Inference Machine:

From Sequential to Parallel

by

Shunichi Uchida

Abstract

In this paper, the research and development plan for computer architecture in the fifth generation computer system project (FGCS Project) is described, focusing on the research on the inference machine.

Inference machines planned in this project can be classified into two types. One is a sequential inference machine and another is a parallel inference machine.

The sequential inference machine is to be developed to provide researchers with an efficient programming environment. It is designed as a personal computer which supports a logic programing language named KL0. This is an immediate development target and its design philosophy and machine characteristics are described.

The parallel inference machine which is one of the ultimate goals of this project is in a basic research stage and thus, an abstract discussion is made for introducing current research status and future research direction.

1. Introduction

In this paper, the research and development plan for computer architecture in the fifth generation computer system project (FGCS Project) is described, focusing on the research on the inference machine.

In FGCS project, a logic programming language has been chosen as its base language and it is named FGKL: Fifth Generation Kernel Language. The goal of this project is to develop basic computer technology to build an intelligent computer system and its prototype which will have an inference function, a knowledge base function and an intelligent interface function. [Moto-oka]

A rough sketch of the overall research plan for computer architecture in the FGCS project is shown in Fig. 1. [Uchida]

The inference machine is defined as a part of the prototype hardware system which support the inference function. It will actually be a parallel logic programming machine which consists of hundreds of such hardware elements as a processing element, a structured memory element and a network element. Thus, in this paper, inference mechanisms mean various software and hardware functions to support the logic programming language (FGKL).

To implement the project, a new institute named ICOT: Institute for New Generation Computer Technology was established in April 1982. And several research groups are now being organized in ICOT and also in eight manufacturers and some universities in Japan.

ICOT is the center of these research groups and it has a research center consisting of about 40 researchers. In this paper, the activity of ICOT's research center is mainly described. ICOT has now four research groups which relate to the inference machine, namely, a sequential inference machine group, a parallel inference machine group and a kernel language group.

Activities of these groups are closely related each other. In this paper, the author will try to summarize their current and future research activities from the author's personal viewpoint.

2. Research plan for the inference machine

The project spans 10 years, divided into three stages, namely, a 3-year initial stage, a 4-year intermediate stage and a 3-year final stage.

In the initial stage, technological components in programming languages and computational models, computer architectures, and hardware elements are chosen and studied to propose a couple of promising computational models and architectures which support the inference function.

In the intermediate stage, these computational models and architectures will be studied in more details and some of them will actually be implemented in hardware to evaluate the feasibility and the performance. Similar research will be done for the knowledge base mechanism as shown in Fig. 1. and experimental hardware systems will also be implemented.

In the final stage, research results of both the inference mechanism and the knowledge base mechanism are expected to be combined to build a prototype of the fifth generation computer system.

The research activity was started form June 1982 and it is now in the initial stage. The research items in the initial stage, which relate to the inference machine are as follows.

1) Fifth generation kernel language (FGKL or KL):

The base language of FGCS is a logic programming language called FGKL. A well-known example of this language is Prolog. The logic programing language was chosen because of its descriptive power for the applications in knowledge information processing and its clear semantics and computational model.

FGKL is defined as the machine language for inference machines. Now there are two versions, namely, KL0 and KL1. KL0 is the language for the sequential inference machine and KL1 is for the parallel inference machine.

2) Sequential inference machine (SIM and PSI):

SIM is a logic programming language machine which can directly execute KL0 by its firmware and hardware. The first experimental model of SIM is named PSI: Personal Sequential Inference machine. PSI is being developed to provide researchers with a efficient programming environment for software and hardware development, however, it will also become the first research step of the architecture and hardware mechanisms to support the inference mechanisms.

3) parallel inference machine (PIM):

This research item covers basic research subjects on the parallel inference machine (PIM) including languages, computational models, architectures, and hardware components. This has such sub-items as parallel inference mechanisms, dataflow mechanisms, and object oriented mechanisms. The details of these sub-items have not yet been determined, however, several research topics have been chosen and basic research on computational models and architectures has been begun.

These three items are closely related, however, they have differences in both theoretical and technological aspects.

SIM and KL0 may be closer to current technology and thier goals can be attained by extending current software and hardware technology.

Research on PIM and KL1 is still in the basic research stage and theoretical study is very much needed to attain these goals. Thus, in the architectural research, software and hardware simulation is immediate work, however, the experimental development of some hardware modules may also be necessary to evaluate the feasibility and performance of the proposed architectures.

For these differences, research on SIM and KL0 can be described in thier details, however, only an abstract discussion on the languages and computational models can be made for PIM.

3. SIM and KL0

3.1 Research aims and background

-To provide a powerful and efficient research tool to develop a larger scale experimental software systems such as a natural language understanding system and an expert system.

-To develop a hardware system as a personal machine which efficiently support KL0, using current architectural and hardware technology. The immediate goal is to develop PSI.

-To develop an efficient programming environment for Prolog (KL0) as being given by LISP on the personal machine. It must have such software and hardware systems as a screen editor, a system description language and its compiler, a file system and a local area network.

3.2 KL0

-KL0 is designed for PSI and thus it is considered as the machine language and executed sequentially.

-A couple of control primitives is added to usual prolog [Pereira], such as a bind-hook, an exception-hook and remote-cut operations. KL0 also includes many build-in predicates to efficiently implement various OS functions such as multiprocess control, interrupt handling, and input/output device control. [Chikayama]

-A compiler will be prepared to translate from KL0 to PSI's internal machine codes. It will have some optimizing functions. Thus, the internal codes of KL0 and a firmware interpreter of PSI is designed to make full use of these compiler's function.

-A system description language, ESP: Extended Self-contained Prolog will also be designed for systems programmers. It may correspond to a macro-assembler of conventional computers and it has macro predicates and special predicates to communicates with PSI operating system.

-A part of the concurrent prolog function is planned to be introduced to describe concurrent processes.

3.3 PSI architecture

-Sequential prolog machine with tag architecture and microprogram control.

-Efficient and compact format of KL0 internal code and its interpretation using four stacks and a structure sharing algorithm.

-Firmware implementation of KL0 interpreter and various hardware mechanisms to support quick execution of KL0 internal code. Examples of the mechanisms are a couple of fast register files with versatile pointers, a tag dispatcher for quick case branch and so forth.

-Addition of cache to improve memory access speed and address translation mechanism to implement multiple virtual stacks in its main memory. No virtual memory support.

-Firmware and hardware supports for process switching, interrupt handling and I/O device control.

-Local area network is also developed to connect between PSI's and other computers.

3.4 PSI hardware

-Medium performance (20 to 30 KLIPS) nearly equal to DEC-10 Prolog on DEC 2060 and a large capacity memory (40 bits X 4 to 16 MW).

-A data width is 40 bits (8 bit tag and 32 bit data part). Internal data path is 32 bit wide.

-Microprogram control is employed. A micro instruction is horizontal type and 64 bit wide. The micro cycle is less than 200 ns. The capacity of WCS is 16 KW. The data path and sequencer is implemented in TTL.

-Cache size is 45 bits x 4 KW x 2. The size of cache block is 4 words. Cache access time is one micro cycle.

-Logical address is 32 bits (8 bit area number, 14 bit page number and 10 bit displacement).

-Various devices can be attached to its I/O bus (Multibus) including a bit-map display, a pointing device, a printer, disks, and etc.

-Supervisor processor (LSI-11) will be attached to the first experimental model to measure various statistics of machine's and program's behavior.

4. PIM and KL1

4.1 How to approach PIM

There are many problems to be solved in the research on PIM. These problems may be classified into four areas, namely, language, computational model, architecture and hardware areas.

For PIM research, many similarities to the dataflow machine (DFM) research can be observed. In the architectural area, many research results on DFM will be useful for PIM, however, problems which have been left unsolved in DFM research will also be common to PIM.

In the following discussion, the author's personal opinions about the current research status and future trends are described.

4.2 Research on the parallel logic programing language: KL1

Arguments on the language features of the logic programming which relate to PIM may be summerized as follows.

-Level of abstraction is higher in logic programming languages than in other languages such as functional languages. The description in logic programming languages is more declarative than in functional languages. Then,it is expected that inherent parallelism involved in the problems are more naturally expressed in logic programming languages than the other languages. Non-deterministic part of the programs can be regarded as a candidate of parallel processable parts.

-Unification is a very powerful concept. It may be understood in many ways such as a pattern directed function invocation, an activation of processes, a message passing, a kind of type checking and a kind of an associative search operation. It can be extended to include the semantics of terms in equality checking. [Kahn].

-In unification process, direction of bindings for the arguments (input or output) is dnamially detrmied. These conditions mean that an architectural support for unification is an interesting but difficult research item.

-In the design of any parallel machine, the language is always very important because it gives the top level specification of the machine. The language must have the power for describing the behavior of parallel processes, namely, the creation and deletion of processes, syncronization and communication among processes.

-Examples of these languages which have this power are relational language [Clark 81] [Clark 83] and concurrent prolog [Shapiro]. These languages introduced such concepts as "stream" and "object oriented programming" in addition to language features of logic programming such as "unification". The model of these language (semantics) has many similarities to the dataflow languages like Id [Arvind] and Valid [Amamiya] as long as the process control including lazy evaluation is concerned.

-It is desirable to design the machine architecture which support the language of this level, however, in the current PIM research, this approach is difficult because the language research has not yet matured enough to start the architectural research.

In ICOT, a strong interest is being taken in concurrent prolog. Subset of concurrent prolog is now running on DEC 2060. Its interpreter was implemented in DEC-10 prolog by E.Shapiro. To write a larger programs in concurrent prolog, the power and memory space of the machine is insufficient. Thus, it is planned to develop more efficient interpreter of it by writing its interpreter in a conventional language. It is expected that this work will produce some ideas for considering the architecture which directly support this language.

4.3 Research on computational models

Another approach to the PIM architecture is to start from computational model of logic programming. Usually in this level, the execution of logic programming language is regarded as an and-or tree search. Thus, roughly speaking, parallel processing is applied for and-nodes or or-nodes or both of

them. In this level, how to deal with the non determinism (Backtracking) is an interesting point to study.

It may be a little easier to start from this level than to start from the language level described above because the problem to attack is more restrictive and seems to be more intuitive. Thus, in this approach, a core part of PIM is mainly considered. The image of the machine considered here may be more like a theorem proving engine than a high level language computer.

Several computational models have been discussed for logic programming. [Furukawa], [Conery]

-Or-parallelism can be considered in straight forward way. If there are several alternative or-nodes, they can be processed in parallel, however, it is not always apparent whether or not the parallel processing is meaningful because some or-nodes may be disjunctive. Thus, some control primitive is desirable to specify which or-nodes are meaningfully parallel processable to avoid vain computations.

-For and-parallelism, there may be two different ways to interpret and-nodes. One is to interpret and-nodes of a clause body as creation of parallel processes which are activated by binding of some variable to non variable terms like in concurrent prolog. In this case, deep backtracking is usually avoided.

Another way is more straightforward. It tries to process all the and-nodes in a clause body in parallel. However, in this case, there may be variables shared among some and-nodes. Thus, the order of processing these and-nodes is dependent on the direction of bindings for these variables. As long as no backtracking occurs, the order would be determined rather easily. If backtracking occurs, its control of redoing the dependent and-nodes has to be very complex. Thus, some control primitives are desirable to avoid the complexity.

-If all the input variables included in and-nodes are bound to grand terms and all the output variables can be also bound to grand terms after unification, an operation like "join" in relational database may be applied.

-Introduction of control primitives may be necessary for two reasons. One is for describing more efficient algorithms like a cut operation in sequential prolog. Another is for efficient resource allocation on a machine which has finite resources. Furthermore, in the resource allocation, dynamic assignment of priority for parallel process are required to use the resources such as processers and memories efficiently.

-In addition to the parallel processing of the and-or tree search, the parallel processing of unification is also studied. [Greene] This indicates another possibility of introducing parallel processing in the logic programming. It is interesting because it may be incorporate into both a sequential and parallel machine, however, the merit of this parallel processing is not always apparent because the mechanism to implement this seems to require much hardware.

The research of this level is currently the immediate work for PIM. A couple of conceptual machine models are now being designed in ICOT and some research groups in universities. And the development of software simulators are planned to verify the consistency of the models.

4.4 Research on architecture and hardware

A little research on PIM has been done in this level, however, it is expected that the research results on DFM's and reduction machines will be useful to design the architecture of PIM. Furthermore, generally speaking, it is assumed that PIM's are implemented as MIMD machine. Then, architectural and hardware technology which has been accumulated by the research on MIMD machines will also be useful.

It is apparent that more restrictions have to be imposed on the computational models if actual implementation in hardware is considered. Then, a certain computational model should be chosen to make step-by-step refinement of it, considering about the feasibility. On the selection of a computational model, such strategy as follows may be appropriate.

-To introduce only the and-parallelism may be an appropriate first step to study about the architectural support. In this case, the "stream" type processing like concurrent prolog may be appropriate because it can be considered as a moderate extension of sequential prolog.

-This processing can be implemented on a sequential machine as the language to control concurrency. In the design of KL0, introduction of a part of this processing is planned to write some OS functions. Extension of this processing from an uni-processor to a multi-processor may be a moderate first step toward PIM. And the implementation of this processing on a SIM can be regarded as a simulation of this processing.

This approach has many similarities to the dataflow model which can deal with lazy evaluation. Then, it is expected that the architecture of the DFM can suggest the architecture of PIM, although the support for unification is inherent in PIM.

-To introduce only or-parallelism may be more difficult than the previous one. The deep-backtracking can be avoided by the exhaustive search of all the or-nodes, however, too many processes would be easily invoked. Then, some strategy for restricting the invocation and for effective resource allocation may be essential. The architecture to support this has to be include the mechanism to implement this strategy.

-The next one may be the machine architecture which support both of the stream type and-parallelism and or-parallelism. This means that this architecture support the parallel language like concurrent prolog. Thus, this is the goal of these research steps.

-Another research step to introduce parallel processing in the architecture may implement a parallel unifier. It may be incorporate in a sequential machine to enhance the processing speed, however, it is not clear whether or not it can be implemented by reasonable amount of hardware.

After a consistent computational model and its overall machine architecture have been designed, functional specification of the machine components such as a processing element and a structured memory element should be determined. In this level of design, the simulation of the overall machine architecture and each machine components is essential.

If the conceptual machine consists of hundreds of these machine components, the scale of the simulation has to be large enough to exceed the power of an usual general purpose computer system. Thus, a hardware simulator for specific architectures will be desirable.

In some stage of the simulation, the detail design of certain hardware components may be necessary to make more precise estimation on the feasibility and the performance of the architecture.

The development of hardware systems for PIM may be required for these reasons as long as the initial stage is concerned.

In ICOT, a couple of software simulator and a few hardware simulator are expected to be developed in the initial stage. It is also expected that PSI can be used as a simulator for the stream type and-parallelism with a slight modification and addition of its firmware. If a part of the PSI cpu can be replaced with custom LSI chips, there will be a possibility to use it as a processing elements of the hardware simulator which includes several processing elements.

5. Conclusion

Recently, many researchers have started the research on logic programming and PIM. In this paper, the author has tried to summarize the research activities on the inference machines to indicate the rough sketch of future direction. The discussions and his opinions described so far are incomplete and controversial. Thus, they should be changed according to the up-to-date research results.

It is apparent that much more efforts have to be made to design a feasible machine architecture. Roughly speaking, the efforts may be classified into two groups.

One is to solve the problems which are specific to logic programming. Examples are the mechanism to support unification and the method to implement non determinism.

Another one is to solve the problems which may be common to most MIMD machines which intend to support high level languages like a DFM for a functional language. An example of these problems is a mapping from a computational model, in which infinite number of resources are assumed, to an actual machine architecture.

There is no simple way to solve these problems. Thus, various approaches on PIM must be tried in all the levels from language to hardware components including VLSI technology. Research on such machines as a DFM and a reduction machine is also important because they may give the architectural bases for PIM. To prepare for the hardware implementations and to make more precise simulation, research and development of various hardware components such as a

content addressable memory and a router are also important for PIM as well as other high level language machines.

Reference

[Moto-oka] Moto-oka, T. "Overview to the fifth generation computer system project", Proc. of 10th Int'l Sympo. on Computer Architecture, June 1983.

[Uchida] Uchida, S. "Toward a New Generation Computer Architecture", in VLSI Architecture as chap. 19, Prentice Hall (to appear), and also as Tech. Rep. of ICOT, TR-001, Aug. 1982.

[Chikayama] Chikayama, T., et al, "A Draft Proposal of Fifth Generation Kernel Language", Tech. Memo of ICOT, TM-007, Dec. 1982.

[Pereira] Pereira,L.M., Pereira,F.C.N. and Warren,D.H.D. "User's Guide to DEC system-10 PROLOG", Sept. 1978.

[Kahn] Kahn,K.M. "The implementation of Uniform, A Knowledge-Representation/Programming Language based upon Equivalence of Descriptions", Tech. Rep. of UPMAIL, Dec. 1981.

[Clark 81] Clark, K.L. and Gregory, S. "A relational language for parallel programming, Res. Rep. DOC 81/16, Dept. of Computing, Imperial College of Science and Technology, University of London, July 1981.

[Clark 83] Clark,K.L. and Gregory, S. "PARLOG: A parallel Logic Programing Language", Res. Rep. DOC 83/5, Dept. of Computing, Imperial College of Science and Technology, University of London, July 1981.

[Shapiro] Shapiro, E.Y. "A Subset of Concurrent Prolog and Its Interpreter", Tech. Rep. of ICOT, TR-003, Feb. 1983.

[Arvind] Arvind, et al, "An asynchronous programming language and computing machine", Tech. Rep. No.114A, UCI, 1976.

[Amamiya] Amamiya, M., et al, "A list-processing oriented dataflow machine architecture", AFIPS NCC, pp.143-151, 1982.

[Furukawa] Furukawa, K., et al, "Prolog Interpreter Based on Concurrent Programming", Proc. of the First Int'l Logic Programming Conf. pp. 38-44, Sept. 1982.

[Conery] Conery, J.S. and Kibler, D.F., "Parallel Interpretation of Logic Programs", Proc. of the Conf. on Functional Programming Languages and Computer Architecture, pp. 163-170, Oct. 1981.

[Greene] Greene, K.J., "A Concurrent unification Algorithm", Res. Rep. of Logic Programming Group, Syracuse Univ., Nov. 1982.

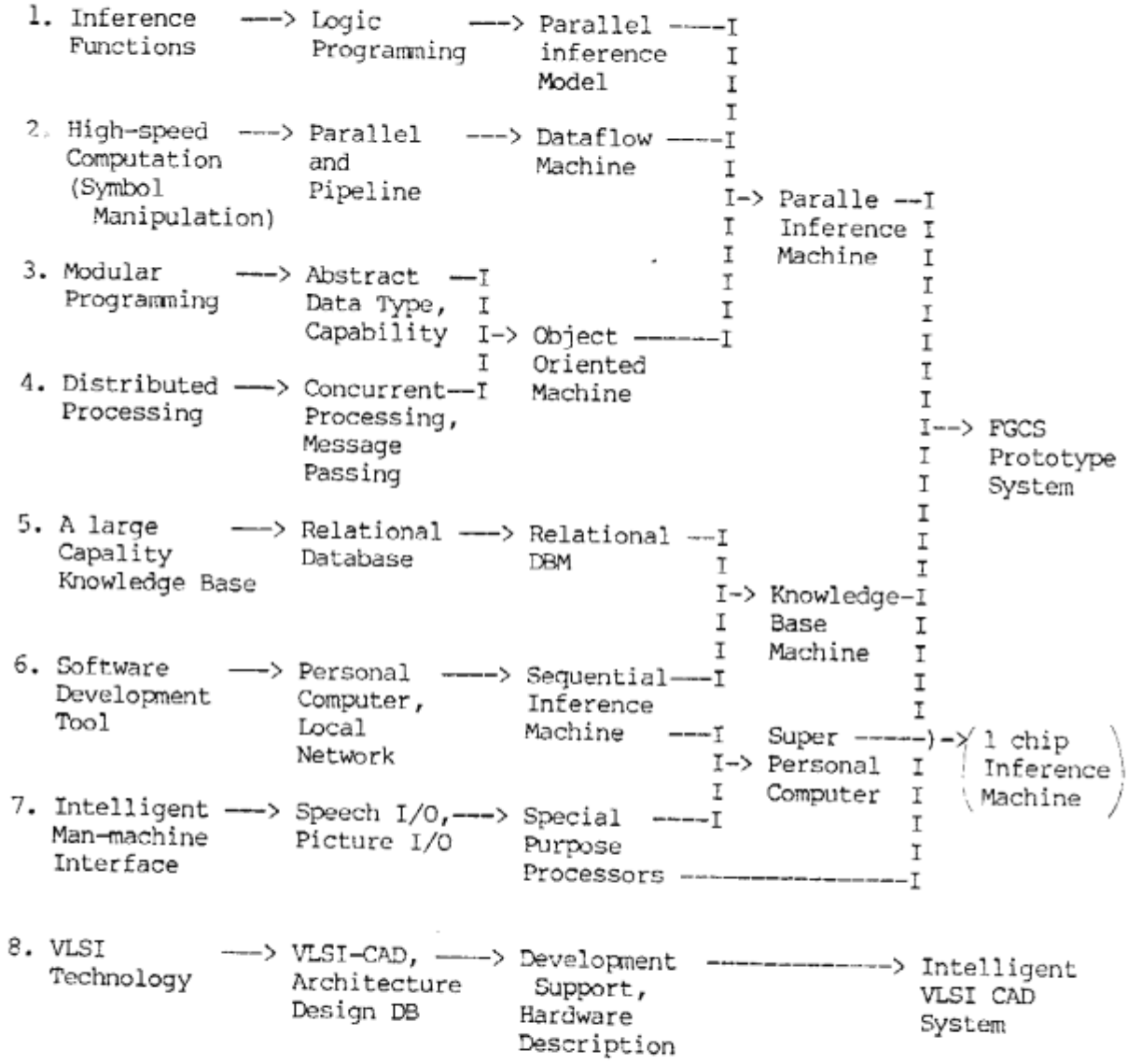

Fig.- 1. An Approach to the Fifth Generation Computer System