PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1984,
edited by ICOT.

SEQUENTIAL INFERENCE MACHINE: SIM PROGRESS REPORT

Shunichi Uchida and Toshio Yokoi
ICOT Research Center
Institute for New Generation Computer Technology
Tokyo, Japan

ABSTRACT

The Fifth Generation Computer Systems (FGCS) project adopts a logic programming language as its kernel language. It specifies the general framework of both software and hardware systems of target computer systems. Thus, a new tool, that is, a new computer system is needed to provide researchers with an efficient programming environment for logic programming.

This new computer system is called a sequential inference machine (SIM) because it has a high level machine language, called Kernel Language version 0 (KL0). It is based on logic programming and executes KL0 sequentially.

A subproject is made to develop a SIM system and is called the SIM project. This project includes the development of both hardware and software systems, such as a personal sequential inference machine, PSI; a local area network system, ICOT-Net; and a programming and operating system, SIMPOS.

This paper describes the progress of the SIM project, mainly, the development of its architecture and hardware system. However, its software system, including SIMPOS and its system description language, ESP, are closely related to the architecture. Therefore, they often require and will require the modification and extension of the KL0 function, as well as of the architecture and hardware system.

This paper briefly describes this feedback from the software system to the KL0 functions and hardware system. It also includes a R&D plan for the intermediate stage.

1 INTRODUCTION

1.1 Motivation

The FGCS project adopted logic programming as its basic language because the language is considered to be the interface between the software system and hardware system of the project's target computer systems. It was also chosen as it is considered the most appropriate base from which to implement a sophisticated software system for a knowledge information processing system and a powerful hardware system using highly parallel architectures and VLSI technology.

To carry out the research and development, certain tools were considered very important. In particular, an efficient programming environment is essential. Thus, a subproject to develop a new computer system which supports logic programming was planned.

This computer system was named a sequential inference machine, SIM, and the subproject is called the SIM project. The SIM project was classified as one of the most

important subprojects in the initial stage of the ten-year FGCS project. Its goal is to develop hardware and software systems which can provide researchers with an efficient programming environment and also make a lot of software experiments possible.

1.2 Requirements

The SIM system was and is required to have sufficient processing power and memory space for the hardware system, as well as efficient and easy-to-use programming facilities for the software system.

For processing power and memory space, it was estimated that the hardware system should attain at least the same level of processing power as DEC-10 Prolog, the world's fastest software processor running on the DEC 2060. A faster hardware processor was also hoped for by some software researchers to cope with larger scale software experiments.

These basic requirements were considered to be best realized by a super personal computer equipped with sophisticated man-machine interface devices. Its software system should have many new features to make programming more productive and reliable. These features are realized by such software subsystems as a multi-window system with a mouse, an efficient editor, an easy-to-use debugger, a librarian for software modules and a coordinator (or session manager).

A personal computer should also have a local area network system to exchange messages, programs and so forth. Its software system should, thus, include a network subsystem to enable users to access remote files and processes of other computers.

To implement these sophisticated software systems summarized as a programming and operating system, a powerful system description language is required. Programs written in this language are compiled in the machine language and executed.

All of these requirements for both the software and hardware systems were investigated and an outline of the R & D plan was made before the FGCS project actually started in June 1982.

1.3 R&D Schedule

As the SIM system should be used as a project's standard tool and be available in the intermediate stage, R & D of the both software and hardware system should be completed in the first three years of the project, namely, the initial stage. This means that the hardware system had to be built as soon as possible, namely, in a year or less, so that the final phase of software development could be done on the actual hardware system.

After the FGCS project started and ICOT began its R & D activities, a detailed plan of the SIM project was made. In this plan, we decided to build two types of hardware systems, which were named "basic hardware system" and "extended hardware system". The basic hardware system had to be built within a year so that it could be used for software development. The extended hardware system was allowed a little more time, two years. However, it had to attain faster speed than the basic one.

1.4 Organization of SIM System

From the project management viewpoint, the SIM project can be divided into such components as the following:

(1) SIM hardware system

- Basic hardware system

This system consists of a personal sequential inference machine, PSI, and a local area network system, ICOT-Net. This also includes PSI's firmware development

and also its development support system.

- Extended hardware system

The main part of this system is a high-speed processor system which is designed as to be the backend hardware processor of the PSI. Some specialized I/O equipment such as a picture input and output device was also included in this system.

(2) SIM software system

SIM's total software system is named SIMPOS, i.e. SIM Programming and Operating System.

- Kernel operating system

The kernel operating system provides functions for managing processor, memory and I/O devices; for formulating abstract resources such as process, pool, stream; and for providing sophisticated I/O functions such as a multi-window system, file system and network system.

- Programming system

The programming system has such functions for programming as language processors for the system description language, ESP, and other modules as an editor, an debugger, and an coordinator.

2 PROGRESS OF R & D

2.1 Machine Language Design

The design of the SIM system began with the design of Kernel Language, version 0 (KL0). The basic design philosophy determined that KL0 should be based on logic programming. However, existing logic programming languages such as DEC-10 Prolog of Edinburgh (Bowen, et al. 1983) and Prolog-II of Marseille (Van Caneghem 1982) were not designed as for machine language or system description languages.

Therefore, we had to reconsider the control structure, the data type and low level built-in predicates so that we could write the kernel operating system and the expected application programs.

For extension of the control mechanism, we did not think that the simple cut operation in DEC-10 Prolog was sufficient to implement the interpreter, debugger, error handler and so on. We also added a multi-level cut operation. Furthermore, we added a bind-hook operation and an exception-hook operation like the freeze operation of Prolog-II.

For extension of the data type, we discussed the necessity of such data types as strings and vectors. We added these data types and also built-in predicates to handle these data types. These data types, especially vectors, are often used to implement a variety of operating system control tables.

For the low-level built-in predicates, we had to add many special built-in predicates to access and control hardware resources such as various hardware registers to control and test the CPU, memory and I/O devices. These special predicates are mainly used in the kernel operating system, namely, the memory manager and the device handlers. Adding to these predicates, we included the usual arithmetic and logical operations, e.g. integer and floating-point operations.

Another discussion on KL0 design was about the level of the KL0. The level of the KL0 could have been designed to be as high as the usual prolog level or as low as the intermediate code of DEC-10 Prolog, in which source programs are compiled in the intermediate codes which are interpreted by a virtual software processor called the PLM. (Warren 1977)

We have designed the level of the KL0 to be as high as the source program of DEC-10 Prolog because we expected to have more room to introduce firmware and hardware mechanisms to enhance the processing speed

of the KL0.

We started the design of the KL0 in July 1982 and the specifications of the KL0 (first draft) were finalized by a team of three ICOT researchers around the end of November that year.

2.2 Design of Main Components

The function and performance goals of the main components were designed depending on various factors. Some examples are the time span to complete the development, and the technical difficulty of attaining these goals, the usable man-power, the anticipated technical skills of the researchers and engineers, and so forth.

To provide the hardware system to software people who had to develop the SIM's operating system and programming system, a part of the SIM total system was separated as the basic hardware system which had to be developed as soon as possible. Then we divided the SIM hardware system into two parts, namely, the basic hardware system and the extended hardware system.

2.2.1 Basic Hardware System

The basic hardware system was planned so that the development would be completed before the end of 1983. This system includes the PSI and ICOT-Net.

Thus, a team for designing the PSI, consisting of six ICOT researchers, was formed around the end of August 1982. This team's aim was to design both PSI's firmware and hardware systems. It started its activities with the purpose of making the PSI a super personal computer equipped with innovative I/O equipment like a bit-mapped display and LAN system. The team also planned for the PSI to attain the same level of performance as DEC-10 Prolog on the DEC 2060, however, the available memory space had to be much larger than that.

First, the team tried to draw a rough sketch of the PSI's architecture and hardware organization, the KL0 interpretation mechanism being implemented with firmware, the development support system (including a minicomputer to be used as a workbench), and so forth. The architecture design began with the design of an internal representation of the KL0, namely, the data format in the memory and also the interpretation mechanism.

We had a discussion and found out that the mechanisms for multi-process support combined with I/O control and memory management related to garbage collection were also interesting, but very difficult, research problems. We decided to provide some hardware support for shortening process switching. However, we did not attach a front-end processor like some Lisp machines. This was because software researchers want to keep the chance to write very low-level control programs like device handlers in the KL0 so that they can evaluate the descriptive power of the KL0.

For the memory management mechanism, we did not include the virtual memory system. This was because we could use brand new 256K bit memory chips and the 40-bit 16M-word main memory would occupy less than one third of the single cabinet. We estimated that the 16M-word would be sufficient for most application programs. This made it possible to use the simple slide compaction type garbage collection method.

Along with the design of the PSI, the design of the ICOT-Net was done by another team. ICOT-Net is an CSMA/CD type network like the Xerox Ether-net. It uses a microcomputer controlled adapter called LIA (LAN Interface Adapter) for each node computer. The LIA handles low-level protocols so that the computer, e.g. PSI, can be freed from such low-level processing

as retransmission of missing data.

Work on this design phase was just like doing a jigsaw puzzle. This is because each member of the team tried to design some parts of the PSI while observing its total image and adjusting the boundaries of each part. The functional specifications of the PSI (first draft) were made in December 1982. The final specification of these are described in the next section.

After this, we entered the detail design phase of the PSI. In this phase, we started the design of the micro-interpreter, namely, the microprogram to interpret the KL0, and the micro-instruction format. Along with this, detailed design of the main hardware modules like the data path and the sequencer of the CPU, memory controller, etc was done. The specifications of the micro-instructions (first draft) were made in April 1983. From the middle of this stage, manufacturers were asked to join the SIM project. The final phase of this design was carried out by a joint team of ICOT members and manufacturers.

The design of PSI firmware, namely, the KL0 micro-interpreter started along with the hardware design. The first design step was to determine the internal representation of the KL0, namely, the machine instruction format of the KL0 and its interpretation mechanism. This mechanism could execute KL0 programs as fast as possible and efficiently implement several augmented control predicates such as bind-hook and multi-level cut operations.

We made many sample programs to verify algorithms for such basic operations as unification, execution control and stack manipulation. In this design process, we often referred to the implementation of DEC-10 Prolog. We finally decided to adopt the structure sharing algorithm and use four stacks. As we extended the data types, methods to handle the heap data area were a little different from those of DEC-10 Prolog.

The other firmware modules we had to design were mechanisms to support multi-process control, memory management to implement multiple virtual stacks, I/O control including interrupt handling, and so forth. Some effort was required to harmonize these low level mechanisms with the interpretation mechanism of the KL0.

The design of these mechanisms was deeply related to the hardware design. Furthermore, the design of the KL0 proceeded in parallel with these design effort. Detailed design of the PSI firmware system was completed around the end of January 1983. After this, the design of firmware development started.

2.2.2 Extended Hardware System

The extended hardware system was aimed at enhancing the SIM system's capability to cope with larger software experiments requiring more processing power. Another goal was to enhance input and output capability to handle drawings, pictures and so forth.

In the conceptual design phase of the SIM system, we investigated future needs for processing power and memory capacity. We concluded that some application systems would require much more processing power than the DEC-10 Prolog one. Thus, we made plans to develop a faster hardware processor.

Here we aimed at the development of the fastest possible processor having the development a more research oriented than that of the PSI, that is, one permitting more risk. However, we hoped that this processor would be applicable to some larger scale software experiments than PSI could not cope with.

In the conceptual design phase, we had two types of machine organizations. One was an upward-compatible machine to the PSI and the other was the backend high-speed processor of PSI. After a series of discussions including members of a manufacturer we determined that this machine was the backend processor of the PSI, mainly because this type of processor could eliminate quick process switching and input and output control. Thus, the design could be concentrated on the speed-up of the processor.

We tried to introduce a different approach from the PSI, namely, a different interpretation method based on structure copying, a different machine instruction set, different architecture, and different device technology. Important differences between the high-speed processor and the PSI were that the interpretation mechanism was based on structure copying and the level of machine language was lower than that of the PSI so that its compiler could make greater optimization for the speed-up of program execution.

In spite of these differences, we developed the language so that it could support a subset of the KL0, including important built-in predicates. Hardware oriented built-in predicates were, of course, different between the high-speed processor and the PSI. Thus, most user programs written in ESP could be run on this processor by using its compiler.

For the hardware design, we decided to use CML (Current Mode Logic) circuits for the speed-up. In addition, we tried to introduce pipe-lined local parallelism in the CPU of the processor.

These had produced several research problems in functional design and made production more difficult. Thus we had to spend several months on functional design because we needed a detailed design of the important parts of the interpretation mechanisms and hardware modules to verify the feasibility of the functional specifications.

The functional specifications of the hardware system (first draft) were made around the end of November 1983. After that, a detailed design followed and it was almost completed in May 1984. In these design phases, most detailed work was done by the manufacturer.

2.2.3 Software System

Development of SIM's software system began with conceptual design of the SIM programming and operating system, SIMPOS. Development was carried out so as to achieve a personal operating system having sophisticated functions for smoothly performing man-machine interactions as well as to facilitate constructions of a distributed system by connecting many PSI's via a LAN system (ICOT-Net) in addition to ordinary functions of recent personal computer systems. Furthermore, an efficient programming system had to be built using these functions including an editor, a language processor, a debugger and many utility software modules.

Though the functional goal of SIMPOS was as high as the innovative operating systems of super personal computers like LISP machines, there was another risky condition, that is, it should be written in a completely new language which had not been precisely designed. And also it was anticipated that most members of SIMPOS development team would not be familiar with this language.

Until the end of 1982, most effort made was in the design of functional specifications of important software modules and subsystems such as a window system, file system and editor, and for designing the module structure of the kernel operating

system based on the object oriented concept. Adding to these design efforts, primitive functions to control hardware systems and implement the kernel operating system were selected and added as builtin predicates of KL0.

The functional specification of SIMPOS were summarized around the end of March, 1983. The design work of this stage was done by a team of less than ten ICOT members.

Almost in parallel with this work, the design of the system description language, ESP, had begun considering the specifications of KL0 and also requirements imposed by SIMPOS's design philosophy. One of the interesting points of discussion was how to introduce modularization functions in ESP so that SIMPOS could be smoothly programmed and debugged. After a series of discussions, a class system with a multiple inheritance mechanism was adopted as the ESP's modularization mechanism. The ESP specification (first draft) were made in July, 1983.

From April, 1983, all the ICOT members related to the SIM project were gathered in one section of the ICOT research center so that the activities from software to hardware could be more smoothly coordinated.

From June, 1983, the SIMPOS development team was expanded to be about 30 people including 7 ICOT members and detailed design was started. New members coming from manufacturers needed to learn many new things, namely, logic programming, object-oriented programming, KL0, ESP and so forth. They were somewhat disoriented at first but were able to understand the material after a few months.

They were divided into several small groups for the purpose of starting the detailed design of all the subsystems of SIMPOS. As the first versions of the ESP compiler and simulator were made on the DEC 2060 as a cross system in August and they were available to actually write small sample programs in ESP. The work in the detailed design was to actually define class modules and determine the interface specifications required to implement each subsystems. The detail specifications of SIMPOS (first draft) were summarized around the end of November, 1983.

2.3 Production of SIM

2.3.1 Production of PSI and LAN

The production of the PSI hardware system was partially started around May, 1983 along with the detailed design of such parts as memory system and I/O bus interface. To enhance maintainability, RAS functions were added to make trouble shooting easier. The detailed specifications of the hardware system were completed around the end of June, 1983 and actual production started. After this, minor modifications and improvements were repeated and finally a first PSI hardware system including I/O devices was brought to ICOT just before Christmas, 1983.

As one of the development support tools for PSI, a minicomputer, PDP11/23 was introduced as a development support system and was connected to the DEC 2060 via DECnet. This minicomputer was named SVP (PSI's supervisor processor) and was intended to be used for hardware and firmware debugging. It was later used for debugging the kernel parts of SIMPOS, for example, an IPL (initial program loading) program.

2.3.2 Firmware Development of PSI

The first very important task in the firmware development was to develop an easy-to-use micro assembler and a powerful micro simulator to write and debug micro-

programs. It was requested that the micro assembler could permit high-level language-like expression and detect coding errors as precisely as possible.

It was determined that a general purpose micro assembler should be written in Prolog. General purpose micro assemblers usually lack precise error detection mechanisms because of their inability to defining the details of constraints unique to a specific micro architecture. However, for our assembler, Prolog interpreter could be regarded as the kernel part of the assembler processor. Thus the architecture definition could be written just like a Prolog program. Then the constraints could be specified very precisely. Thus, the result is an assembly language of high readability with precise error detection capability.

For a micro simulator, the main requirements were a powerful debugging ability and also an easy-to-use human interface. The basic philosophy of the design was that the simulator's debugging environment should be more efficient and comfortable than that of the real machine with the SVP. Our simulator was written in PASCAL using about 6000 lines. Its first version became available on the DEC 2060 at the end of June, 1983.

The detailed design and production of firmware for the PSI, namely, a micro-interpreter of KL0, were carried out in parallel.

As KL0 included more than 100 built-in predicates, some simple built-in predicates were quickly designed and programmed, however, the kernel parts of the micro-interpreter, such as a basic control module and unification module, required a longer design time. The specifications of the PSI firmware (first draft) were completed at the end of August 1983

The coding and debugging of micro-programs were almost done on the DEC 2060 using the micro assembler and simulator which were improved several times and had become very comfortable and reliable tools. The first small subset of the microprograms had been debugged on the cross system, namely the DEC 2060, at the end of November, 1983 and given to the hardware people of the manufacturer for testing the hardware system of PSI. Most part of the microprograms, except for some microprograms to control process switching, memory control, garbage collection and so on, had been debugged on the cross system before the end of February, 1984.

Along with the firmware development, a PSI hardware monitoring program on the SVP, which is an firmware debugging aids on SVP, had been made around the end of January, 1984. Thus, the firmware debugging using the real machine started in February and first subset of microprograms for SIMPOS debugging was completed at the end of the month. This enabled the operating system people to start debugging of the IPL program.

Almost all the microprograms required for SIMPOS development had been completed around the end of March, 1984. The total number of microprogram steps was about 12K. The firmware development team consisted of 12 people at its peak and included 3 ICOT members. However, some of them were converted from the hardware design team.

2.3.3 Production of extended hardware system

The production of the high-speed processor was partially started before May, 1984 and currently is still in progress. The firmware development of this processor was started in March, 1984. The hardware production is scheduled to be completed by the end of 1984.

programs. It was requested that the micro assembler could permit high-level language-like expression and detect coding errors as precisely as possible.

It was determined that a general purpose micro assembler should be written in Prolog. General purpose micro assemblers usually lack precise error detection mechanisms because of their inability to defining the details of constraints unique to a specific micro architecture. However, for our assembler, Prolog interpreter could be regarded as the kernel part of the assembler processor. Thus the architecture definition could be written just like a Prolog program. Then the constraints could be specified very precisely. Thus, the result is an assembly language of high readability with precise error detection capability.

For a micro simulator, the main requirements were a powerful debugging ability and also an easy-to-use human interface. The basic philosophy of the design was that the simulator's debugging environment should be more efficient and comfortable than that of the real machine with the SVP. Our simulator was written in PASCAL using about 6000 lines. Its first version became available on the DEC 2060 at the end of June, 1983.

The detailed design and production of firmware for the PSI, namely, a micro-interpreter of KL0, were carried out in parallel.

As KL0 included more than 100 built-in predicates, some simple built-in predicates were quickly designed and programmed, however, the kernel parts of the micro-interpreter, such as a basic control module and unification module, required a longer design time. The specifications of the PSI firmware (first draft) were completed at the end of August 1983

The coding and debugging of micro-programs were almost done on the DEC 2060 using the micro assembler and simulator which were improved several times and had become very comfortable and reliable tools. The first small subset of the microprograms had been debugged on the cross system, namely the DEC 2060, at the end of November, 1983 and given to the hardware people of the manufacturer for testing the hardware system of PSI. Most part of the microprograms, except for some microprograms to control process switching, memory control, garbage collection and so on, had been debugged on the cross system before the end of February, 1984.

Along with the firmware development, a PSI hardware monitoring program on the SVP, which is an firmware debugging aids on SVP, had been made around the end of January, 1984. Thus, the firmware debugging using the real machine started in February and first subset of microprograms for SIMPOS debugging was completed at the end of the month. This enabled the operating system people to start debugging of the IPL program.

Almost all the microprograms required for SIMPOS development had been completed around the end of March, 1984. The total number of microprogram steps was about 12K. The firmware development team consisted of 12 people at its peak and included 3 ICOT members. However, some of them were converted from the hardware design team.

2.3.3 Production of extended hardware system

The production of the high-speed processor was partially started before May, 1984 and currently is still in progress. The firmware development of this processor was started in March, 1984. The hardware production is scheduled to be completed by the end of 1984.

approaching 90K ESP lines.

3 MAIN COMPONENTS OF SIM

3.1 Basic Hardware System

As the details of PSI's firmware and hardware systems are described in other papers, the main features of its architecture and hardware system are summarized. (Taki, et al. 1984) (Yokota, et al. 1984)

3.1.1 PSI Architecture

(1) Tag architecture: All memory cells are 40 bits consisting of an 8 bit tag and 32 bit data fields.

(2) Microprogrammed control: Compiled internal KL0 codes are to be executed by a microprogrammed interpreter.

(3) Hardware stack mechanism: Main memory is divided into 256 logically independent areas, each of which can be used as stack or heap area and a demand-based page assignment mechanism is provided.

(4) Multiple-process support: Up to 63 hardware-supported processes are available.

(5) I/O bus: A standard IEEE 796 bus is used.

(6) LAN: An CSMA/CD type similar to the Ether-net. Transmission speed is 10 Mbps.

3.1.2 PSI Hardware

(1) Execution speed: about 30K LIPS (Logical Inference Per Second)

(2) Microprogram store: 64 bits x 16K words

(3) Machine cycle time: 200 nano second

(4) Memory capacity: 40 bits x 16M words, using 256K bit chips

(5) Cache memory: 40 bits x4K words x 2

(6) Device technology: Mainly TTL for CPU and NMOS for main memory

3.2 Extended Hardware System

The extended hardware system is still under development. The main features are described.

3.2.1 High-speed Processor Module

(1) Machine language: Based on a stack oriented instruction set, the level of which is lower than the one of PSI. (Tick, et al. 1984) It will include more than 170 built-in predicates. It is designed to be able to support KL0 and ESP.

(2) Interpretation mechanism: Based on structure copying and uses three stacks.

(3) Tag architecture: All memory cells are 36 bits consisting of an 4 bit tag and 32 bit data fields or an 7 bit tag and 29 bit data fields.

(4) Microprogram control: The machine language is interpreted by a microprogrammed interpreter.

(5) Memory management: Main memory is divided into 8 logically independent areas, each of which can be used as stack or heap area, and a demand-base page assignment mechanism will be provided.

(6) I/O: Designed to be connected to PSI via its supervisor processor.

(7) Execution speed: about 200K LIPS

(8) Microprogram store: 80 bits x 11K words

(9) Machine cycle time: 100 nano sec.

(10) Memory Capacity: 36 bits x 64M words

(11) Cache memory: 36 bits x 8K words

(12) Device technology: Mainly CML for CPU and NMOS for main memory.

The design of this processor still includes many research items to be studied. Thus, some of these features may possibly be modified because of the experimental na-

ture of this processor, however, it is expected that it will attain more than 200K LIPS.

3.2.2 Specialized I/O equipment

To provide a facility for handling pictures a specialized work station for picture input and output is being developed. This work station includes a high resolution color display and a color printer for picture output and also a TV camera and a FAX for picture input. It is controlled by micro processors and supports graphic functions using a specialized chips. The work station will be connected to PSI's I/O bus and can be used for experiments using pictures.

3.3 Software System

The details of SIMPOS and its system description language, ESP, are described in other papers. (Chikayama, 1984) (Yokoi, et al. 1984) SIMPOS is still under development. Thus, a profile of SIMPOS is given.

(1) A personal operating system with multi-process support.

(2) Its module structure is based on the object oriented concept.

(3) Man-machine communications are made through the window subsystem which includes input and output of Japanese character. Thus, in the SIMPOS world, all characters are expressed in 16 bits.

(4) The network subsystem enables a user to access remote resources such as remote processes and files.

(5) Its programming system includes a library module including an ESP compiler and linker, an interpreter and debugger, a structured editor, and so forth.

All the SIMPOS software modules are written in ESP. ESP will become a standard language not only for system programmers but also for SIMPOS users.

4 FUTURE PLAN

4.1 Improvement of SIM

The most important goal of the SIM project is to produce a usable tool providing researchers in the FGCS project with a logic programming environment.

Although some parts of the SIM project are still in progress, this goal will first be attained by the basic hardware system, namely, PSI and ICOT-Net , and the software system SIMPOS. These systems will be released to the researchers in the intermediate stage. However, even these systems will require modifications and extensions, e.g, extensions of utility software packages, addition of I/O devices.

The extended hardware system will also needs many modifications and extensions so that it can be stable and reliable as much as the basic hardware system. Thus, the efforts to improve SIM system will continue in the intermediate stage.

4.2 Parallel Software Development Support

As the most important research problem in the intermediate stage is the development of parallel processing systems. Currently, a parallel logic programming language called KL1 is being designed in ICOT. To implement an experimental language processor and simulator for the KL1, a parallel hardware system which is stable and easy-to-use will be needed.

Therefore, a small scale multi-processor system is planned to be developed using the PSI. This system will be consisted of several tightly coupled PSIs, each of which supports the KL1 by extending its firmware. As the PSI is a sequential machine, a part of KL1 programs will be sequentially executed, however, some other parts can be executed in parallel. It is expected that this system will be effective to study a method

to divide a program into parallel processable modules, and to estimate communication cost between parallel processes, and so forth.

5 CONCLUSION

The SIM project is still in progress, however, its goal is now in sight. The basic hardware system, namely PSI and ICOT-Net, and the software system, SIMPOS, will be available as a usable tool, although they will need various improvements.

This project has already produced many fruitful results. One of the results is the one related to the KL0. Its base has been changed from DEC-10 Prolog to ESP. Now, ESP is the standard language of the FGCS project. Thus, the specifications of the KL0 are extended to support both logic programming and object-oriented programming concepts.

Next result is that this project has answered to such a question like "Is it possible to write an operating system in Prolog ?". Although the object-oriented concept is added to the logic programming concept, ESP is proved to be very effective in writing the operating system.

Furthermore, for a performance problem, this project has proved that ESP like languages will provide architectual researchers with more room to introduce new mechanisms to speed the execution. This research effort is directly related to improvement of software productivity and reliability.

These research results will surely encourage future R&D of the FGCS project.

ACKNOWLEDGMENTS

The SIM project is carried out in very tight cooperation between ICOT and five manufacturers. The authors express their gratitude to all the researchers and engineers who join in this project. Furthermore, they express their thanks to Dr. D.H.D. Warren who established the base of Prolog implementation and gave us many advices. They also thank to many Japanese and foreign researchers for their fruitful advices and comments.

REFERENCES

Taki, K., et al., Hardware design and implementation of the personal sequential inference machine (PSI), Proc. of FGCS '84, Tokyo, Nov. 1984

Yokota, M., et al., A microprogrammed interpreter for the personal sequential inference machine , Proc. of FGCS '84, Tokyo, Nov. 1984

Chikayama, T., Unique features of ESP, Proc. of FGCS '84, Tokyo, Nov. 1984

Yokoi, T., et al., Sequential inference machine: SIM Its programming and operating system, Proc. of FGCS '84, Tokyo, Nov. 1984

Bowen, D. L., et al., DECsystem-10 PROLOG User's Manual, Department of Artificial Intelligence, Univ. of Edinburgh, 1983

Warren, D. H. D., Implementing Prolog-Compiling Predicate Logic Program, Vol. 1-2, D.A.I. Research Report, No.39-40, Department of Artificial Intelligence, Univ. of Edinburgh, 1977

Tick, E. and Warren, D. H. D., Towards a pipelined Prolog Processor, Proc. of Int. Sympo. on Logic Programming, Atlantic, Feb. 1984

Van Caneghem, M., Prolog II Manual D'Utilisation Groupe Intelligence Artificielle, Faculté des Science des Lumminy, Marseille, 1982