ISSN 0910-6707

1993

Japan Computer Quarterly

見本

Japan Information Processing

Development Center

Fifth Generation Computer Systems (FGCS)

Project in Japan

No. 93

Japan Computer Quarterly

1993

Japan Computer Quarterly (JCQ) is published quarterly by the Japan Information Processing Development Center (JIPDEC), Kikai Shinko Kaikan Bldg., 3-5-8 Shibakoen, Minato-ku, Tokyo 105 Japan.

Publisher: Eiji Kageyama, President

Editor: Yuji Yamadori, Director Research & International Affairs

JIPDEC is a non-profit organization founded in 1967 with the support of the Ministry of International Trade and Industry, the Ministry of Posts and Telecommunications and related industry circles for the purpose of promoting information processing and the information processing industry in Japan.

JCQ, formerly called the *JIPDEC* Report, was first published in September, 1970 and is prepared with the assistance of the Japan Keirin Association through its Machine Industry Promotion Funds.

Translated and Printed by The Translation Institute of Technology, Science & Culture

Printed in Japan, March, 1993.

CONTENTS

No. 93

From the Editor

The word "generation" is frequently used to express progress made in the computer field, especially in relation to hardware. In other words, in accordance with the elements used to build computers, computers using vacuum tubes were called the first generation, those based on transistors and diodes the second generation, those adopting ICs the third generation, and those built using LSIs the third and a half generation. Since today's computers use VLSIs as elements on a large scale, it is said that computers have now entered a fourth generation. Computers have developed rapidly ever since the announcement of the first general-purpose computer using a built-in program system, and processing speed has also increased considerably along with innovations in element technology. However, conventional computers have all been based on the von Neumann theory, and theory has essentially followed in its footsteps, even up to the present.

The Ministry of International Trade and Industry has implemented various measures to improve computer technology in Japan. In the mid-1970's, MITI requested JIPDEC to carry out a basic investigation into the research and development of an epoch-making computer that would mark a significant departure from past theory and technology. To do this, JIPDEC decided to carry out investigations and research for a three year period starting in 1979 with cooperation from industry, academia, and government. For this investigation, JIPDEC established a research committee composed of researchers and specialists from universities, governmental and industrial laboratories, and users, and held active discussions in a wide range of fields, such as basic technical theory, the conditions of the social environment, and computer architecture. With expectations for the advent of a new generation computer, this computer was called the next generation computer, that is, the "Fifth Generation Computer". The results of this three-year investigation were reported to the government and were made public at an international symposium to an audience of Japanese and overseas specialists. The results were evaluated highly by the government, and in 1982, the government decided to establish an "Institute for New Generation Computer Technology (ICOT)" through joint investment with private enterprise to promote this project. The Fifth Generation Computer Project was established as a long-term plan extending over about 10 years. The project had achieved the desired objectives to a certain extent by the end of March, 1992 and was brought to a close in March, 1993 after one year of final unification, evaluation, and improvement work.

Japan learned many things from the major industrialized countries at the dawn of the computer age, and today's Japanese computer technology was fostered based on this knowledge. Making a contribution internationally through the execution of this new kind of project was an important role for Japan. Therefore, at the start of the Fifth Generation Computer Project, MITI widely appealed to overseas research institutes for involvement in the project. At the same time, interim results from the project were made public externally by holding international symposiums as necessary. To promote smooth execution of the project, JIPDEC dispatched many engineers to ICOT to offer their cooperation. AI played an important role in this project in the research of basic theory, etc. Regarding AI, trial introduction of expert systems has become popular in private enterprise, but it has been pointed out that AI cannot be expected to spread further because development techniques have not yet been established. Therefore, ICOT and JIPDEC established an ICOT-JIPDEC AI Center in 1986 to investigate, study, and disseminate AI technology. Its specific accomplishments include the publication of "The AI Vision", and "The AI White Paper", etc., and the sponsorship of various lectures and seminars.

The research and development phase of the Fifth Generation Computer Project was completed at the end of March, 1993, but the propagation of its results will continue to be important from now on. Therefore, for a period of two years starting in April, 1993 JIPDEC will make efforts to disseminate the parallel knowledge processing technology fostered in the project.

The history of research and development of the project, an outline of its results, and the prospects for the future, etc. will be introduced in this issue to commemorate the completion of the Fifth Generation Computer Project. In drawing up this summary, we received full-scale cooperation from ICOT. We would like to use this opportunity to acknowledge this support, and we hope this report will be of help to our readers.

[illegible]

Yuji Yamadori
Director
Research & International Affairs

Fifth Generation Computer Systems (FGCS) Project in Japan

Dr. Koichi Furukawa
Professor
Faculty of Environmental Information
Keio University

Overview of the FGCS Project

1. Preliminary Study Stage for the FGCS Project

The circumstances prevailing during the preliminary stage of the FGCS Project, from 1979 to 1981, can be summarized as follows:

Japanese computer technologies had reached the level where they are now among the most up-to-date overseas computer technologies.

A change of the role of the Japanese national project for computer technologies was being discussed whereby there would be a move away from improvement of industrial competitiveness by catching up with the latest European computer technologies and toward worldwide scientific contributions through the development of leading computer technologies with all its inherent risks.

Regarding this situation, the Japanese Ministry of International Trade and Industry (MITI) began study on a new project — the Fifth Generation Computer Project. This term expressed MITI's commitment to developing leading technologies that would progress beyond the fourth generation computers due to appear in the near future and which would anticipate upcoming trends.

The Fifth Generation Computer Research Committee and its subcommittee were established in 1979. It took until the end of 1981 to decide on target technologies and a framework for the project.

Well over one hundred meetings were held with a similar number of committee members participating. The following important near-future computer technologies were discussed:

- Inference computer technologies for knowledge processing
- Computer technologies to process large-scale data bases and knowledge bases
- High performance workstation technologies
- Distributed functional computer technologies
- Super-computer technologies for scientific calculation

These computer technologies were investigated and discussed from the standpoints of international contribution through the development of original Japanese technologies, the important technologies of the future, social needs and conformance with Japanese government policy for the national project.

Through these studies and discussions, the committee decided on the objectives of the project by the end of 1980, and continued future studies of technical matters, social impact, and project schemes.

The committee's proposals for the FGCS Project are summarized as follows:

(1) The concept of the Fifth Generation Computer: to have parallel (non-Von Neumann) processing and inference processing using knowledge bases as basic mechanisms. In order to possess these mechanisms, the hardware and software interface is to be a logic program language (see Figure 1).

(2) The objectives of the FGCS project: to develop these innovative computers which are capable of knowledge information processing and to overcome the technical restrictions of conventional computers.

(3) The goals of the FGCS project: to research and develop a set of hardware and software technologies for FGCS, and to develop an FGCS prototype system consisting of a thousand element processors with inference execution speeds of between 100M LIPS and 1G LIPS (Logical Inferences Per Second).

(4) R&D period for the project: estimated to be ten years, divided into three stages.

- 3-year initial stage for R&D of basic technologies
- 4-year intermediate stage for R&D of subsystems
- 3-year final stage for R&D of total prototype system

MITI decided to launch the Fifth Generation Computer System (FGCS) project as a national project for new information processing, and made efforts to acquire a budget for the project.

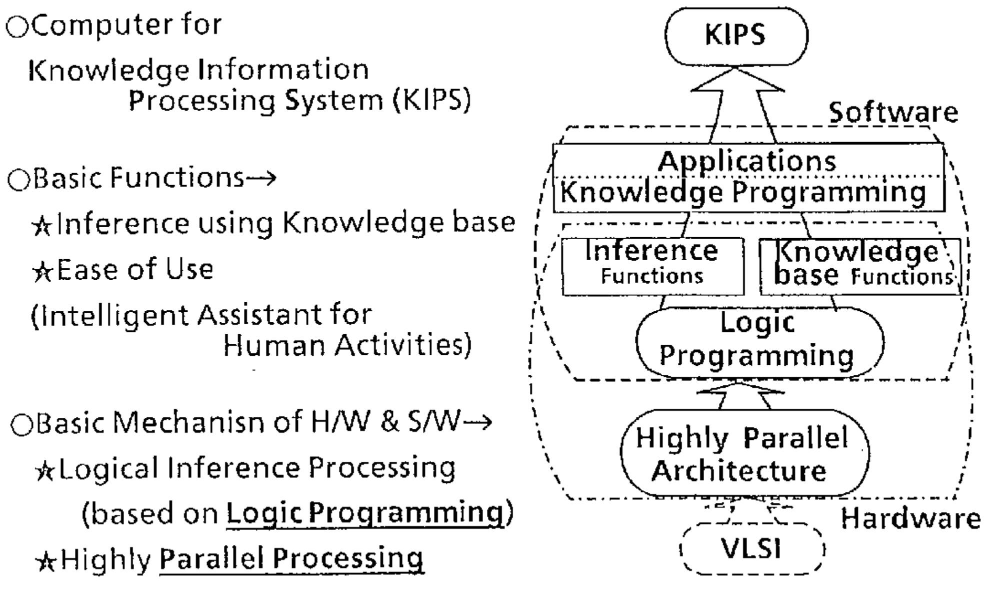

○Computer for Knowledge Information Processing System (KIPS)
○Basic Functions→
☆Inference using Knowledge base
☆Ease of Use
(Intelligent Assistant for Human Activities)
○Basic Mechanisn of H/W & S/W→
☆Logical Inference Processing
(based on Logic Programming)
☆Highly Parallel Processing
KIPS
Software
Applications
Knowledge Programming
Inference Functions
Knowledge base Functions
Logic Programming
Highly Parallel Architecture
Hardware
VLSI

Figure 1. Concept of the Fifth Generation Computer

At the same time, the international conference on FGCS '81 was prepared and held in October 1981 to announce these results and to hold discussions on the topic with foreign researchers.

2. Stages and Budgeting in the FGCS Project

The FGCS project was designed to investigate a large number of unknown technologies that were yet to be developed. Since this involved a number of risky goals, the project was scheduled over a relatively long period of ten years. This ten-year period was divided into three stages.

- In the initial stage (fiscal 1982 - 1984), the purpose of R&D was to develop the basic computer technologies needed to achieve the goal.

- In the intermediate stage (fiscal 1984 - 1988), the purpose of R&D was to develop small to medium subsystems.

- In the final stage (fiscal 1989 - 1992), the purpose of R&D was to develop a total prototype system. The final stage was initially planned to be three years. After reexamination halfway through the final stage, this stage was extended to four years to allow evaluation and improvement of the total system in fiscal year 1992. Consequently, the total length of this project has been extended to 11 years.

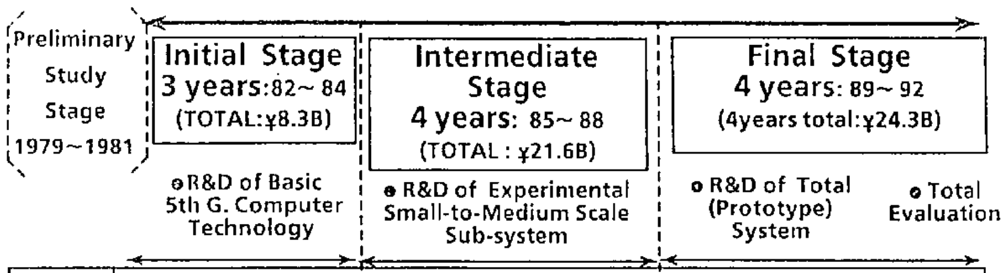
Each year the budget for the following years R&D activities is decided. MITI made strenuous efforts in negotiating each year's budget with the Ministry of Finance. The budgets for each year, which are all covered by MITI, are shown in Figure 2. The total budget for the 3-year initial stage was about 8 billion yen. For the 4-year intermediate stage, it was approximately 22 billion yen. The total budget for 1989 to 1991 was around 21 billion yen. The budget for 1992 is estimated to be 3.6 billion yen. Consequently, the total budget for the 11-year period of the project will be about 54 billion yen.

3. Summary of the Project Research Results

In the Fifth Generation Computer Project, two main research targets were pursued: knowledge information processing and parallel processing. Logic programming was adopted as a key technology for achieving both targets simultaneously. At the beginning of the project, we adopted Prolog as our vehicle to promote the entire research of the project. Since there were no systematic research attempts based on Prolog before our project, there were many things to do, including the development of a suitable workstation for the research, experimental studies for developing a knowledge-based system in Prolog and investigation into possible parallel architecture for the language. We rapidly succeeded in promoting research in many directions.

From this research, three achievements are worth noting. The first is the development of our own workstation dedicated to ESP: Extended Self-contained Prolog. We developed an operating system for the workstation completely in ESP [1].

The second is the application of partial evaluation to meta programming. This enabled us to develop a compiler for a new programming language by simply describing an interpreter of the language and then partially evaluating it. We applied this technique to derive a bottom-up parser for context-free grammar given a bottom


Preliminary Study Stage 1979~1981
Initial Stage 3 years:82~84 (TOTAL:¥8.3B)
Intermediate Stage 4 years: 85~88 (TOTAL : ¥21.6B)
Final Stage 4 years: 89~92 (4years total:¥24.3B)
R&D of Basic 5th G. Computer Technology
R&D of Experimental Small-to-Medium Scale Sub-system
R&D of Total (Prototype) System
Total Evaluation

Budget	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992
(for each	¥400M	¥2.7B	¥5.1B	¥4.7B	¥5.55B	¥5.6B	¥5.7B	¥6.5B	¥7.0B	¥7.2B	¥3.6B
fiscal	$1.86M*1	$12.6M	$23.7M	$21.9M	$34.5M*2	$35.0M	$35.6M	$40.6M	$43.7M	$51.4M*3	$25.7M
year)	£1.30M*1	£8.80M	£16.6M	£15.3M	£22.0M*2	£22.4M	£22.8M	£26.0M	£28.0M	£30.0M*3	£15.0M

10- year initial plan

- **R&D is carried out under the auspices of MITI.**
  (All budgets (Total budgets: ¥54.6B) are covered by MITI.)

*1 $1 = ¥215, £1 = ¥307 (1982~1985)
*2 $1 = ¥160, £1 = ¥250 (1986~1990)
*3 $1 = ¥140, £1 = ¥240 (1991~)

Figure 2. Budgets for the FGCS project

up interpreter for them. In other words, partial evaluation made meta programming useful in real applications.

The third achievement was the development of constraint logic programming languages. We developed two constraint logic programming languages: CIL and CAL. CIL is for natural language processing and is based on the incomplete data structure for representing "Complex Indeterminates" in situation theory. It has the capability to represent structured data like Minsky's frame and any relationship between slots' values can be expressed using constraints. CIL was used to develop a natural language understanding system called DUALS. Another constraint logic programming language, CAL, is for non-linear equations. Its inference is done using the Buchberger algorithm for computing the Grobner Basis which is a variant of the Knuth-Bendix completion algorithm for a term rewriting system.

We encountered one serious problem inherent to Prolog: that was the lack of concurrency in the fundamental framework of Prolog. We recognized the importance of concurrency in developing parallel processing technologies, and we began searching for alternative logic programming languages with the notion of concurrency.

We noticed the work by Keith Clark and Steve Gregory on Relational Language [2] and Ehud Shapiro on Concurrent Prolog [3]. These languages have a common feature of committed choice nondeterminism to introduce concurrency. We devoted our efforts to investigating these languages carefully and Ueda finally designed a new committed choice logic programming language called GHC [4] [5], which was simpler syntax than the above two languages but still has similar expressiveness. We recognized the importance of GHC and adopted it as the core of our kernel language, KL1, in this project. The introduction of KL1 made it possible to divide the entire research project into two parts: the development of parallel hardware dedicated to KL1 and the development of software technology for the language. In this respect, the invention of GHC is the most important achievement for the success of the Fifth Generation Computer Systems project.

Besides this language oriented research, we performed extensive fundamental research in the field of artificial intelligence and software engineering based on logic and logic programming. This includes research on non-monotonic reasoning, hypothetical reasoning, abduction, induction, knowledge representation, theorem proving, partial evaluation and program transformation. We expected that this research would become important application fields for our parallel machines by the affinity of these problems with logic programming and logic-based parallel processing. This is now happening.

In this article, we first describe the research and development of the sequential inference machine PSI. Then, we present our research results on constraint logic programming. Finally, we discuss our research activities in the field of parallel inference from both hardware and software aspects.

FGCS Project Research Results

(1) Personal Sequential Inference Machine (PSI-I)

To actually build the parallel inference system, especially a productive parallel programming environment which is now provided by PIMOS, we needed to develop various element technologies step by step to obtain hardware and software components. On the way toward this development, the most promising methods and technologies had to be selected from among many alternatives, followed by appropriate evaluation processes. To make this selection reliable and successful, we tried to build experimental systems which were as practical as possible.

In the initial stage, to evaluate the descriptive power and execution speed of logic languages, a personal sequential machine, PSI, was developed. This was a logic programming workstation. This development was also aimed at obtaining a common research tool for software development. The PSI was intended to attain an execution speed similar to DEC10 Prolog running on a DEC20 system, which was the fastest logic programming system in the world.

To begin with, a PSI machine language, KL0, was designed based on Prolog. Then a hardware system was designed for the KL0. We employed tag architecture for the hardware system. Then we designed a system description language, ESP, which is a logic language having a class and inheritance mechanisms to make program modules efficiently [6]. ESP was used not only to write the operating system for PSI, which is named SIMPOS, but also to write many experimental software systems for knowledge processing research.

The development of the PSI machine and SIMPOS was successful. We were impressed by the very high software productivity of the logic language. The execution speed of the PSI was about 35K LIPS and exceeded its target. However, we realized that we could improve its architecture by using the optimization capability of a compiler more effectively. We produced about 100 PSI machines to distribute as a common research tool. This version of the PSI is called PSI-I.

The implementation of the PSI-I hardware required 11 printed circuit boards. As the amount of hardware became clear, we established that we could obtain an element processor for a parallel machine if we used VLSI chips for implementation.

For the KL0 language processor which was implemented in the firmware, we estimated that better optimization of object code made by the compiler would greatly improve execution speed. (Later, this optimization was made by the introduction of the "WAM" code [7].

The PSI-II used VLSI gate array chips for its CPU. The size of the cabinet was about one sixth that of PSI-I. Its execution speed was 330K LIPS, about 10 times faster than that of PSI-I. This improvement was attained mainly through employment of the better compiler optimization technique and improvement of its machine architecture. The main memory size was also expanded to 320 MB so that prototyping of large applications could be done quickly.

FGCS Project Research Results

(2) Constraint Logic Programming

Constraint logic programming is one of the most promising areas in the field of logic programming. The domain of Prolog is extended to cover most AI problems. The objective is to combine constraint satisfaction and logic programming. The reasons for its success are 1) it is a straightforward extension of Prolog by extending the notion of unification to deal with constraint satisfaction, and 2) it extends the scope of declarative programming to a wider class of problems such as linear equations and inequations by dealing with them in a way uniform with unification between terms. From the constraint satisfaction viewpoint, it provides programming capability to the description of the problem based on constraint satisfaction. Jaffar and Lassez \cite{jaffar:clip} gave criteria for a constraint logic programming system to inherit important aspects of logic programming such as soundness, completeness and fixpoint semantics. It is worth noting that constraint logic programming is derived by extending unification. We will discuss later how concurrent logic programming is derived by restricting unification.

We began our constraint logic programming research almost at the beginning of our project, in relation to the research on natural language processing. Mukai [8] developed a language called CIL (Complex Indeterminates Language) for the purpose of developing a computational model of situation semantics. A complex indeterminate is a data structure allowing partially specified terms with indefinite arity. During the design phase of the language, he encountered the idea of freeze in Prolog II by Colmerauer [9]. He adopted freeze as a proper control structure for our CIL language.

From the viewpoint of constraint satisfaction, CIL only has a passive way of solving constraint, which means that there is no active computation for solving constraints such as constraint propagation or solving simultaneous equations. Later, we began our research on constraint logic programming involving active constraint solving. The language we developed is called CAL. It deals with non-linear equations as expressions to specify constraints. Three events triggered the research: one was our preceding efforts on developing a term rewriting system called METIS for a theorem prover of linear algebra [10]. Another event was our encounter with Buchberger's algorithm for computing the Grobner Basis for solving non-linear equations. Since the algorithm is a variant of the Knuth-Bendix completion algorithm for a term rewriting system, we were able to develop the system easily from our experience of developing METIS. The third event was the development of the CLP(X) theory by Jaffar and Lassez which provides a framework for constraint logic programming languages [11].

There is further remarkable research on constraint logic programming in the field of general symbol processing [12]. Tsuda developed a language called cu-Prolog. In cu-Prolog constraints are solved by means of program transformation techniques called unfold/fold transformation (these will be discussed in more detail later in this paper, as an optimization technique in relation to software engineering). The unfold/fold program transformation is used here as a basic operation for solving combinatorial constraints among terms. Each time the transformation is performed, the program is modified to a syntactically less constrained program. Note that this basic operation is similar to term rewriting, a basic operation in CAL. Both of these operations try to rewrite programs to obtain certain canonical forms. The idea of cu-Prolog was introduced by Hasida during his work on dependency propagation and dynamical programming [13]. They succeeded in showing that context-free parsing, which is as efficient as chart parsing, emerges as a result of dependency propagation during the execution of a program given as a set of grammar rules in cu-Prolog. Actually, there is no need to construct a parser. cu-Prolog itself works as an efficient parser.

Hasida [13] has been working on a fundamental issue of artificial intelligence and cognitive science from the aspect of a computational model. In this computation model of dynamical programming, computation is controlled by various kinds of potential energies associated with each atomic constraint, clause, and unification. Potential energy reflects the degree of constraint violation and, therefore, the reduction of energy contributes to constraint resolution.

Constraint logic programming greatly enriched the expressiveness of Prolog and is now providing a very promising programming environment for applications by extending the domain of Prolog to cover most AI problems.

FGCS Project Research Results

(3) Parallel Inference System

1. FGHC

The most important feature of FGHC is that there is only one syntactic extension to Prolog called the commitment operator and is represented by a vertical bar "|". A commitment operator divides an entire clause into two parts called the guard part (the left-hand side of the bar) and the body part (the right-hand side).

FGHC is a subset of GHC and allows only unification and test predicates to be written in its guard part to prevent the guard from being nested during execution. FGHC is more amenable to the process interpretation of programs than GHC and is expressive enough.

The guard of a clause has two important roles: one is to specify a condition for the clause to be selected for the succeeding computation, and the other is to specify the synchronization condition. The general rule of synchronization in FGHC is expressed as dataflow synchronization. This means that computation is suspended until sufficient data for the computation arrives. In the case of FGHC, guard computation is suspended until the caller is sufficiently instantiated to judge the guard condition. For example, consider how a ticket vending machine works. After receiving money, it has to wait until the user pushes a button for the destination. This waiting is described as a clause such that "if the user pushed the 160-yen button, then issue a 160-yen ticket". The important thing is that dataflow synchronization can be realized by a simple rule governing head unification which occurs when a goal is executed and a corresponding FGHC clause is called. This rule is stated as follows: the information flow of head unification must be one way, from the caller to the callee.

The caller corresponds to a job scheduler and the callee corresponds to workers in the office. The dataflow corresponds to the job orders from the job scheduler to workers. Each worker has a speciality and conditions about the jobs that can be done. In this case, each worker has to wait for the arrival of a detailed job order before starting work in order to check the conditions. Note that the information flow of job orders is one way -- from the job scheduler to workers.

This principle is very important in two aspects: one is that the language provides a natural tool for expressing concurrency, and the other is that the synchronization mechanism is simple enough to realize very efficient parallel implementation.

2. KL1

KL1 (Kernel Language version 1) is an extension to FGHC and consists of two sublanguages, KL1c (KL1 core), and KL1p (KL1 pragma).

KL1 enables a process to observe and control the execution of other processes at the meta-level, a feature needed in developing our operating system PIMOS for Multi-PSI and PIM. KL1p is a sublanguage for annotating map information; allocation of processes to processors and priorities among processes.

While FGHC programs represent concurrency independently from the machine architecture on which they are executed, KL1p maps them onto parallel processors. Even if the FGHC programs express high potential parallelism, it does not necessarily mean that they run fast on existing parallel hardware. The important issues to be considered in achieving good performance on parallel hardware include: load balancing, locality of communication and priority control. Note that these issues affect only efficiency, not correctness. Although it is desirable to automate mapping, there is no universal way to do this appropriately. We are investigating various techniques of load balancing and priority control. We succeeded in developing automatic load balancing for a class of search problems [14]. This indicates the possibility of extending the problem domains amenable to automatic load balancing.

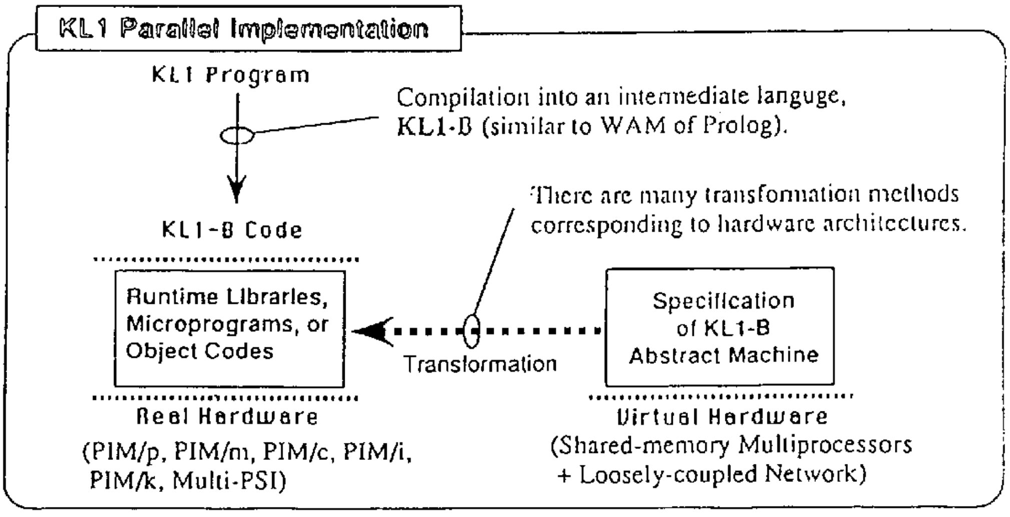
Communication costs are very high, especially in distributed memory architecture. Primitive operations like unification are around 10 to 100 times slower when they are performed across the interconnection network. We need to take communication overheads into account when we distribute loads to processors. The locality of computation is important, and if granularity is insufficient, performance may decrease. In KL1, we can adjust granularity and increase locality by appropriately specifying load distribution using the annotation facilities in KL1p.

3. PIM hardware and KL1 language processor

To find an optimal architectural design, we selected several important features, such as element processor architecture and network structure, and decided to build five PIM modules having different design choices. The main features of these five modules are listed in Table 1. The number of element processors required for each module was determined depending on the main purpose of the module. Large modules have 256 to 512 element processors, and were intended to be used for software experiments. Small modules have 16 or 20 element processors and were built for architectural experiments and evaluation.

Table 1. Features 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	60 nsec	65 nsec	50 nsec	100 nsec	100 nsec
LSI devices	Standard cell	Gate array	Cell base	Standard cell	Custom
Process Technology (line width)	0.96 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	16 PEs	16 PEs



KL1 Parallel Implementation
KL1 Program
Compilation into an intermediate languge, KL1-B (similar to WAM of Prolog).
KL1-B Code
There are many transformation methods corresponding to hardware architectures.
Runtime Libraries, Microprograms, or Object Codes
Transformation
Specification of KL1-B Abstract Machine
Real Hardware
(PIM/p, PIM/m, PIM/c, PIM/i, PIM/k, Multi-PSI)
Virtual Hardware
(Shared-memory Multiprocessors + Loosely-coupled Network)

Figure 1. KL1 Language Processor and VPIM

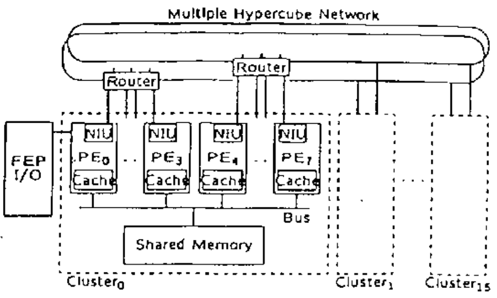
All of these modules were designed to support KL1 and PIMOS, so that software researchers could run one program on the different modules and compare and analyze the behaviors of parallel program execution.

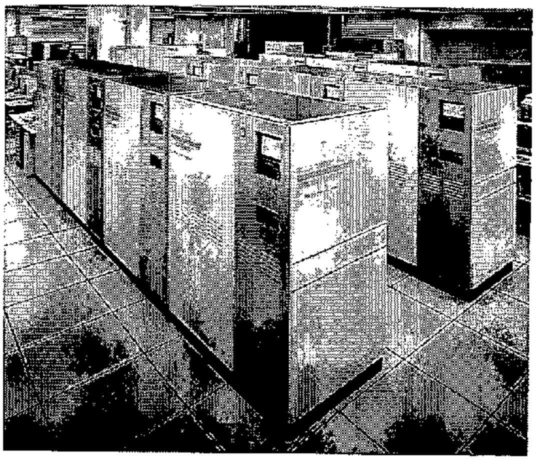
A PIM/m module employed architecture similar to the multi-PSI system. Thus, its KL1 language processor could be developed by simply modifying and extending that of the multi-PSI system. For other modules, namely PIM/p, PIM/c, PIM/k, and PIM/i, the KL1 language processor had to be newly developed because all of these modules have a cluster structure. In a cluster, four to eight element processors were tightly coupled by a shared memory and a common bus with coherent caches. While communication between element processors is done through the common bus and shared memory, communication between clusters is done via a packet switching network. These four PIM modules have different machine instruction sets.

We intended to avoid the duplication of development work for the KL1 language processor. We used the KL1-C language to write PIMOS and the usual application programs. A KL1-C program is compiled into the KL1-B language, which is similar to the "WAM" as shown in Figure 1. We defined an additional layer between the KL1-B language and the real machine instruction. This layer is called the virtual hardware layer. It has a virtual machine instruction et called "PSL." The specification of the KL1-B interpreter is described in PSL. This specification is semiautomatically converted to a real interpreter or runtime routines dedicated to each PIM modules. The specification in PSL is called a virtual PIM processor (the VPIM processor for short) and is common to four PIM modules.

PIM/p, PIM/m and PIM/c are intended to be used for large software experiments; the other modules were intended for architectural evaluations. We plan to produce a PIM/p with 512 element processors, and a PIM/m with 384 element processors. Now, at the beginning of March 1992, a PIM/m of 256 processors has just started to run a couple of benchmarking programs. The main features and appearances of PIM/p and PIM/m are shown in Figure 2 and Figure 3, respectively.

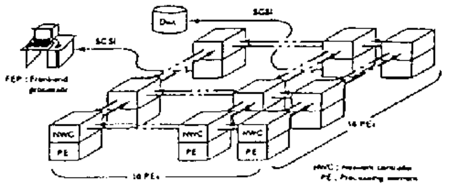
We aimed at a processing speed of more than 100 MLIPS for the PIM modules. The PIM/m

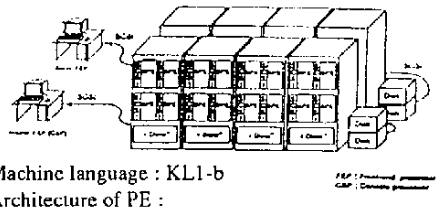


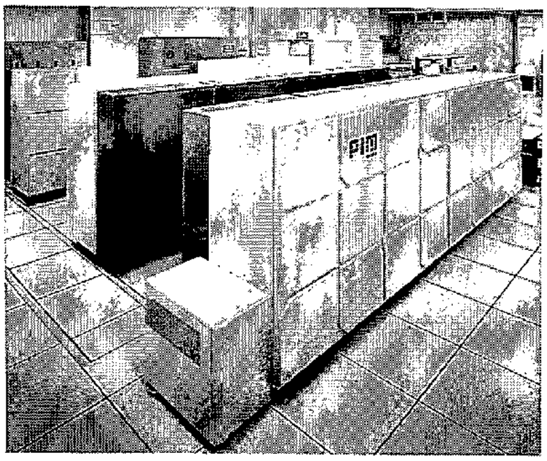


- Machine language : KL1-b
- Architecture of PE and cluster
  - RISC + HLIC (Microprogrammed)
  - Machine cycle: 60ns, Reg. file: 40bits × 32W
  - 4 stage pipeline for RISC inst.
  - Internal Inst. Mem: 50 bits × 8KW
  - Cache: 64 KB, 256 column, 4 sets, 32B/block
  - Protocol: Write-back, Invalidation
  - 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. 20 MB/sec in each link

Figure 2. PIM Model P: Main Features and Appearance of A Cabinet







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

Figure 3. PIM Mode M: Main Features and Appearance of Four Cabinets

with 256 processors will attain more than 100 MLIPS as its peak performance. However, for a practical application program, this speed may be much reduced, depending on the characteristics of the application program and the programming technique. To obtain better performance, we must attempt to augment the effect of compiler optimization and to implement a better load balancing scheme. We plan to run various benchmarking programs and experimental application programs to evaluate the gain and loss of implemented hardware and software functions.

4. Development of PIMOS

PIMOS was intended to be a standard parallel operating system for large-scale parallel machines used in symbol and knowledge processing. It was designed as an independent, self-contained operating system with a programming environment suitable for KL1. Its functions for resource management and execution control of user programs were designed as independent from the architectural details of the PIM hardware. They were implemented based on an almost completely non-centralized management scheme so that the design could be applied to a parallel machine with one million element processors [15].

PIMOS is completely written in KL1. Its management and control mechanisms are implemented using a "meta-call" primitive of KL1. The KL1 language processor has an embedded automatic memory management mechanism and a dataflow synchronization mechanism. The management and control mechanisms are then implemented over these two mechanisms.

The resource management function is used to manage the memory resources and processor resources allocated to user processes and input and output devices. The program execution control function is used to start and stop user processes, control the order of execution following priorities given to them, and protect system programs from user program bugs like the usual sequential operating systems.

PIMOS supports multiple users, accesses via network and so on. It also has an efficient KL1 programming environment. This environment has some new tools for debugging parallel programs such as visualization programs which show a programmer the status of load balancing in graphical forms, and other monitoring and measurement programs.

5. Knowledge base management system

The knowledge base management system consists of two layers. The lower layer is a parallel database management system, Kappa-P. Kappa-P is a database management system based on a nested relational model. It is more flexible than the usual relational database management system in processing data of irregular sizes and structures, such as natural language dictionaries and biological databases.

The upper layer is a knowledge base management system based on a deductive object-oriented database [16]. This provides us with a knowledge representation language, Quixote [17]. These upper and lower layers are written in KL1 and are now operational on PIMOS.

The development of the database layer, Kappa, was started at the beginning of the intermediate stage. Kappa aimed to manage the "natural databases" accumulated in society, such as natural language dictionaries. It employed a nested relational model so that it could easily handle data sets with irregular record sizes and nested structures. Kappa is suitable not only for natural language dictionaries but also for DNA databases, rule databases such as legal data, contract conditions, and other "natural databases" produced in our social systems.

The first and second versions of Kappa were developed on a PSI machine using the ESP language. The second version was completed at the end of the intermediate stage, and was called Kappa-II [18].

In the final stage, a parallel and distributed implementation of Kappa was begun. It is written in KL1 and is called Kappa-P. Kappa-P is intended for the use of large PIM main memories for implementing the main memory database scheme, and to obtain a very high throughput rate for disk input and output by using many disks connected in parallel to element processors.

In conjunction with the development of Kappa-II and Kappa-P, research on a knowledge representation language and a knowledge base management system was conducted. After repeated experiments in design and implementation, a deductive object-oriented database was employed in this research.

At this point the design of the knowledge representation language, Quixote, was completed. Its language processor, which is the knowledge base management system, is under development. This language processor is being built over Kappa-P. Using Quixote, construction of a knowledge base can then be made continuously from a simple database. This will start with the accumulation of passive fact data, then gradually add active rule data, and will finally become a complete knowledge base.

The Quixote and Kappa-P system is a new knowledge base management system which has a high-level knowledge representation language and the parallel and distributed database management system as the base of the language processor. The first versions of Kappa-P and Quixote are now almost complete. It will be interesting to see how this large system operates and how much of an overhead it will require.

FGCS Project Research Results

(4) Concurrent Logic Programming

In this section, we first present the methodology to realize search paradigm in FGHC/KL1. Then, we describe three application programs in FGHC/KL1: a routing problem in VLSI CAD, a sequence alignment problem in genetic information processing, and a bottom-up theorem prover. These items are very different from each other in their application areas as well as in the programming techniques used in their development.

1. Search Paradigms in FGHC/KL1

There is one serious drawback to FGHC/KL1 because of the very nature of committed choice; that is, it no longer has an automatic search capability, which is one of the most important features of Prolog. Prolog achieves its search capability by means of automatic backtracking. However, since committed choice uniquely determines a clause for succeeding computation of a goal, there is no way of searching for alternative branches other than the branch selected. The search capability is related to the notion of completeness of the logic programming computation procedure and the leak of this capability is very serious in that respect.

One could imagine a seemingly trivial way of realizing search capability by means of or-parallel search: that is, to copy the current computational environment which provides the binding information of all variables that have appeared so far and to continue computations for each alternative case in parallel. But this does not work because copying non-ground terms is impossible in FGHC/KL1. The reason why it is impossible is that FGHC/KL1 cannot guarantee when actual binding will occur and there may be a moment when a variable observed at some processor remains unchanged even after some goal has instantiated it at a different processor.

One might ask why we did not adopt a Prolog-like language as our kernel language for parallel computation. There are mainly two reasons. One is that, as stated above, Prolog does not have enough expressiveness for concurrency, which we see as a key feature not only for expressing concurrent algorithms but also for providing a framework for the control of physical parallelism. The other reason is that the execution mechanism of Prolog-like languages with a search capability seemed too complicated to develop efficient parallel implementations.

We tried to recover the search capability by devising programming techniques while keeping the programming language as simple as possible. We succeeded in inventing several programming methods for computing all solutions to a problem which effectively achieve the completeness of logic programming. Three of these methods are listed as follows:

(1) Continuation-based method [19]
(2) Layered stream method [20]
(3) Query compilation method [21]

In this paper we pick up (1) and (3), which are complementary to each other. The continuation-based method is suitable for the efficient processing of rather-algorithmic problems. An example is to compute all ways of partitioning a given list into two sublists by using append. This method mimics the computation of OR-parallel Prolog using AND-parallelism of FGHC. AND-serial computation in Prolog is translated to continuation processing which remembers continuation points in a stack. The intermediate results of computation are passed from the preceding goals to the next goals through the continuation stack kept as one of the arguments of the FGHC goals. This method requires input/output mode analysis before translating a Prolog program into FGHC. This requirement makes the method impractical for database applications because there are too many possible input-output modes for each predicate.

The query compilation method solves this problem. This method was first introduced by Fuchi [22] when he developed a bottom-up theorem prover in KL1. In this coding technique, the multiple binding problem is avoided by reversing the role of the caller and the callee in straightforward implementation of database query evaluation. Instead of trying to find a record (represented by a clause) which matches a given query pattern represented by a goal, his method represents each query component with a compiled clause, represents a database with a data structure passed around by goals, and tries to find a query component clause which matches a goal representing a record and recurses the process for all potentially applicable records in the database*). Since every record is a ground term, there is no variable in the caller. Variable instantiation occurs when query component clauses are searched and an appropriate clause representing a query component is found to match a currently processed record. Note that, as a result of reversing the representation of queries and databases from straightforward representation, the information flow is now from the caller (database) to the callee (a query component). This inversion of information flow avoids deadlock in query processing. Another important trick is that each time a query clause is called, a fresh variable is created for each variable in the query component. This mechanism is used for making a new environment for each OR-parallel computation branch. These tricks make it possible to use KL1 variables to represent object level variables in database queries and, therefore, we can avoid different compilations of the entire database and queries for each input/output mode of queries.

The new coding method stated above is very general and there are many applications which can be programmed in this way. The only limitation to this approach is that the database must be more instantiated than the queries. In bottom-up theorem proving, this requirement is referred to as the range-restrictedness of each axiom. Range-restrictedness means that, after successfully finding ground model elements satisfying the antecedent of an axiom, the new model element appearing as the consequent of

*) We need an auxiliary query clause which matches every record after failing to match the record to all the real query clauses.

the axiom must be ground.

This restriction seems very strong. Indeed, there are problems in the theorem proving area which do not satisfy the condition. We need a top-down theorem prover for such problems. However, many real life problems satisfy the range-restrictedness because they almost always have finite concrete models. Such problems include VLSI-CAD, circuit diagnosis, planning, and scheduling.

2. A Routing Problem in VLSI CAD

The aim of the routing problem is to determine connection paths between terminals of circuit blocks on a VLSI surface. Well-known routing methods include maze routing, line searching and channel routing. We adopted an extended line searching algorithm called look-ahead line searching [23].

Before introducing the details of the algorithm and its implementation in KL1, let us explain the problem more precisely as illustrated in Figure 1. We assume that there are two layers for connection in VLSI chips: the first surface and the second surface. Let us assume that the first surface is for vertical lines and the second for horizontal lines. There are two kinds of constraints to connection: the first are surface obstacles like Block 1, Block 2 and Block 3 in

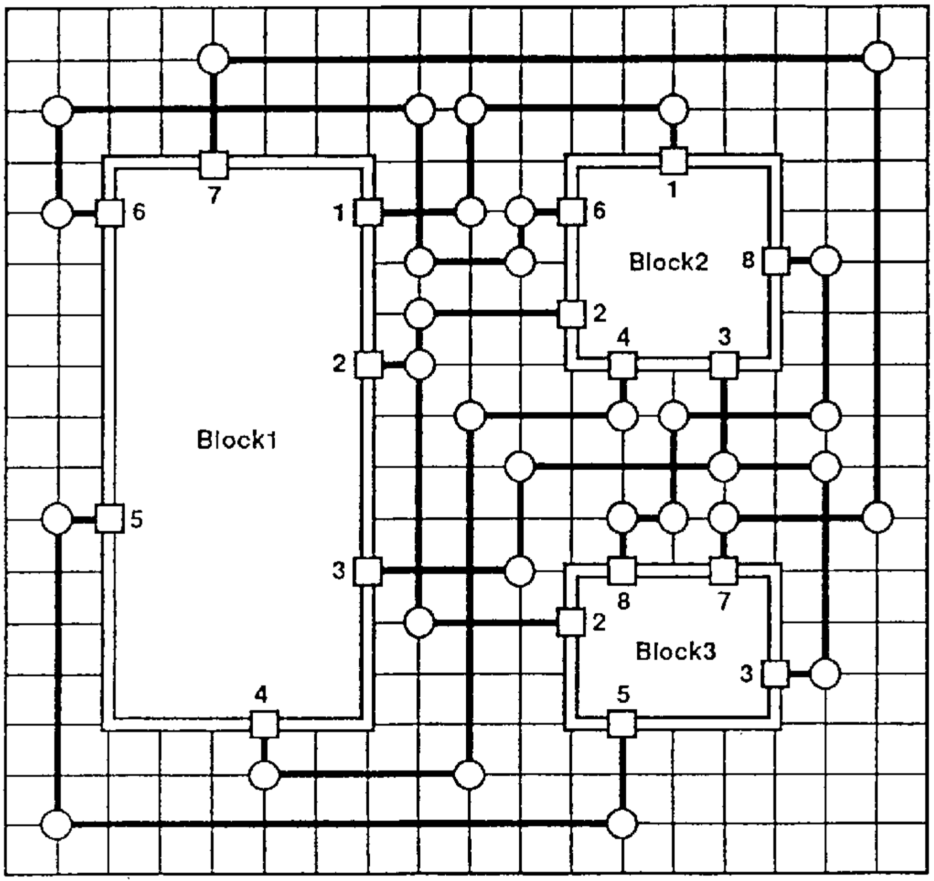


Figure 1. A Routing Problem and Its Solution

Figure 1, inside which wiring is prohibited, the other are via-hole constraints which prevent use of via-hole constraints to change direction (go from one surface to the other). A routing problem is given by a set of nets, which are sets of terminals to be connected (a net is shown by a set of terminals with the same number as in Figure 1).

The look-ahead line searching method tries to find a route for a given pair incrementally by finding the best position to turn based on a local estimate. Estimation is done by computing the next nearest points to the target point for each possible turn and selecting the closest as the best. Since the method tries to compute all possible turns to determine the next turn, it is called look-ahead line searching.

To implement the look-ahead line searching method in KL1, we adopted the object-oriented programming paradigm. Each line segment is represented by an object. To determine the turning point of the current line segment, a message to compute the distance between the nearest attainable point and the target point is sent to all candidate lines intersecting the current line. After receiving all answers from all candidates, the current line object determines the point having the shortest distance to the target as the next turning point. Then, the selected line segment becomes the new current line segment and the same process is repeated.

A line object is realized in KL1 by a recursive program like filter in Erastothenes' sieve algorithm. Note that a current line segment is dynamically divided into a fixed route part and unused (free) parts. Preliminary evaluation showed 16-fold speed-up with 64 processors (compared to a single processor), and comparable execution time with a high-end general purpose computer. The speed will be 20 times faster on PIM/p.

3. Sequence Alignment in Genome Analysis

Sequence alignment is a very important yet time consuming task in genetic information processing. It is difficult to find the best alignment of amino-acid sequences for more than one protein. There are two ways of aligning different sequences: one is to match two different amino-acids with a cost associated to each pair of amino-acids reflecting the similarity between them, the other is to match an amino-acid to a gap. Generally, the cost of matching two different amino-acids is less than that of matching an amino-acid to a gap. A well-known algorithm for aligning two amino sequences is two dimensional dynamic programming (DP) matching. Let us explain the algorithm for a simple case of aligning two four-letter sequences: ADHE and AHIE. The problem is formulated as the problem of finding the shortest path from the top-left corner to the bottom-right corner in the graph in Figure 2.

The DP matching algorithm proceeds from the top-left corner to the bottom-right corner. For each (ij) node, it computes the distance, Dij, of the two sequences from the top-left corner to the point using the following formula:

$$D_{i,j} = min \begin{pmatrix} D_{i,j-1} + gapcost, \\ D_{i-1,j} + gapcost, \\ D_{i-1,j-1} + Dayhoff \\ \quad matrix(X, Y) \end{pmatrix}$$

where X and Y are amino-acids associated with the arc from the node of (i-1, j-1) to that of (i, j), and the Dayhoff-matrix is a cost matrix of pairs of amino-acids. Note that this formula represents a simple expression of dynamic program-

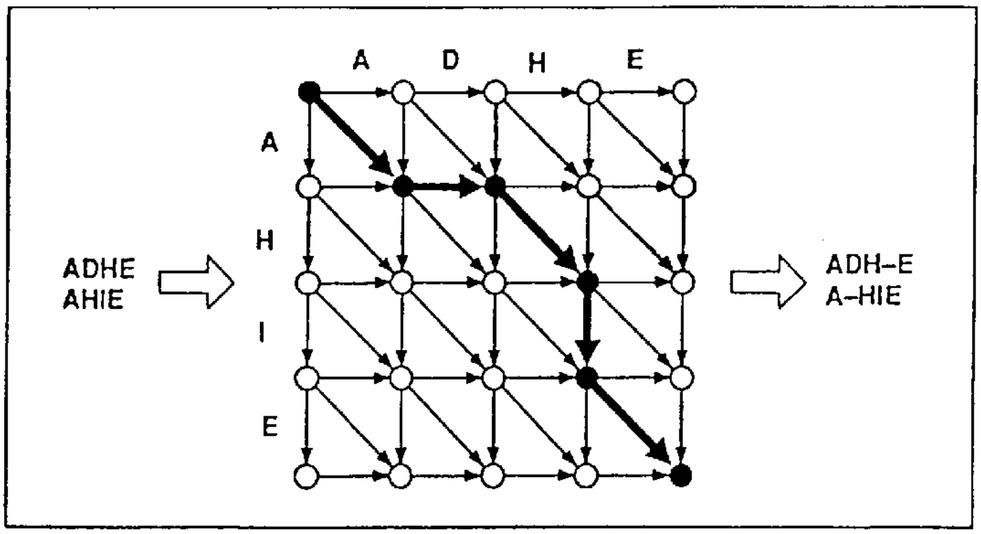


Figure 2 Two Dimensional DP Matching for Shortest Path Finding

ming which can compute each cost locally and prunes possibilities other than the minimum one.

Implementation of this algorithm in KL1 is straightforward: first, a set of processes representing each node is created, and then each Dij is computed by this expression in a data driven style from top-left to bottom-right.

We developed a three-dimensional DP matching program in KL1 [24]. Ideally we need N dimensional DP matching where N is the number of sequences to be aligned. However, the computational complexity grows exponentially and, therefore, such an extension is not feasible. Thus, we tried to merge multiple three-dimensional alignments by aligning similar sequences of different alignments. Then, recognizing that the alignments were not optimal, we tested different gaps to maximize alignment scores. Furthermore, we applied a simulated annealing procedure to avoid the local optimum and attain a further increase in alignment scores. The improvement in alignment scores by simulated annealing was surprisingly good, resulting in a relatively short execution time.

4. Theorem Prover

Since head unification is limited to one-way in GHC/KL1, it is generally difficult to implement a first order theorem prover since these usually require the extensive use of full unification. There are two ways to solve this problem. One is to write a full unification program in GHC/KL1 and regard the language as a very low-level language like C. The result is not very attractive because it decreases efficiency by 10 to 100 times that with direct use of unification (in the case of one way unification).

The other is to limit the usage of unification to only one way. Man-they and Bry proposed a new bottom up theorem prover, called SATCHMO, based on model generation [25]. SATCHMO tries to generate all possible models incrementally by bottom-up evaluation of clauses. It tries to prove the antecedent conjunction of literals by searching for their instances in each model. If it succeeds in finding a model in which the proof succeeds, then it extends the model by adding the consequent disjunctive ground literals. If there is more than one literal in its disjunct, then SATCHMO splits the model into the number of literals in the disjunct, and adds each literal to each split model. Since every element in every model is ground, we need only one-way unification to search models during the proof of antecedent conjuncts. This enables us to implement an efficient theorem prover in GHC/KL1.

One significant problem was how to implement multiple bindings for finding all possible proofs by different instantiations of variables. In the case of Prolog, this function is achieved by utilizing backtracking. However, there is no backtracking mechanism in GHC/KL1. Fuchi [26] invented an elegant coding technique when implementing SATCHMO in FGHC. Fuchi utilized FGHC variables as object variables appearing in the theories to be proved. This was improved by Fujita and Hasegawa [27]. In their coding technique, the multiple binding problem is avoided by reversing the role of the caller and the callee in naive implementation of database query evaluation: instead of trying to find a model element (a database item) with a pattern appearing in a theorem (a query pattern), their method tries to find a theorem (a query component) with a given model element (a database item) as an instantiation of a literal of the theorem. Since every model element is a ground literal, there is no variable in the caller. The variable instantiations occur when a theorem database is searched and an appropriate clause representing a literal of some theorem is found to match a given model element.

We tried to apply this theorem prover to solve several hard problems on our PIM with 256 processors and, for some problems, we obtained approximately 200 times performance improvement compared to the run time on a single processor.

We have been applying the theorem prover to such divergent problems as mathematical theorem proving, legal reasoning, design problems and syntax analysis. In particular, we have succeeded in solving several theorems of semigroup which were not solved earlier.

FGCS Follow-on Project & Forecasts

1. FGCS Follow-on Project

As described in the above, the FGCS Follow-on Project is a two year project which runs from the beginning of fiscal 1993 until the end of fiscal 1994. A major role of the FGCS Follow-on Project is to promote a diffusion of parallel knowledge processing technologies that have been developed in the FGCS Project.

Much of the KL1 software which aims at the provision of the new infrastructure for advanced computer research has been developed for research on parallel knowledge processing technologies in the FGCS Project. Moreover, the major software has been released as IFS. However, a sequential inference machine, PSI, or parallel inference machine, Multi-PSI or PIM, is required to execute the software. Though a "Pseudo Multi-PSI," that is, a pseudo parallel system for KL1 software, has been released as IFS, a PSI-III is required to execute it. A "PDSS", that is a KL1 programming environment on UNIX machines, has also been released as IFS, there are some limits to its efficiency and functions for executing KL1 software on it. Thus, although the PDSS system is suitable for learning KL1 language, it cannot be used to execute large KL1 software released as IFS. Therefore, it is difficult to execute the KL1 software released as IFS at hand.

In the FGCS Follow-on Project a series of KL1 programming environments, including a KL1 language processor and a parallel operating system, PIMOS, shall be ported onto sequential and parallel UNIX machines so that they can be used easily at any site. These UNIX based KL1 programming environments shall be designed as machine-independent as possible. They are also planned to be released as IFS.

An experimental version of the UNIX based KL1 programming environment is currently under development for evaluating an implementation scheme. Although we plan to release it as IFS in April '93, this version is for language implementation experts and is not suitable for application users since it lacks some important features for application users, such as debugging aids.

The first version for application users is planned release in September '93. This version shall provide reasonable software development functions, including debugging and performance analyses. This system shall be ten times faster than the PDSS, although it is for single processor UNIX machines.

The release of a KL1 programming environment for parallel UNIX machines is planned for the second quarter of '94. It shall be designed avoiding the use of machine-dependent functions. Various improvements are planned after these releases.

2. Forecasts for Some Aspects of 5G Machines

LSI technologies have advance in accordance with past trends. Roughly speaking, the memory capacity and the number of gates of a single chip quadruple every three years. The number of boards for the CPU of an inference machine was

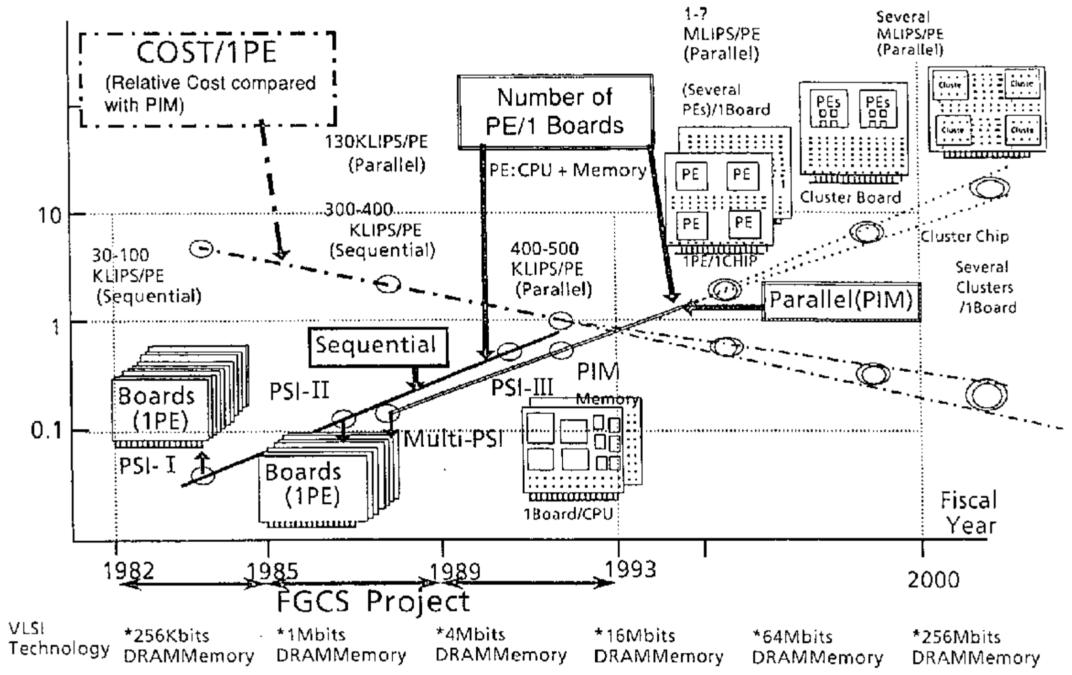


Figure 1. Size and Cost Trends of 5G Machines

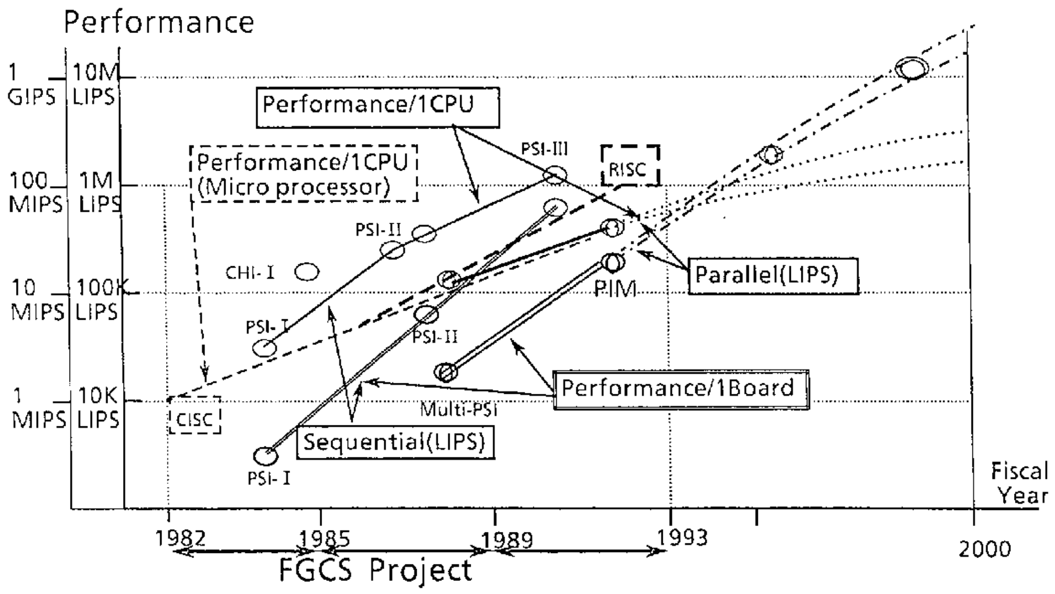


Figure 2. Performance Trends of 5G Machines

more than ten for PSI-I, but only three for PSI-II, and a single board for PIM.

The number of boards for 80M bytes memory was 16 for PSI-I, but only four for PSI-II, and a single board for PIM (m).

Figure 1 shows the anticipated trend for board numbers for one PE (processor element: CPU and memory) and the cost of one PE based on the actual value of inference machines developed by this project.

The trend reveals that by the year 2000 approximately ten PEs will fit on one board, around 100 PEs will fit in one desk side cabinet, and 500 to a 1,000 PEs will fit into a large cabinet. This trend also shows that the cost of one PE will have every three years.

Figure 2 shows the performance trends for 5G machines based on the actual performance of inference machines developed by this project.

The sequential inference processing performance for one PE quadrupled every three years. The improvement in parallel inference processing performance for one PE was not as large as it was for sequential processing, because PIM performance is estimated at around two and one half times that of multi-PSI. Furthermore, Figure 2 shows the performance of one board for both sequential and parallel processing, and the performance of a conventional micro-processor with CISC and RISC technology. In this figure, future improvements in the performance of one PE are estimated to be rather lower than a linear extension of past values would indicate because of the uncertainty of whether future technology will be able to elicit such performance improvements. Performance for one board is estimated at about 20 MLIPS, which is 100 times faster than PIM. Thus, a parallel machine with a large cabinet size could have 1 GLIPS. These parallel systems will have the processing speeds needed for various knowledge processing applications in the near future.

Several parallel applications in this project, such as CAD, theorem provers, and genetic information processing, natural language processing, and legal reasoning were described previously. These applications are distributed in various fields and aim at cultivating new parallel processing application fields.

We believe that parallel machine applications will be extended to various areas in industry and society, because parallel technology will become common for computers in the near future. Parallel application fields will expand gradually according to function expansion by the use of advanced parallel processing and knowledge processing technologies.

References

[1] T. Chikayama, Programming in ESP - Experiences with SIMPOS -, In Programming of Future Generation Computers, Fuchi and Nivat (eds.), North-Holland, 1988.

[2] K. L. Clark and S. Gregory, A Relational Language for Parallel Programming. In Proc. ACM Conf. on Functional Programming Languages and Computer Architecture, ACM, 1981.

[3] E. Y. Shapiro, A Subset of Concurrent Prolog and Its Interpreter. Tech. Report TR-003, Institute for New Generation Computer Technology, Tokyo, 1983.

[4] K. Ueda, Guarded Horn Clauses. In Logic Programming '85, E. Wada (ed.), Lecture Notes in Computer Science, 221, Springer-Verlag, 1986.

[5] K. Ueda and T. Chikayama, Design of the Kernel Language for the Parallel Inference Machine. The Computer Journal, Vol. 33, No. 6, pp. 494-500, 1990.

[6] T. Chikayama, "Unique Features of ESP", In Proc. Int. Conf. on Fifth Generation Computer Systems 1984, ICOT, 1984, pp. 292-298.

[7] D. H. D. Warren, An Abstract Prolog Instruction Set. Technical Note 304, Artificial Intelligence Center, SRI, 1983.

[8] K. Mukai, and H. Yasukawa, Complex Indeterminates in Prolog and its Application to Discourse Models. New Generation Computing, Vol. 3, No. 4, 1985.

[9] A. Colmerauer, Theoretical Model of Prolog II. In Logic Programming and Its Applications, M. Van Caneghem and D. H. D. Warren (eds.), Albex Publishing Corp, 1986.

[10] A. Ohsuga and K. Sakai, Metis: A Term Rewriting System Generator. In Software Science and Engineering, I. Nakata and M. Hagiya (eds.), World Scientific, 1991.

[11] J. Jaffar and J-L. Lassez, Constraint Logic Programming. Technical Report, Department of Computer Science, Monash University, 1986.

[12] H. Tsuda, cu-Prolog for Constraint-based Grammar. In Proc. of the International Conf. on Fifth Generation Computer Systems 1992, Tokyo, 1992.

[13] K. Hasida, Dynamics of Symbol Systems - An Integrated Architecture of Cognition. In Proc. of the International Conf. on Fifth Generation Computer Systems 1992, Tokyo, 1992.

[14] Furuichi, M., Taki, K. and Ichiyoshi, N. A Multi-Level Load Balancing Scheme for OR-Parallel Exhaustive Search Programs on the Multi-PSI. In

Proc. of the 2nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 1990.

[15] T. Chikayama, "Operating System PIMOS and Kernel Language KL1", Proc. Int. Conf. on Fifth Generation Computer Systems, Tokyo, Jul.1-5, 1992.

[16] K. Yokota and S. Nishio, "Towards Integration of Deductive Databases and Object-Oriented Databases-A Limited Survey", Proc. Advanced Database System Symposium, Kyoto, Dec., 1989.

[17] K. Yokota and H. Yasukawa, "Towards an Integrated Knowledge-Base Management System", Proc. Int. Conf. on Fifth Generation Computer Systems, Tokyo, Jul.1-5, 1992.

[18] K. Yokota, M. Kawamura, and A. Kanaegami, "Overview of the knowledge Base Management System (KAPPA)", Proc. Int. Conf. on Fifth Generation Computer Systems, Tokyo, Nov.28-Dec.2, 1988.

[19] K. Ueda, Making Exhaustive Search Programs Deterministic. In Proc. of the Third Int. Conf. on Logic Programming, Springer-Verlag, 1986.

[20] Akira Okumura and Yuji Matsumoto, Parallel Programming with Layered Streams, In Proc. 1987 International Symposium on Logic Programming, pp. 224-232, San Francisco, September 1987.

[21] K. Furukawa, Logic Programming as the Integrator of the Fifth Generation Computer Systems Project, Communication of the ACM, Vol. 35, No. 3, 1992.

[22] K. Fuchi, An Impression of KL1 Programming - from my experience with writing parallel provers -. In Proc. of KL1 Programming Workshop '90, ICOT, 1990 (in Japanese).

[23] Date, H., Ohtake, Y. and Taki, K. A parallel router based on a concurrent object model. In Proc. of the Logic Programming Conference, ICOT, 1991 (in Japanese).

[24] Ishikawa, M., Hoshida, M., Hirosawa, M., Toya, T., Onizuka, K., Nitta, K. and Kanehisa, M. Protein sequence analysis by parallel inference machine. Technical Report TR-681, ICOT, 1991 (in Japanese).

[25] Manthey, R. and Bry, F. SATCHMO: A theorem prover implemented in Prolog. In Proc. of CADE-88, Argonne, Ill., 1988.

[26] Fuchi, K. An Impression of KL1 programming - from my experience with writing parallel provers. In Proc. of KL1 Programming Workshop, ICOT, 1990 (in Japanese).

[27] Fujita, J. and Hasegawa, R. A model generation theorem prover in KL1 using a ramified-stack algorithm. In Proc. of the Eighth International Conference on Logic Programming, Paris, 1991.

Current News

*** Toshiba develops voice entry software**

Toshiba has developed voice entry software to operate computers using voice commands such as "delete" and "end". With this software, the user can perform jobs such as retrieving documents and transferring graphs simply by speaking into a microphone. The software also makes it possible to rapidly carry out a variety of operations by means of multimedia entry, through combination with existing entry devices such as keyboard and mouse. This general-purpose software can be used with many types of applications such as desktop publishing and communications software. No special entry devices are needed other than the microphone.

The software runs on a workstation platform. With desktop publishing software, the user can use voice commands such as "copy", "reverse top-bottom", and "reverse left-right" to move around an illustration to be inserted into a document. With E-mail, the user can use voice commands such as "today", "yesterday", "first", and "last" to start reading letters from any desired position.

Another feature of the software is the ability to perform simultaneous, parallel processing of two jobs. For example, while creating a document using the keyboard, the user can simultaneously retrieve documents using voice instructions. Toshiba says the software can recognize a maximum of 400 words, using any voice, and that it has a recognition rate with an accuracy of 99%.

*** Computer virus damage multiplies 4.4 times**

A total of 253 cases of virus damage were reported in 1992, according to the computer virus damage status report for Japan as a whole released by the IPA (Information-technology Promotion Agency, Japan). This was a sharp growth of 4.4 times over the previous year. In 1992, the virus that caused the most damage was the "Yankee Doodle" virus, with 118 reported cases. (This virus plays the tune "Yankee Doodle Dandy" when it infects a computer.) There were 43 cases of the "Cascade" virus, which makes characters fall off the screen a specified period of time after infection. In addition to these viruses, which entered Japan from overseas via PC communications, etc., new strains of domestically produced viruses were also reported, such as the program-destroying virus that displays "Be My Valentine" on February 14th.

The system for reporting computer virus damage was instituted in April, 1990. Since then, a total of 347 cases had been reported as of the end of January, 1993. The number of damage reports is thought to have increased so markedly in 1992 due to rising concern about the more frequent appearance of new, vicious viruses, especially among personal computer users such as educational and research organizations. More widespread knowledge of the reporting system also contributed to the increase in reports.

*** Sony, Matsushita, and 5 other Japanese, U.S., and European companies to invest in "General Magic."**

In February, 1993, General Magic, a multimedia technology research company associated with Apple Computer, announced that seven companies would enter into capital participation in the company, including Japan's Sony and Matsushita Electric Industrial, America's Apple Computer, Motorola, and AT&T, and Europe's Philips. Through this U.S.-Japan-Europe business partnership, the company will engage in joint development of multimedia software and equipment to unify video, voice and character information using computer and communications technology. The "Telescript" transmission processing standard under development at General Magic appears likely to become a worldwide standard, leading the various enterprises to plunge headfirst into this capital participation arrangement.

U.S. and Japanese companies have been competing to take the lead in standardization in the multimedia field, but General Magic now appears to be in the forefront in this area. This arrangement ties together principal U.S., Japanese, and European companies in the audio, household electric, communications, computer, and semiconductor fields.

*** NTT and KDD to jointly research common software for next-generation communications with worldwide carriers**

NTT and KDD are initiating research into software to be used as a common specification in providing next-generation communications services through cooperation with telecommunications carriers in various countries, such as America's AT&T. A research consortium known as "TINA" will be established at an early date with participation expected from carriers in a variety of countries, including NTT, KDD, AT&T, British Telecom, France Telecom, and Korea Telecom, as well as from Bellcore and from computer and communications equipment makers such as IBM. Technical experts from the participating companies will be dispatched to Bellcore research facilities, where they will engage in joint development of software.

Fusion of computer and communications functions is expected to progress in next-generation communications services. It is expected that new communications services offering a wide range of information processing functions will appear (such as translation services and automatic retrieval systems that operate by voice command). However, most of the world's telecommunications carriers have developed services independently, and standardization work has fallen behind. The result is that late-entry carriers have been confronted with mounting costs when opening new services, since to connect lines they must revise software, etc. to match the systems of other-party carriers in other countries. Under this new agreement, common specifications for a wide range of software will be developed to respond to the globalization of communications. This will enable communications lines to be linked and services to be mutually extended even if carriers in various countries use different computers and switching systems.

*** Government to make legal information available to public at charge**

The Japanese government will soon open its legislative and governmental ordinance databases to private concerns on a fee basis. Previously, these databases could only be accessed by central government offices. A total of 4,007 laws and ordinances will be included in this database service, including the Constitution. The Institute of Administrative Information Systems, which is under the jurisdiction of the Management and Coordination Agency, will transfer magnetic data to private database companies. Based on that data, the companies will develop software that can meet the demands of general users. Users will receive information from the companies via PC communications. They will be able to review all related laws and ordinances that include specific words simply by entering the relevant words by keyboard.

Because up until now only central government offices have been able to utilize these databases for reference to laws and ordinances, companies and local governments have had no choice but to make frequent visits to numerous related government offices when they meed to perform legal reviews. (For example, when private enterprises enter into new undertakings, they must investigate the governmental provisions in related laws and ordinances, and when local governments establish regulations, they must reference the entire compendium of laws in Japan to ensure that the regulations do not conflict with any laws or ordinances). For this reason, the Management and Coordination Agency has been besieged with strong demands for the opening of this data to the public. The decision to implement this new service was made as part of an administrative reform policy, and the service is scheduled to begin in July, 1993. In addition to corporate enterprises and local governments, it is expected that the service will also be utilized by persons involved in the judicial system, such as lawyers and judicial scriveners, and specialists from the patent departments of manufacturers. Fees will be collected from users, but since these will include software production costs and hardware operating expenses, the usage fees have not yet been determined.

Please send the ORDER FORM directly to:

Promotion Division
JIPDEC
3-5-8 Shibakoen, Minato-ku
Tokyo 105 JAPAN

TEL : +81-3-3432-9384
FAX : +81-3-3432-9389

ORDER FORM

○ **Please send me JAPAN COMPUTER QUARTERLY as checked below:**

☐ Annual Subscription — ¥ 13, 000 (including air mail charge)

☐ Back Copies — ¥ 3, 500 per copy (including air mail charge)

No. ____________

____

____

____

Total: ¥ ____________

○ **I will make a payment as follows:**

☐ Bank funds transfer
Bank Account: Mitsubishi Bank, Toranomon Branch
Account Number: Futsu Yokin 0000739
Account Holder : (Zaidan Hojin)
Japan Information Processing
Development Center (JIPDEC)

☐ Check enclosed

Signature: ____________ Date: ____________

Name: ____________

Position: ____________

Company: ____________

Address: ____________

Tel: ____________ Fax: ____________

JIPDEC

Japan Information Processing Development Center