Report on Visit to ICOT
February 15-March 6, 1987

Evan Tick
Computer Systems Laboratory
Stanford University

Introduction

This report summarizes my three week visit to ICOT to exchange ideas on the design and analysis of sequential and parallel inference machines. The visit centered around research in the Fourth Research Laboratory: the Psi machine, Multi-Psi machine, and PIM projects. The visit was beneficial to me because it gave me the opportunity to learn in detail about current and future research, and convinced me that I would enjoy working at ICOT if I get the opportunity. I believe the visit was helpful to the Fourth Laboratory because of the numerous discussions we had concerning architecture and memory design methodology.

The atmosphere at ICOT was friendly and flexible, which made it easy for me to quickly adjust (from my "university lifestyle") and accomplish a great deal. In this respect, I must apologize for not having contributed as much as I received. Although I spent much time with numerous researchers, discussing and understanding their work, I failed to contribute any solutions to the important problems facing the PIM project. However, possibly I infused a sense of importance to the verification of design performance, not simply correctness, during the design process.

This report first summarizes the presentations I gave and laboratory visits I attended. An overview is then given of the discussions I participated in at ICOT. This overview is structured in terms of the Fourth Laboratory's research plan, and includes my detailed impressions. Conclusions are then discussed.

Presentations and Visits

During my visit, I gave two lectures concerning my doctoral thesis, "Studies In Prolog Architectures." The first lecture was given to the Second and Fourth Research Groups and was well attended by both ICOT members and university researchers. I was honored that Professor Tanaka of Tokyo University and Professor Tomita of Kyushu University attended. This lecture concentrated on the design of uniprocessor and multiprocessor memory organizations for Prolog and Restricted-AND Parallel Prolog, respectively.

I also gave a shortened version of the presentation at Fujitsu Research Laboratory in Kawasaki. I became interested in visiting Fujitsu because of Kumon-san's lecture (in Japanese) the previous week at ICOT on performance measurements of FGHC benchmarks. I was impressed by the researchers at Fujitsu, many who I had met previously at conferences - they were very knowledgable about Prolog and FGHC architecture design and although we were limited for time, we had an enlightening discussion.

Discussions at ICOT

Because I am currently attempting to graduate from Stanford University, I am quite familar with the incessant criticisms of my thesis committee. Of course, I understand that no matter how much extra work correcting these deficiencies entails, the importance of these criticisms cannot be understated! For these reasons, I am now

an expert critic, and I apologize if my criticisms in discussions with ICOT members extended beyond what can be accomplished in a single ten-year FGCS Project!

Currently, with the Psi-I, Psi-II and SIMPOS completed, the Second and Fourth Laboratories are concentrating their efforts on Multi-Psi, PIM, and PIMOS. These projects are closely related because Multi-Psi will be used as a testbed for PIMOS, the operating system for PIM. A six processing element (PE) Version-1 (Psi-I) Multi-Psi has already been completed, but is not yet stable or user-friendly enough to present demonstrations on. Taki-san demonstrated a simulation of Multi-Psi, running on a Psi-I. A Multi-Psi PE represents a PIM cluster in these experiments. A PIM cluster, however, will be a high-performance shared-memory multiprocessor of about ten PIM PEs. The Version-2 Multi-Psi, with Psi-IIs, will offer higher performance and thereby allow more extensive experimentation.

I had many interesting discussions with Nakajima-san, the designer of the Psi-II, about the Psi-II datapaths, micro-controller and memory organization. He explained the microcode in more than adequate detail and also the most recent performance evaluation conducted on the prototype. I was disappointed that only simple benchmarks were measured to date. I presented the Fourth Research Laboratory with all my Prolog benchmarks, so I am looking forward to some more extensive measurements.

In addition, comparisons of Psi-II performance with Psi-I has been stressed, which I think is short-sighted. There are other Prolog implementations, such as UC Berkeley PLM machine, that deserve attention. Comparisons with Lisp machines are also of great importance. I sometimes get the impression that ICOT narrows its focus too sharply on its own research path, and as a result fails to anticipate the emergence of alternative paths. It is interesting to note that Psi-II offers only 20% - 90% improvement in execution speed over the WAM microcoded Psi-I. In the university environment, this small improvement would not be considered sufficient reason to build a new machine. Certainly for Multi-Psi, a factor of two in the performance of the PEs will not make a significant difference in PIMOS experiments, where a Multi-Psi PE represents a much faster PIM cluster anyway. The Psi-II was in fact designed primarily to reduce the size of the Psi-I, and thus improve its cost-performance; however, I believe that higher performance should have been the first goal.

The Psi machines have advantage of being microcoded, so they can be temporary hosts for KL1-B. However, implementing KL1-B on these hosts is not optimal and therefore, a new architecture is being designed for the PIM PE, based on a reduced instruction set. The PIM PE architecture under design by Goto-san, is based on Kimura-san's abstract KL1 machine instruction set. It attempts to reduce the complexity of the CPU by simplifying the machine instructions. To keep code size low, dynamic macro-expansion of certain high-level abstract machine instructions is necessary; however, the overall size of this "internal code store" is significantly smaller than the Psi-II microstore, for instance. Performance improvement over Psi-II is not a primary concern, however, cluster size is.

I had many interesting discussions with Kimura-san and Goto-san concerning these issues, and firmly believe this is the right direction. The Psi-I and Psi-II are microcoded machines where little effort was made designing or measuring the instruction set requirements of logic programming languages (Psi-II has hundreds of instructions). The high-level definition of the WAM can be defended by its requirement of non-determinate execution and bi-directional unification. Without these attributes, KL1 is a simpler language and the abstract KL1 machine instruction set may be best suited to a RISC-like host. I believe that more thought should be given to

compiler optimization techniques for KL1-B. Indexing and inter-procedural optimizations can prove to have a significant impact of execution performance. In addition, as is pointed out below, optimizations may be necessary to make the KL1-B execution strategy more efficient.

The most important design criteria of the KL1-B architecture is to incorporate an efficient storage model, to ensure high locality and therefore high performance execution. Kumon-san and Kishimoto-san of Fujitsu have pointed out two major problems with the current KL1-B approach. The first is that the execution mechanism is inefficient, and that even with bounded depth-first goal reduction, frequent suspensions cause frequent context switching (i.e., attempting to reduce the next goal record in the ready-queue). Because the KL1-B instruction set is based on a register file to capture a single goal record, context switching is expensive. This implies that a single register-set based KL1-B model is not optimal, and that a contour-stack model or register-file set model may offer higher performance. Chikayama-san also suggested a dual-register set design. Further investigation, of the type I conducted at Stanford, is necessary to select the most cost-effective memory organization for KL1-B.

Fujitsu researchers are currently attempting to define a higher-level language to solve the first problem. In a discussion with Ueda-san, it was also stressed that a compiler could be expected to automatically "collect" inefficient fine-grain parallelism into larger-grains, for instance through partial-evaluation techniques. I agree that the importance of compiler optimization is of great importance, but think that in addition to source-to-source optimizations, low-level optimizations are possible. A reduced instruction set implementation of the KL1 abstract machine may allow static analysis to reduce the number of context switches. I had a interesting discussion with Miyazaki-san and Chikayama-san concerning such possibilities.

An even more important problem is that the KL1-B data storage model is centered around a heap which exhibits little locality, growing out-of-bounds without garbage collection. Chikayama-san explained his proposed solution to this problem - a novel single-bit reference count protocol (MRB) for facilitating garbage collection. I await empirical results confirming the effectiveness of this scheme. It is truely a luxury to have private tutoring sessions for this and many other upcoming ICLP and SLP papers!

Other important design criteria of the PIM PE instruction set are to ensure small code size with simple decoding, and to minimize pipeline breaks with efficient branch instructions. This area of research is also underway for Prolog (Mills at Arizona, Borriello at Berkeley) and FCP (Alkalaj at Weizmann). Within a year, I expect some performance results for these architectures to be available, and I except they will be successful.

Related to this research are the coherent cache and locking protocols of the PIM cluster, being designed by Matsumoto-san and Nishida-san. Their design is based on currently proposed distributed snoopy (broadcast) cache protocols. Currently, the coherent cache simulator takes its input from a trace-file produced by Sato-san's PIM emulator on the Balance. I had several informative discussions with these researchers concerning the detailed lock protocols and PIM emulator construction. I also spent some time with Sato-san trying to bring my own Prolog emulator up on the Balance, but this was only a partial success.

A possible alternative is to simulate the coherent caches on the Balance during KL1 emulation, to give more accurate statistics

(involving time). A related problem is the lack of measurement hooks in the PIM emulator. More importantly, time measured is Balance execution time. Since the ratio of PE computation:communication cost ratio is different for the Balance and PIM, a more informative methodology is to calculate "virtual PIM cycles". Currently however, the only estimate of this is KL1 Psi-II microcode cyles. More accurate estimates, possibly from hand-timings of the assumed PIM PE dataflow, are needed. I have proposed implementing a cache simulators on the Balance in my NSF proposal, so I would greatly appreciate it if Sato-san and Matsumoto-san save me the trouble.

I agree that broadcast coherent caches are necessary for KL1 because of the frequent communication between processes. Thus the expensive ICOT cache design is justified. Some critics have called ICOT's PIM design "conservative" implying negative connotations. The term "conservative" is used synomously with "previously proved effective" hence, why is research necessary? I disagree with this criticism. First, shared-memory systems offer high-performance using current technology. Certainly ICOT cannot design machines with hypothesized, future technology. Second, development of cost-efficient shared-memory multiprocessors still requires much research - to design inexpensive, well-balanced systems that avoid bottlenecks in the bus and memory. Currently there is no concrete plan for how the clusters should be interconnected. I think this indicates a sound design, because by clustering processor power, thereby reducing the number of clusters, inter-cluster topology is not a technologically difficult problem. Alternative one-layer designs, which interconnect a large group of PEs, are failing to take advantage of this. I think critics of ICOT's lack of a cluster interconnection design do not understand that quite to the contrary, the necessity to design "innovative" large-scale interconnections networks is a failure not a success. If ICOT succeeds in efficiently running single applications on 100 high-performance PEs, this will be a major success.

Sato-san is largely concerned with goal distribution strategies within a PIM cluster, and is currently making performance measurements. I agree that goal distribution is of primary importance, but I am also concerned with questions such as "How much can performance be improved by 'stylistic' changes in the benchmarks?" and "How does the execution of these benchmarks written in Prolog (or Lisp) compare, i.e., how many times faster than a good sequential implementation?" In other words, it is important to qualify, from every perspective, what it is PIM performance is being compared to.

In addition to the previous "low-level" research, Ichiyoshi-san and Rokusawa-san explained their work on the higher-level design of goal distribution protocols for Multi-Psi. The Multi-Psi machine is meant to be a testbed for PIMOS and its associated management functions for PIM. Thus a Multi-Psi PE models a PIM cluster. Another Multi-Psi project goal is of course an efficient Multi-Psi multiprocessor in its own right. These two goals are not entirely compatible because the communication:computation ratio for Multi-Psi and PIM are different. Therefore, what may be best for Multi-Psi, as far as goal distribution and external referencing protocols, may not be best for PIM. Ichiyoshi-san's passive and active external unification algorithms should be flexible enough to allow experimentation for PIM requirements, while still offering high-performance for Multi-Psi. Again, the concept of virtualization, of the measurement tool (Multi-Psi) to correspond to target system (PIM), is critical here.

At the very highest-level, an discussion about the "future of parallel inference machines" was held between Dr. Buchberger of Kepler University, Chikayama-san, Goto-san, and myself. Dr. Buchberger stressed that additional attention must be devoted to goal distribution stategies. In his opinion, this was one of the most pressing problem facing the Fourth Research Laboratory. We often

loose sight of really large problems that can alter machine performance by orders of magnitude, when searching for local optimizations that give, by comparison, only a few percent improvement. Even the design of special purpose PEs for PIM can be considered such "local minima" when compared to impact of load balancing on performance. Chikayama-san explained his ideas about a "PE power plain", but I was disappointed that little progress was made since a previous discussion during the Manchester conference last summer. The PIM organization is a two-level hierarchy, so I believe that goal distribution should be similarly managed at two levels; however, the programmers model should remain independent of the PIM structure. Thus I am concerned that the pragma, which are being developed for operation within a cluster, will not operate effectively when crossing cluster boundaries. In terms of the power plain model, this effect can be implemented by treating certain nearest neighbors differently than others, but of course, much additional work is needed to convert these ideas into a working scheme.

Conclusions

Overall, my impression of ICOT was extremely favorable, but people may say that this is because I like Japan so much. I don't agree - I have also visited a few "parent companies" and by comparison appreciate the freedom at ICOT. Of course, I missed taking the afternoon off to ride my bicycle in the Foothills, but that is a problem with Tokyo, not ICOT. I can make no accurate assertions concerning how the structure of ICOT affects the communication between and the efficiency of ICOT researchers. This is primarly because I do not understand what is being said around me. I could compare ICOT to a university environment, or an American start-up evironment, but these are not fair comparisons because the resources are different here. For instance, I think that an electronic bulletin-board would allow quick communication between researchers in the different laboratories, and create a forum for critical discussion; however, I understand that there are problems with Kanji on the systems currently being used.

My most negative impression is that sometimes research is conducted from the bottom-up without sufficient attention to the Big Picture. For instance, I would pose three questions (there are many others, some alluded to above):

> Can one write applications in KL1 (or derivatives)?
>
> How reliable is designing architectures and hardware organizations with a weak set of benchmarks?
>
> Can the FGHC approach compete with current AND/OR-parallel implementations of Prolog?

Although concern was expressed by ICOT researchers over the first two questions, I do not think enough is being done to answer them. Both the DEC-10 FGHC compiler and algorithmic debugger are written in Prolog. The Psi-II micro-simulator was implemented in Pascal. The KL1-B Multi-Psi compiler is written in Prolog. The Fujitsu KL1-B interpreter is written in C. I understand the necessity to develop efficient tools; however, tools are also important applications. In addition to efficiency problems, it is also true that FGHC programs are difficult to write (or understand in my opinion). The time should be invested in developing medium-sized applications "by hand", like the network router program, without awaiting the development of a higher-level KL2 language. This is especially necessary if the KL2 language is meant to be compiled into KL1, so that it must first be shown that efficient code can be generated with KL1.

I was impressed by the large attendance of a Multi-Psi general meeting

- over 40 researchers attended from all levels of the design (hardware up to language design) - this interest helps develop a highly coherent and unified research goal. For me, this participation seems incredible, after spending the last five years at Stanford where I am the only researcher involved in this field. Of course, I am disappointed that I cannot speak Japanese and therefore these meetings have little value for me. In my private discussions with ICOT researchers, everyone has made a great effort to speak with me (in both languages) - for this I am very grateful.

I think that ICOT's schedule is so tight that the researchers are under too much pressure. As a result, fundamental design methods, such as experimentation with simulated architecture designs and analytical performance modeling, are skipped. This can be illustrated with Psi-I design, which I do not think had a valid foundation. The WAM architecture was then adopted, after implementation in Psi-I micro-code, where its performance was simply compared against Psi-I code. Thus PSI-II was designed with few experimental results about the execution and memory referencing characteristics of large Prolog programs. PSI-II had the flexibility to avoid various design decisions because it was a general-purpose micro-coded machine, with a large micro-store and hundreds of high-level instructions. However, Psi-II organization indicates that few if any experiments were made concerning memory design. The Psi-II micro-simulator collects no statistics. The cache was designed with the primary criteria of board space. No choice point buffer was included as in the PLM, when even superficial statistics indicate its importance. I hope that the new family of KL1-B machines are designed with proper experimentation. In the case of these architectures, experimentation is more crucial to success than for Psi-II because of two main reasons. First, multiprocessor PE cache performance is critical in a shared memory system. Since the PIM PE has limited space, a trade-off must be made between cache size, local memory size and internal code memory size. Second, the locality of KL1-B appears to be much lower than the WAM for instance, so that an instruction set which allows extensive compiler optimization and garbage collection to increase locality is essential. I hope that the time is invested to properly instrument the KL1-B emulators to supply statistics with which an optimal architecture can be designed.

Critics of the Fourth Laboratory's research plan may fail to appreciate the significance of PIM; however, if successful, I think PIM would be very significant. I define a successful PIM as delivering 10-15 MLIPS for 100 PEs. A lower target may be in severe competition with a single cluster of high-performance parallel Prolog engines. As technologies improve, the PIM design could be scaled in performance at three levels: PE, cluster and inter-cluster network (allowing increased numbers of clusters). However, can a single application be efficiently, dynamically distributed across the PIM? And if so, can such applications be easily developed? These problems are intimately related, and therefore should be solved by a single group of researchers. Possibly, those currently working on these problems at ICOT are too widely distributed.

Acknowledgements

I thank the many people at ICOT who made my stay in Tokyo as enjoyable as my previous visit to Japan. I am grateful to Fuchi-san, director of ICOT, for the invitation to visit ICOT, and to Uchida-san, chief of the Second and Fourth Research Laboratories. I especially thank Kusama-san, ICOT research manager, and my hosts, Nakajima-san and Kimura-san, for making the preparations necessary to make the three week stay comfortable. Finally, I also thank Kimura-san for organizing my trip to Fujitsu Research Laboratory, and for his hospitality.

Even without onsen, keirin, or skiing, this "working trip" was very much a vacation because I greatly enjoy Tokyo. Most memorable were the numerous excursions to restaurants, pubs, soba shops, cafes, takoyaki stands, bars, and ryootei. With luck, I will be returning to ICOT at the end of this summer, not as a visitor, but as a member. At that time I hope to make a more significant research contribution, and of course do all those other things.

Name:	Tick, Evan
Program/Advisor:	EEPHD / Flynn, Michael
Date/place of birth:	30 April, 1959 / New York, New York
Citizenship:	U.S.
Home address:	2171 Manzanita Avenue, Menlo Park, California 94025
Home phone:	(415) 854-0691
Office address:	408D ERL, Stanford
Office phone:	(415) 497-0377 723
Electronic mail:	tick@shasta.stanford.edu
Undergraduate school:	MIT
Graduate school:	MIT Stanford University
Degrees conferred:	B.S., Electrical Engineering, MIT, 1982 M.S., Electrical Engineering, MIT, 1982
Honors:	IBM Graduate Fellowship 1985,1986 member Sigma Xi member Tau Beta Pi member Eta Kappa Nu National Merit Scholar, 1978 Westinghouse Science Talent Search (4th place) 1978

Publications:

H. Mulder and E. Tick, *Performance Comarison Between PLM and an MC68020 Prolog Processor*, TR 86-302, CSL Stanford University, August 1986.

E. Tick, *Memory Performance of Lisp and Prolog Programs*, Third International Conference on Logic Programming, London, July 1986.

E. Tick, *Lisp and Prolog Memory Performance*, TR 86-291, CSL Stanford University, January 1986.

E. Tick, *Prolog Memory-Referencing Behavior*, TR 85-281, CSL Stanford University, September 1985.

E. Tick, *Sequential Prolog Machine: Image and Host Architectures*, Seventeenth Annual Microprogramming Workshop, New Orleans, October 1984.

E. Tick, *Towards a Multiple Pipeline Prolog Processor*, International Workshop on High-Level Computer Architecture, Los Angeles, May 1984.

E. Tick and D. H. D. Warren, *Towards a Pipelined Prolog Processor*, International Symposium on Logic Programming, Atlantic City, February 1984.

E. Tick, *An Overlapped Prolog Processor*, TR-308, SRI AI Center, Menlo Park, August 1983.

Interest Areas:	High-end computer architecture, parallel logic and functional programming, natural language processing.
Date available:	June, 1987
Hobbies:	Bicycling, industrial music, windsurfing, skiing, fixing cars.
Dissertation Title:	Studies in Prolog Architectures
Dissertation Area:	Computer Architecture

Dissertation Abstract:

This dissertation addresses the problem of how logic programs can be made to execute at high speeds. Prolog, chosen as a representative logic programming language, differs from procedural languages, in that it is applicative, nondeterminate and uses pattern matching as its primary operation. Program performance is equated here with memory performance because high speed processors are ultimately limited by memory bandwidth and architectures that require less bandwidth have greater potential for high performance.

A derivation of a family of canonical Prolog architectures is given from the first principles of ideal

machine architectures. The WAM architecture, designed by Warren, is shown to be a member of this family. Measurements of the Prolog Canonical Interpretive Form indicate upper memory performance bounds afforded by ideal attributes. Analysis of high-level architecture statistics indicate the cost of attributes such as nondeterminacy and point to efficient memory designs.

High speed uniprocessor performance is necessary, even within a multiprocessor, because not all types of parallelism exist or can be exploited in all applications. Within a shared memory multiprocessor, local processor memories are necessary to reduce bandwidth and allow undegraded execution of sequential code. Two-level hierarchies for both sequential and parallel Prolog architectures are modeled. Local memories are measured using trace driven simulations. Main memories are measured using open queueing models. For sequential Prolog, various hierarchies are analyzed using the WAM architecture. For parallel Prolog, various shared memory multiprocessors are analyzed, using the Restricted AND Parallelism (RAP) architecture of Hermenegildo. Solutions to the local memory consistency problem are given for RAP.

Other research work: Lisp-Prolog Comparison - both languages were compared for a subset of the Gabriel benchmarks. This work is ongoing and gives interesting insights into the performance gaps between logic and functional programming.

PLM-M68020 Comparison - both architectures were compared to determine the performance gaps between specialize Prolog hardware and state-of-the-art generic microprocessors. This work is ongoing.

Career goals: Academic Position or Research Position

Employment experience:

Sep 1982 - present	Research Assistant, Stanford University
Jun 1984 - present	Quintus Computer Systems, Inc.
Jun 1983 - Sep 1983	SRI International, AI Center
Jun 1981 - Jun 1982	IBM, T. J. Watson Research Center