ICOT Technical Memorandum: TM-0299

TM-0299

Co-operative High Performance Sequential Inference Machine : CHI

by
S. Habata, R. Nakazaki, A. Konagaya,
A. Atarashi and M. Umemura (NEC)

May, 1987

ICOT
Mita Kokusai Bldg. 21F
4 28 Mita 1-Chome
Minato ku Tokyo 108 Japan
(03) 456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

CO-OPERATIVE HIGH PERFORMANCE SEQUENTIAL INFERENCE MACHINE : CHI

S.Habata,R.Nakazaki,A.Konagaya,A.Atarashi and M.Umemura

C&C Systems Research Laboratories,NEC Corporation
4-1-1 Miyazaki Miyamae-ku,Kawasaki 213,Japan

Abstract

A Co-operative High performance sequential Inference (CHI) machine has been developed within the Fifth Generation Computer Systems (FGCS) project in Japan. This paper describes the CHI-II (advanced version CHI) design approach, its characteristics and performance estimation. The CHI-II was designed for personal use and to be as small as a desk-side computer. In the design, Prolog oriented specialized hardware and machine instruction code optimization technique were considered for attaining high performance in Prolog program execution. CHI-II performance is estimated to be at 490 KLIPS for deterministic append program execution.

1 Introduction

Logic-programming language Prolog has been considered to provide useful capabilities in an Artificial Intelligence field, and has been used to develop practical application programs[1]. However, on conventional computers, the Prolog programs require many execution steps, and use much memory space. Therefore, it is necessary to develop a specialized machine, which executes the Prolog programs efficiently.

The Co-operative High performance sequential Inference (CHI) machine has been developed within the Fifth Generation Computer Systems (FGCS) project in Japan. In the project, CHI-I (prototype CHI)[2] and CHI-II (advanced version CHI) have been developed.

CHIs, including CHI-I and CHI-II, are compiler-oriented machines to realize high performance Prolog processors. Their architecture is based on Warren Abstract Machine (WAM)[3]. They are implemented as back-end high performance Prolog machines with large main memory capacity for large-scale practical Prolog programs.

CHI-I is an experimental machine to prove its architecture suitable for realizing a high performance Prolog processor. It is implemented with high-speed CML (Current Mode Logic) circuits. Therefore, it has attained 285 KLIPS (Kilo Logical Inferences Per Second) in deterministic append program execution[2]. However, its size is too great for personal use. When evaluating the architecture and hardware configuration, the authors found a few improvable design points.

Therefore, CHI-II was planned to be a practical machine for personal use and to have more powerful performance than CHI-I. The CHI-II is designed as small as a desk-side computer, and is implemented with CMOS gate-array LSIs and off-the-shelf TTL ICs. This paper describes CHI-II design approach and its characteristics.

2 CHI-II design approach

CHI-I has about 80,000 gates in processor and 64 mega words (1 word : 36 bits) in main memory. Its machine cycle time is 100 nano-second. CHI-II is designed as a desk-side machine with 128 mega-word (1 word : 40 bits) main memory and with more functions in the processor than those in the CHI-I processor.

To realize a desk-side machine, 2 CMOS gate-array LSIs (totally about 20,000 gates) have been developed in the CHI-II processor, and 1 mega-bit D-RAMs are used as main memory.

To realize a high performance machine, the following three points were considered.

- Machine cycle time
- Multiple function execution in a micro cycle
- Compiler optimization technique

2.1 Machine cycle time

Machine cycle time is a strict factor to decide processor performance. CHI-II machine cycle time is restricted by its cache memory cycle time. In Prolog program execution, memory accesses occur very frequently, and much memory space is required. Even if plural instructions can be parallelly executed, most fundamental Prolog operations require sequential data accesses. For example, variable dereferencing operation requires step by step memory access to trace a reference chain sequentially. Therefore, the ideal program execution time is coincident with the total memory access time

required to execute the program. The CHI-II processor is designed to execute many micro operations in a micro step, in order to eliminate micro steps without memory access. Then, the memory access ratio, between memory access count and micro steps, required in machine instruction or program execution, was measured to estimate the hardware function suitability. When the memory access ratio is 1, and the machine cycle time is equal to the cache memory access time, the machine design is ideal. Therefore, CHI-II machine cycle time (170 nS) was determined, based on cache memory access time (about 150 nS).

2.2 Multiple function execution in a micro cycle

CHI hardware was designed on the basis of tagged-architecture, and its data format consists of two fields. One is for specifying data types, called a *tag* field, and the other is for data values, called a *value* field. In CHI-II processor, *tag* field is 8bits in length, and *value* field is 32 bits in length.

CHI-II processor provides Prolog language oriented hardware functions: tag check, value comparison and argument access. It also provides conventional hardware functions: instruction prefetching and microprogram multi-way branching. In CHI-II, these operations are executed simultaneously in a micro step, as many as possible, in order that the memory access ratio may be equal to 1. For example, on an argument matching operation, which is the most characteristic operation in Prolog program execution, the CHI-II processor compares *tags* and *values*, respectively, in a single micro cycle.

2.3 Compile optimization technique

Clause-indexing instructions[3] markedly reduce the number of machine instructions to be executed, by checking the first argument *tag* field in a predicate. Two kinds of new clause-indexing instructions are introduced in CHI-II, in order to take advantage of CHI-II multiple function execution in a micro cycle. One of the newly introduced clause-indexing instructions combines get instructions, for the first argument, with an original clause-indexing instruction. These instructions are as follow.

```
switch_on_list_register_variable
switch_on_list_register_value
```

The other includes a kind of *delayed branch type* instructions, as follow.

```
delayed_switch_on_term
delayed_switch_on_list
delayed_switch_on_list_register_variable
delayed_switch_on_list_register_value
delayed_execute
```

```
append(□,X,X).
append([X|A],B,[X|C]):-append(A,B,C).
```

(a). Append program in Prolog

```
$app:  switch_on_term              $nil,$list,$1  ; * 4 μ steps
$1:    try_me_else                 $2             ;
$nil:  get_nil                     A0             ;
       get_register_value          A1,A2          ;
       proceed                                    ;
$2:    trust_me_else_fail                         ;
$list: get_list                    A0             ; * 1 μ step
       unify_register_variable     A3             ; * 2 μ steps
       unify_register_variable     A0             ; * 2 μ steps
       get_list                    A2             ; * 4 μ steps
       unify_register_value        A3             ; * 2 μ steps
       unify_register_variable     A2             ; * 2 μ steps
       execute                     $app           ; * 3 μ steps

* instruction executed in the 2nd clause
```

(b). Machine program without using the newly introduced instructions

```
$app:  switch_on_list_register_variable   A3,$nil,$list,$1  ;
$1:    try_me_else                        $2                ;
       get_nil                                              ;
$nil:  get_register_value                 A1,A2             ;
       proceed                                              ;
$2:    trust_me_else_fail                                   ;
       get_list_register_variable_variable A0,A3,A0        ;
       get_list_register_value_variable   A2,A3,A2         ;
       switch_on_list_register_variable   A3,$nil,$list,$1 ;
$list: unify_register_variable            A0                ; * 2 μ steps
       get_list_register_value_variable   A2,A3,A2          ; * 6 μ steps
       switch_on_list_register_variable   A3,$nil,$list,$1  ; * 4 μ steps

* instruction executed in the 2nd clause
```

(c). Machine instructions using the newly introduced instructions

Figure 1: Append Program in CHI-II machine

Extended unification instructions for list data are also introduced in CHI-II.

```
get_list_register_variable_variable
get_list_register_value_variable
unify_list_register_variable_variable
unify_list_register_value_variable      etc.
```

Each new clause-indexing and extended unification instruction replaces several machine instructions. For example, in append program execution, 5 machine instructions and 8 micro steps are reduced in every logical inference, by using above new instructions (as shown in Figure 1).

3 CHI-II configuration

CHI-II system provides an excellent Prolog program development and execution environment for a single user, including multi-processing and multi-window environment. It consists of a host machine and CHI-II machine (as shown in Figure 2).

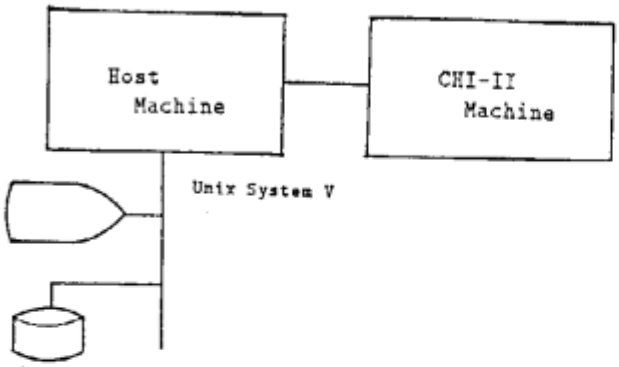

Figure 2: CHI-II System Configuration

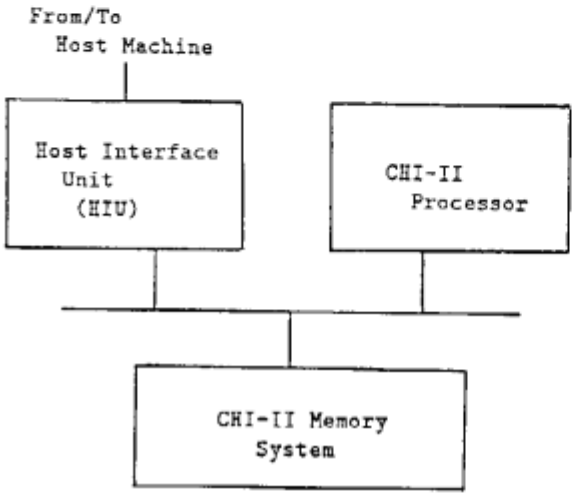

Figure 3: CHI-II Machine Configuration

A host machine controls peripheral devices, including man-machine interface control and CHI-II file management on secondary storage devices. An engineering workstation with UNIX system V is used as the host machine. CHI-II machine is composed of a high performance processor (CHI-II processor), a large capacity main memory (CHI-II memory system) and a host interface unit (HIU) (as shown in Figure 3)

3.1 CHI-II processor

CHI-II processor provides two ALUs, one for address calculation and the other for data computation, and an argument comparator for data type check (as shown in Figure 4).

Therefore, it can execute address computation, *value* comparison and *tag* check in a micro cycle. It also provides a multiport (4 read ports and 2 write ports) register file, composed of 32 data registers and 32 control registers. The data registers specified by instruction code can be accessed in a micro step. They are used as argument registers.[3] The control registers are used to control stack operation, backtracking and predicate

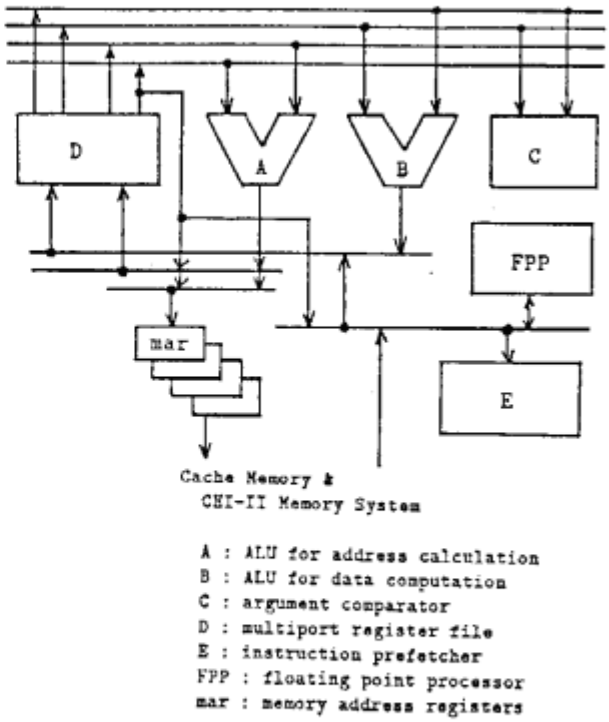

Figure 4: CHI-II Processor Configuration

call operation.

Most machine instructions can be executed in two or three micro steps in CHI-II processor. A few instructions can be executed in a micro step. The authors have developed two CMOS gate-array LSIs for CHI-II processor. One is a micro sequence controller (about 13,000 gates) and the other is an instruction code prefetch controller (about 6,000 gates).

3.2 CHI-II memory system

CHI-II memory system consists of a mapping unit and main memory. The mapping unit translates logical addresses to real addresses. In CHI-II machine, 8 independent logical spaces are required to execute Prolog program. These spaces are called:

system control area:	for system information and micro work area
code area:	for executable machine code
common data area:	for shared data
system object area:	for system resource
local stack area:	
global stack area:	
trail stack area:	for undo control
local heap area:	for process information

The main memory provides a 128-mega-word memory space with error correction. It is implemented with 1 mega bit D-RAMs, and high-density packaging technology. Therefore, a single main memory board, with surface-mounted memory ICs on both sides, provides 40-mega-byte memory capacity (8 mega words). There are 16 main memory boards in CHI-II machine.

Table 1: Memory Access Ratio in typical 18 machine instructions

Machine Instruction	Micro Step Count CHI-I	Micro Step Count CHI-II	Memory Access Count	Memory Access Ratio CHI-I	Memory Access Ratio CHI-II
get_permanent_variable	2	2	2	1.000	1.000
get_temporary_variable	2	1	1	0.500	1.000
get_permanent_value	5	3	2	0.600	0.667
get_temporary_value	3	1	1	0.667	1.000
get_list	2	1	1	0.500	1.000
get_structure	6	3	2	0.500	0.667
put_permanent_variable	3	2	2	0.667	1.000
put_temporary_variable	3	2	2	0.667	1.000
put_permanent_value	4	3	2	0.500	0.667
put_temporary_value	3	1	1	0.333	1.000
put_list	3	1	1	0.333	1.000
put_structure	5	2	2	0.400	1.000
unify_permanent_variable	4	3	3	0.750	1.000
unify_temporary_variable	4	2	2	0.500	1.000
unify_permanent_value	5	3	3	0.600	1.000
unify_temporary_value	5	2	2	0.400	1.000
unify_list	2	1	1	0.500	1.000
unify_structure	6	3	2	0.333	0.667
total	67	6	32	—	—
average	3.7	2.0	1.8	0.478	0.889

3.3 Host Interface Unit

The HIU controls asynchronous bi-directional communications between the host machine and CHI-II machine. It also converts CHI-II 40-bit-data into 32-bit-data used in the host machine, and vice versa. Therefore, the host machine is considered as an excellent I/O processor for the CHI-II processor. It provides file management function, man-machine interface function and CHI-II hardware monitoring function.

4 Performance estimation

CHI-II performance estimation is based on microprogram execution for machine instructions and a deterministic append program. The number of micro steps in each machine instruction was counted to compare CHI-II and CHI-I performance (as shown in Table 1).

In most machine instructions, the number of micro steps in CHI-II is smaller than that required in CHI-I, when the cache hit ratio is 100%, because of the improvement in multiple function execution in CHI-II. The average memory access ratio, in typical 18 machine instructions execution, is 0.889 in CHI-II, while it is 0.478 in CHI-I.

In append program execution, 8 machine instructions and 20 micro steps are required for a logical inference in CHI-II. The memory access ratio is 0.850. In CHI-I, 9 machine instructions and 35 micro steps are required, and its memory access ratio is 0.457. Therefore, in spite of long machine cycle time, 170 nS, CHI-II performance is estimated to be at 294 KLIPS in append program execution, while it is estimated as 285 KLIPS in CHI-I (as shown in Table 2).

Moreover, using the compile optimization technique, CHI-II performance is improved at 490 KLIPS in append program execution. Then, 3 machine instructions and 12 micro steps are required, and the memory access ratio is improved at 0.917.

Table 2: Performance Estimation in append program execution

	Machine Instruction Count/LI	Micro Step Count/LI	Memory Access Ratio	Performance (KLIPS)
CHI-I	9	35	0.457	285
CHI-II without compile optimization	8	20	0.850	294
CHI-II with compile optimization	3	12	0.917	490

5 Conclusion

CHI-II has been designed and implemented as a practical desk-side computer. It has improved hardware functions and machine architecture for realizing a machine superior to CHI-I. Moreover, as a result of improvement in the memory access ratio, applying compile optimization technique, CHI-II realizes a powerful Prolog processor. In append program execution, CHI-II performance is estimated to be at 490 KLIPS.

The authors consider that, if a few mega bit SRAM and 100,000 gates could be integrated on a chip, the ideal CHI processor could be realized, resulting in a practical Prolog processor with several MLIPS (Mega LIPS).

6 Acknowledgment

The authors would like to express their thanks to Shunichi Uchida (ICOT), to Yasuo Kato and Tatsuo Ishiguro (NEC) for their continuous encouragement and to Masahiro Yamamoto (NEC) for valuable advice and support. Thanks are also due to Takashi Chikayama and Kazuo Taki (ICOT), as well as to Minoru Yokota and Noriko Kijima (NEC) for their instructive discussions and machine implementation researching efforts.

References

[1] F.Maruyama, et al. *Prolog-Based Expert System for Logic Design*, Proc. of the International Conference on FGCS 1984, Nov. 1984

[2] R.Nakazaki, et al. *Design of a High-speed Prolog Machine (HPM)*, Proc. of the 12th International Symposium on Computer Architecture, June 1985

[3] D.H.D.Warren, *An Abstract Prolog Instruction Set*, Tech. report 309, Artificial Intelligence Center, SRI International, 1983