware design, including chip design.

6 Requirements for Future Device Technology

In the hardware design of PIM modules, there were too many functional requirements for a machine instruction set to actually implement them on a single board. We were evaluated and compared these requirements. We decided to realize some of them, firmware or software of language processors, including compilers. To enhance the hardware support as much as possible, we used sub-micron VLSI chips. Finally, each PE of the PIM modules could be contained on a single board.

Basically, architectures of PIM PEs are based on a mixture of tag architecture and the architecture of general purpose microprocessors. Tag architecture is simple but effective to support operations for symbolic processing. However, tag architecture combined with a multi-stage pipeline increased the complexity of the logic of PEs. Submicron VLSI technology was necessary to contain the logic for instruction execution and data processing in one chip. This is indispensable to make machine cycle time 50 to 60 ns. Cache control logic is another complex part of PEs. Submicron VLSI technology was again used to contain this part in a chip. In these two chips, about one million transisters are contained.

The design of the instruction set also influenced on the amount of the hardware, especially, the memory size of each PE. Pim/p employed a RISC type instruction set combined with a macro instruction set which is implemented by microprogramming. Each PE of Pim/p has a 50 KB microprogram memory. It also has a cache buffer of 64 KB and a local memory of 8 MB. Pim/m employed a fully microprogrammed control. Each PE of Pim/m has a 256 KB microprogram memory. It also has a 25 KB cache buffer and a local memory which is extendable up to 80 MB. In addition to these chips, one or two network chips and a floating point arithmetic chip are put on each PE board.

Currently, the sizes of PE boards of main PIM modules are between an A4 paper and a B4 paper. Thirty two PEs of Pim/p are now contained in a cabinet of which width, depth and height are 140 x 80 x 160 cm, respectively. This size is adequate as far as parallel programming technology is remain in an inmatured status.

Recently ICOT software researchers have become confident that they can make proper structures of many knowledge processing problems suitable to parallel processing. For example, the resource management scheme of PIMOS has now a scalable structure suitable for parallel processing.

This confidence will create new styles of PIM usage which demand more PEs, thus

require higher-density hardware implementation. One style of the usage is to use PIMs as a super computer of knowledge processing which is kept in a central machine room and shared by many users. Current implementation of PIM modules enables us to contain less than one thousand PEs. This style of the usage will demand more PEs, thousands of PEs.

Another style of the usage is to use smaller scale PIMs as knowledge processing workstations. The size of this workstation has to be small enough to be installed on the desk or under the desk. The maximum number of PEs for this style is limited mainly by the density of the hardware implementation. Thus, compact implementation of the PIM hardware is the key for this style.

Then, requirements for future hardware technology come from these new styles of the PIM usage. As shown in Fig. 6, the size of a CPU or PE of the machines developed in the FGCS project has changed from 11 boards of the PSI-I to a single board of the PIM PE. A next target is to make a PIM PE be a single chip. This will also enable us to make a cluster having eight PEs and a shared memory be a single board. This implementation could increase the maximum number of PEs for one machine to be thousands to ten thousands.

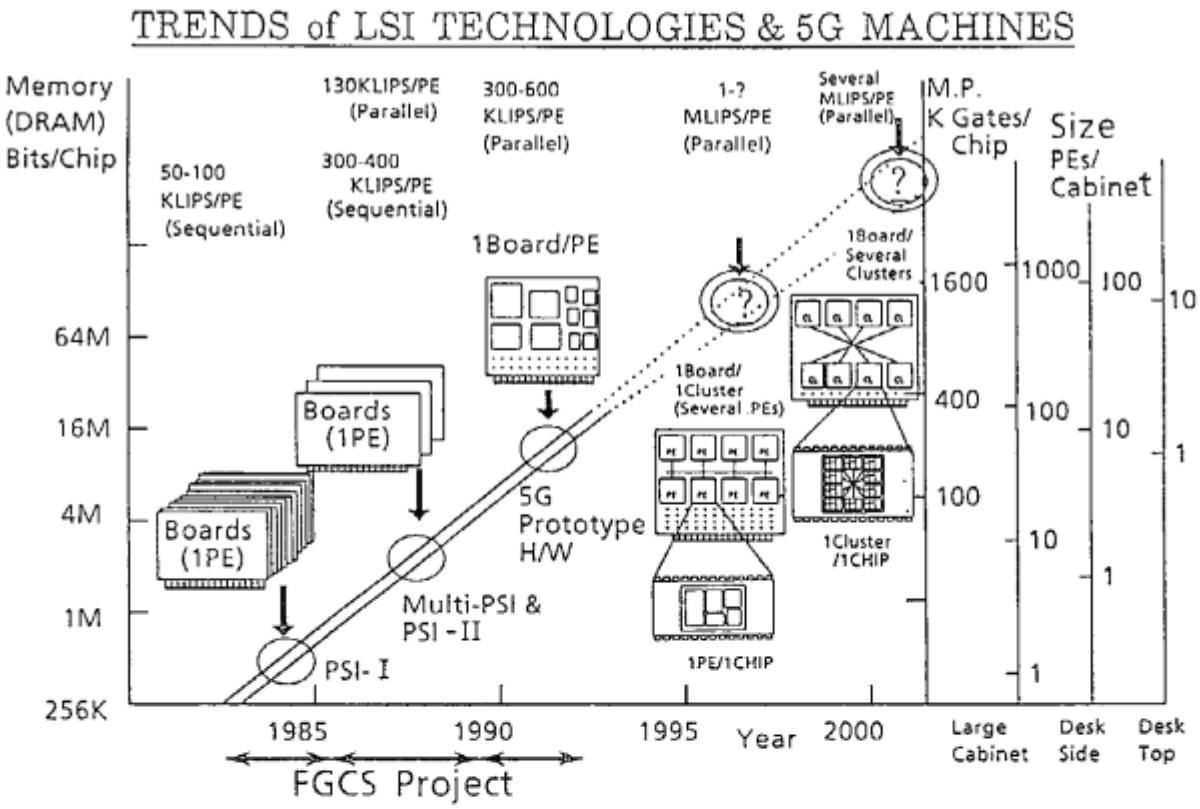


Fig.6 Changes of CPU implementation and VLSI technology on ICOT machines

After this target, a wafer scale integration will be a next target. If one cluster or

eight PEs are on a wafer, its performance will also be improved greatly because machine cycle time is expected to be short. The multi-processor configuration is suitable for the wafer scale integration because its repetition of the same processor cells has tolerance in chip production. For the implementation of the inter PE network, optical and electrical device technology may be used.

This direction will be common to that of general purpose machines, namely, main frames. Now, some main frames has employed a multi-processor configuration using general purpose microprocessor chips. This will become more popular in the future, although this is beyond the traditional von-Neumann machine model. A coherent cache system is now common not only to PIM like machines but also to these general purpose machines.

As the most part of the logic contained in a PIM PE is almost common to that of general purpose machines, it may be possible that a future general purpose microprocessor chip will include additional logic circuits dedicated to symbolic processing if a single chip can contain more logic circuit. This means that FGCS will merge with general purpose machines in the hardware level.

As an achievement of the FGCS project, the parallel programming technology based on parallel logic programming has been established firmly enough to guarantee the applicability of large scale parallel processing technology for knowledge processing applications. This will also encourage the research and development of more advanced semiconductor technology to realize highly parallel hardware having more sophisticated functions.

7 Acknowledgment

The author would like to thank many people who helped him in the preparation of this paper, especially, Mr. Takashi Kurozumi, vice director of ICOT research center, who gave him several comprehensive pictures.

The details of the systems mentioned in this paper are described in ICOT's Technical Reports and Memos. A list of these are available and free. Please contact to ICOT's international relations department by Fax (+81-3-3456-1618) or to the author by email (uchida@icot.or.jp).