Two-Level Controlled Parallel Reconfigurable Architecture

Takanobu Baba
Center for Optical Research and Education
Utsunomiya University
Utsunomiya, Japan 321-8585
baba@cc.utsunomiya-u.ac.jp

Kanemitsu Ootsu
Graduate School of Engineering
Utsunomiya University
Utsunomiya, Japan 321-8585
kim@is.utsunomiya-u.ac.jp

***Abstract*— After reviewing the key technologies of microprogramming, this paper focuses on the utilization of two-level microprogramming scheme combined with multiprocessor parallelism. Based on our experience by the development of the two-level microprogrammed multiprocessor machine, called MUNAP, and the increasing importance of reconfigurable parallel architecture, we propose a new two-level controlled, parallel reconfigurable architecture. This architecture is expected to realize flexible control of many fine-grained operational units by distributed nanoprograms of horizontal type under single microprogram of vertical type. It also reduces the difficulty of designing applications of reconfigurable hardware by replacing a lot of sequence control circuits with nanoprograms. By isolating the programming view from the detailed implementation view and allowing the flexibility of the underlying hardware, the proposed scheme will be much more palatable to the application developers than the use of an HDL and logic compiler scheme.**

Keywords—two-level microprogramming; reconfigurable computing; parallel computer

I. INTRODUCTION

Since microprogramming was invented by M.V. Wilkes in 1951 as a means for systematically designing computer's control part [1], the technology was utilized by commercially available computers to realize a wide variety of hardware organizations under the same instruction set architecture and keep compatibility of architecture by bridging the gap between the instruction set definition and hardware architecture [2]. By the use of semiconductor memory as control storage, the microprogram became writeable or reloadable and *dynamic microprogramming* attracted our attentions [3], [4]. System support functions, such as micro-diagnostics, and application-oriented functions, such as elementary function evaluation and programming language processing, were realized as microprograms [2]. The synthesis of microprograms for frequently executed instructions' sequence and their optimization were also studied [5], [6]. The term *dynamic architecture* represents the adaptable nature. The term *firmware* was coined as the intermediate layer between software and hardware [7]. Types of micro-instruction set were categorized into *vertical* and *horizontal* types, depending on their degree of encoding – from direct control to highly-encoded. It affects the complexity of decoding, parallelism, and control memory amount [8]. A *two-level microprogramming* scheme was proposed as a combination of upper-level vertical micro-instruction and lower-level horizontal one [8]. The aim is to save the total control memory amount without sacrificing the parallelism of horizontal micro-instructions

The basic features of micro-instruction set level architecture can be summarized as follows:

- A micro-instruction is basically executed in one-machine cycle by pipelining micro-instruction fetch and execution.
- Decoded signals, called *micro-order*, directly control register-level hardware behaviors, such as gate and ALU operations, called *micro-operations*.
- Single micro-instruction controls not only usual arithmetic and memory operations but also sequence of micro-instructions in various ways. Example sequencing functions include a branch by using an operation code of machine instruction and four-way branch by using two-kinds of tests.
- The control signal generation may be changed directly by a hardware resource value, such as some counter (called *residual control*), and other field of the same micro-instruction (called *indirect control*).

These features affect the later architectural innovations. Many of the microprogramming technologies have been well inherited both by RISC and CISC architectures [9]. The VLIW architecture also inherits the features of horizontal micro-instructions that control a lot of hardware resources directly in parallel [10]. The local and global optimization technologies [11], [12], developed for the microprograms, have been utilized for the RISC and VLIW compilers.

The objective of this paper is to review the technologies of micro-instruction set level architecture and show the possibility of two-level microprogramming scheme coupled with multiprocessor architecture for implementing two-level controlled reconfigurable architecture.

The rest of this paper is organized as follows. Section 2 reviews the two-level microprogramming scheme and describes a two-level microprogrammed multiprocessor

architecture. It also describes the implementation and evaluation of the architecture. Section 3 proposes a new parallel architecture with fine-grained reconfiguration capability under the two-level control scheme. Section 4 concludes the paper.

II. Two-Level Microprogrammed Multiprocessor Architecture

A. Review of two-level microprogramming

In the two-level microprogramming scheme, the control memory is divided into two parts, namely, *microprogram* memory (MPM) and *nanoprogram* memory (NPM). There are two types of relations, as shown in Fig. 1.

In the type of Fig.1(a), the *micro-instructions* (μIs) in the microprogram memory are narrow and of vertical type. They have pointers to *nano-instructions* (nIs) in the nanoprogram memory. They also perform global control of the machine, such as sequence control, data transfer at the micro-level and, in some cases, literal values to be assigned to specified registers. The nano-instructions are wide and of horizontal type. They directly control functional units, such as multiple ALUs and registers in parallel. This scheme requires less control memory when frequently executed short micro-instructions refer to the same long nano-instruction. Examples of this architecture include the Burroughs Interpreter [8] and the Motorola MC68020 [13].

In the second type of Fig.1(b), the nano-instructions in the lower level storage unit interpret micro-instructions, in much the same way as ordinary micro-instructions interpret machine language instructions. This scheme allows us to define the meaning of upper level micro-instructions by the lower level nanoprograms (nPs). The Nanodata QM-1 was an example of this type [14].

B. Two-level microprogrammed multiprocessor architecture

Adoption of the two-level control structure coupled with the multiprocessor parallelism lead us to an attractive architecture as shown in Fig. 2. In this architecture a micro-instruction (μI) in the microprogram memory (MPM) specifies a nanoprogram address and the address is multicast to several

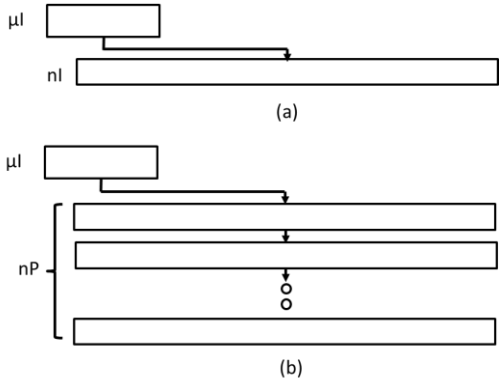

Fig. 1. Two-level microprogramming scheme.

tightly coupled multiprocessor units to activate the multi-nanoprograms. The microprogram also coordinates the execution of activated multi-nanoprograms, controls global dataflow between processing units (PUs) and main memory (MM), and performs sequence control of whole system.

Within PU, a nano-instruction performs the local control of the nano-operations. They include arithmetic and logical operations by ALU, and non-numeric operations by bit operation unit and divide and concatenate unit. A small amount of high-speed memory is provided as a scratchpad memory. PU inputs (or outputs) data from/to micro-level through port registers.

We named this machine MUNAP which stands for MUlti-NAnoProgram machine [15], [16]. The control scheme is expected to have the following effects:

1. Flexibility for meeting a wide range of applications' requirements through two-levels of control;
2. Utilization of maximum parallelism of intra- and inter-multiprocessor units under single microprogram control;
3. Savings in total amount of control storage by having nano-instructions that are common to more than one microinstcution;
4. Modularity of hardware units under the distributed control function of nanoprograms; and
5. Closeness between the control and controlled parts by distributing control functions to multiprocessors.

First two items allow a single micro-instruction to be applied to any combination of the processing units for each unique application of the operation. The processing units may perform both SIMD and MIMD operations in a uniform

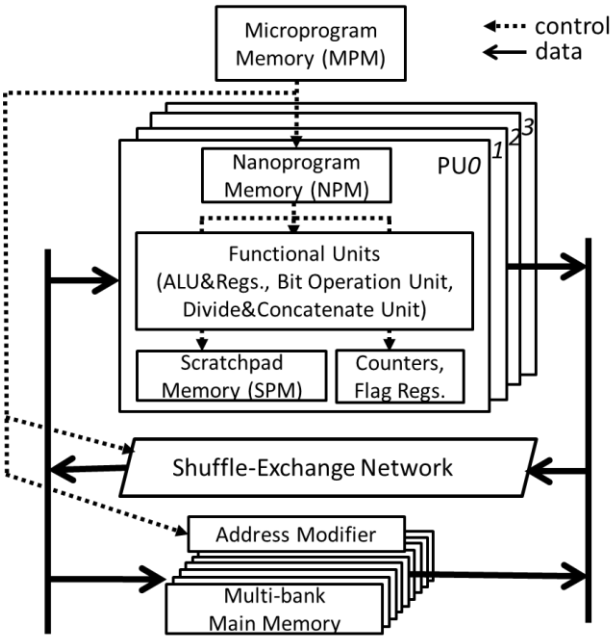

Fig. 2. Two-level microrogrammed multiprocessor.

manner under the single micro-instruction stream. The third through fifth items have advantages over usual single level control scheme from the view point of VLSI implementation.

In order to implement the architecture, we have designed micro-instruction and nano-instruction formats. Except for a few micro-instructions, such as sequencing one, the micro-instructions have an operation and operands fields in addition to nanoprogram control fields which include the nanoprogram start address and the specification of the PUs to be activated. In addition to an operation and operands fields, every nano-instruction has one bit in common, called ENP (End of NanoProgram), which indicates the end of activated nanoprogram. This means that the activated nanoprograms may run arbitrary steps until ENP becomes 1. Thus, the nanoprogram in each processing unit acts like Fig 1(a) when the ENP of the first nano-instruction is 1; otherwise, it acts like Fig.1 (b) under the control of nano-level sequencer in PU.

The interaction mechanism between the micro- and nano-levels is shown in Fig. 3. The micro-nano flags (MNFL) are placed as an interface and play an important role by sending and receiving information between two levels. It consists of the following six kinds of flags:

- Nanohalt (NHLT): the end of nanoprograms of PUs;
- Microrequest (MREQ): the request from micro-level to nanoprograms;
- Test (TEST): the result of tests by nanoprograms;
- Nanorequest (NREQ): the request from nanoprograms to microprogram;
- Nanointerrupt (NINT): the interrupt caused by nanoprogram execution; and
- Microinterrupt (MINT): the interrupt caused by microprogram execution.

This interaction mechanism may be used for the following three operations: (i) activation of nanoprograms from micro-level and notifying the termination of nanoprograms from nano- to micro-level, (ii) transferring the test results from nano- to micro-level, and (iii) requesting from one-level to another. These allow the two-levels of control programs to interact each other flexibly.

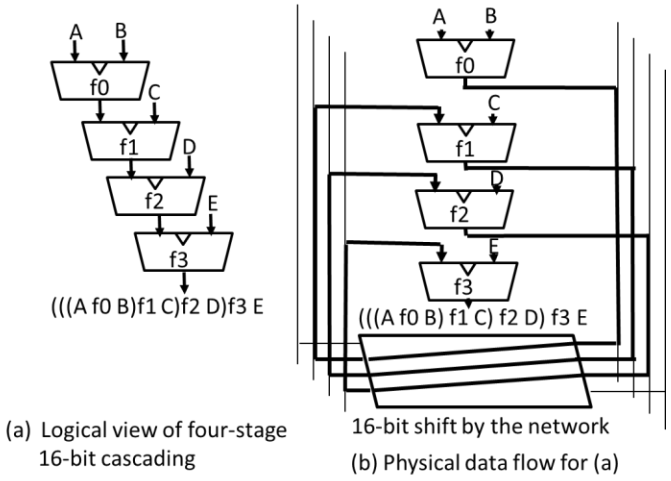

Fig. 4. Cascading of four PUs' functions and its physical implementation.

We would like to note that micro-nano combined data-transfer mechanism and the shuffle-exchange network's data-exchange functions allow us to reconfigure the cascading of multiple-PU operations by ALU and non-numeric units as shown in Fig. 4. Fig.4 (a) illustrates the cascading of functions f0 through f3 in PU0 through PU3, respectively, and (b) shows how the cascading is realized physically by using the shuffle-exchange network. Other cascading operations may be realized by using the same hardware [16].

C. Implementation and evaluation of the architecrure

The designed architecture had been implemented by using about 3,000 ICs and connecting them by wire wrapping. The supporting software systems include the translator for the two-level microprogram description language, optimizing loader

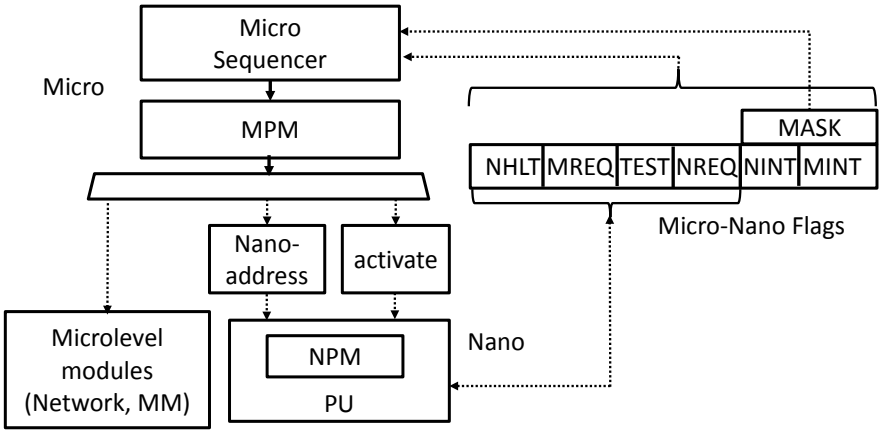

Fig. 3. Miro-nano interaction mechanism.

for compacting and loading the translated two-level microprograms, debugger and evaluator [16]. Then, we applied the machine to various applications, such as language processors for Prolog and Smalltalk-80, three dimensional color graphics system, and numerical computations.

We evaluated the architecture based on these application results. As to the effect of two-level control, we can summarize from the following three points.

First, the usage frequencies of micro- and nano-instructions show the characteristics of the both levels. At the micro-level, the dynamic frequencies of micro-instructions for sequence control, data transfer, and MM access are 50-80%, 20-36%, and 5%, respectively. The highest ratio of sequence control micro-instructions indicates the importance of micro-level sequencing functions in the two-level microprogramming scheme. At the nano-level, the dynamic frequencies of nano-instuctions for ALU, data transfer between the two levels, the nano-level sequence controls, and constant generations are 23-50%, 20%, 13%, and 7-18%, respectively. The usage frequencies of non-numeric operational units vary, depending on applications' characteristics. At the stage of architecture design, we expected the nanoprograms would mainly control ALU and other functional units and would not use the nano-level sequence control function so much. However, the results show that the nano-level sequencing was used to detect the PU status and transfer the result to micro-level in order to reflect the detected result to micro-level sequence control. The nano-level sequencing is also used when long nanoprograms run independently.

Second, the average number of nanoprogram steps, activated by one micro-instruction, indicates how long the nanoprograms can run independently. The experimental results show that the nano-steps for MIMD type processing were small (around 1.1 to 1.5), while those for SIMD type processing were large (7.7 for FFT and 78.5 for LU decomposition). The reason is that the data in the processing units for MIMD processing are mutually dependent and the result of other processing units is needed after a few steps of nanoprogram execution. In SIMD type processing, the processing units are mutually independent. Thus, once necessary data are fetched from the main memory to the scratchpad memory, the nanoprogram can continue their execution for a relatively long time period. The control is returned to micro-level only when accessing the main memory or controlling sequences, such as the call/return microprogram subroutines.

Third, the applications clarified that 86% of nanoprogram memory was effectively used. This high ratio was realized by optimization of nanoprogram address space compaction [16].

These results indicate that the proposed architecture allows us to realize flexible but efficient control mechanism for the multiprocessors with fine grained parallelism.

III. Two-Level Control Structure for Fine Grained Reconfiguration

Recently, the reconfigurable fabric for accelerating the processing of heavy workloads attracts our attention. For example, the Catapult fabric ports a significant fraction of Bing's ranking engine on to a ring structured 8 Field Programmable Gate Arrays (FPGAs) to accelerate the execution of ranking by macro-pipeline [17]. The Xeon and FPGA combined platform has been developed as a heterogeneous computing engine [18]. The FPGA accelerator is expected to complement CPU cores to meet market needs for performance of diverse workloads in the data center. An example usage of such architecture is a high frequency trading accelerator for achieving the lowest possible latency for interpreting market data feeds [19]. An FPGA-based accelerator for deep convolution neural networks is another example application of reconfigurable architecture [20].

These reconfigurable architectures using FPGAs share quite similar objective with dynamically microprogrammable machines, described above. The standing position of the both approaches is between the hardware implementation by an application-specific integrated circuits (ASIC) and the software implementation. The ASIC realizes high performance execution. However, it lacks the capability of reconfigurability and fast development round to meet various applications' requirements. It is hard for the software implementation to meet the performance requirement of heavy workloads. Both the reconfigurable architecture and microprogrammable machines bridge the gap between these two extreme implementations by utilizing register-transfer level parallelism.

A major challenge of FPGA development is the requirement for extensive hand coding using RTL language to define many state machines and manual tuning of the designed configurations to meet the application's requirements [17]. We need to describe the hardware by using a hardware description language (HDL), generate the configuration data, and download it to FPGA. We should repeat these processes for tuning the design.

Thus, as a solution to this problem of designing difficulty of FPGAs, we would like to propose a new parallel reconfigurable architecture that utilizes the two levels of microprogram control. In this architecture, sequence circuit design, which may occur at many places on FPGA, can be replaced with the description of nanoprogram of horizontal type that controls fine-grained operational unit array in parallel. The nanoprogram description and its specification from single-stream microprogram may ease the development of applications on FPGA devices.

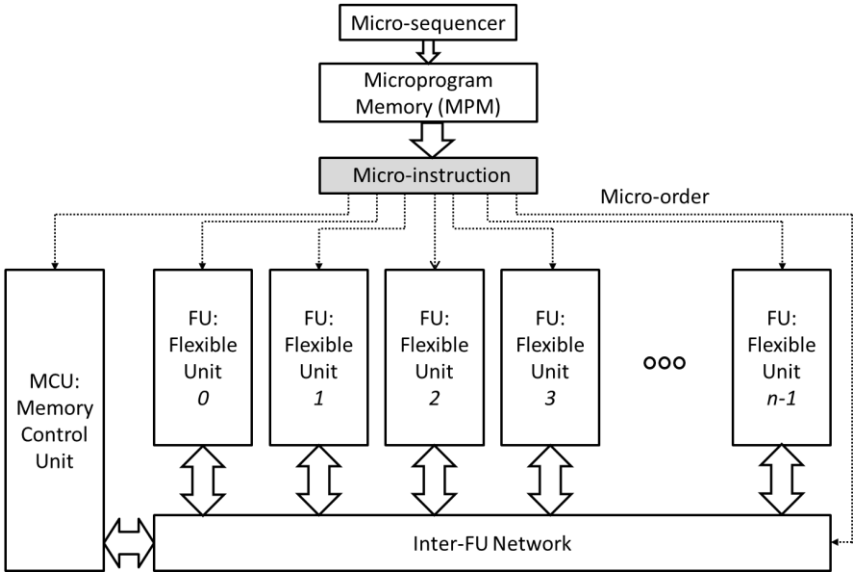

Fig. 5. Top view of parallel reconfigurable architecture

Fig. 5 shows the top view of the proposed architecture. It corresponds to the micro-level architecture of MUNAP. The micro-instruction is read from MPM and decoded to generate control signals, so called *micro-order*. It is sent to multiple Flexible Units (FUs) to activate nanoprogram executions. It also controls micro-level data transfers between FUs, memory and inter-FU network. Thus, various combinations of FUs' operations can be realized as shown in Fig. 4.

Fig. 6 shows the internal organization of the FUs. It receives micro-order from micro-level and determines the nanoprogram start address. The very long nano-instruction of horizontal type is decoded to generate control signals, so called *nano-order*. They control the operational unit array and other resources of FU. The operational unit array can be viewed as an array of fine grained ALUs or non-numeric units with 4- to 16-bit length or some special functional units, such as elementary function evaluation units. Usually, FPGA includes many sequence control mechanisms to realize many state machines. In our scheme, the sequence of nanoprograms replaces these mechanisms for sequencing. Thus, a complex sequence can be realized by nanoprograms systematically as proposed by Wilkes [1]. The nanoprogram realizes reconfiguration of the operational units and intra-FU network so that the architecture may adapt to the applications' requirements. This reconfiguration process is much like rewriting the configuration data of FPGA devices. By providing ready-made configuration data and its usage as nanoprograms, the user can easily utilize reconfiguration capability just by specifying the nanoprogram address from the micro-level.

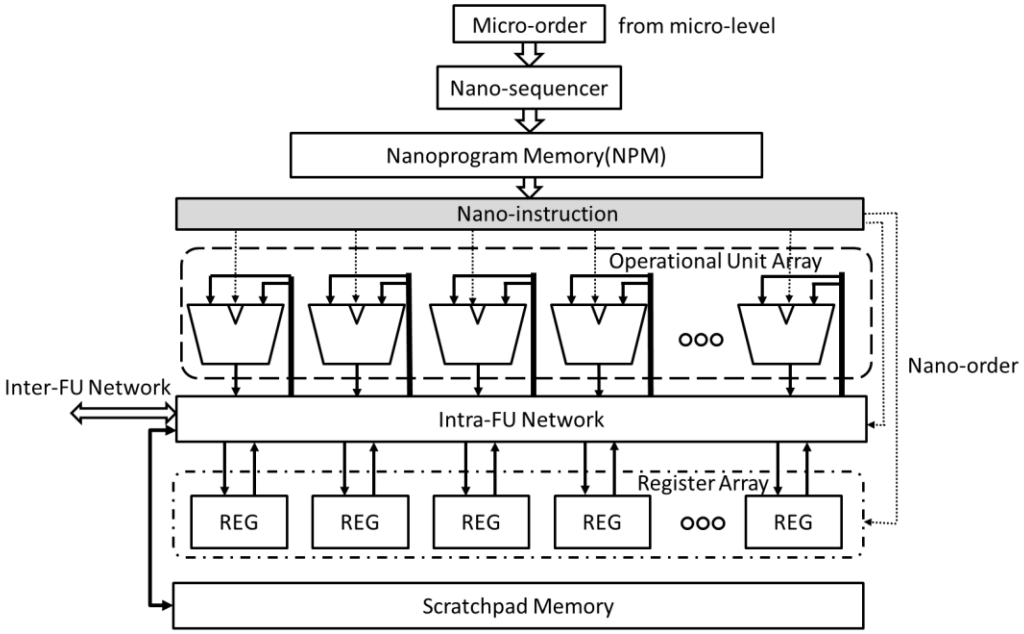

Fig. 6. Internal organization of FUs.

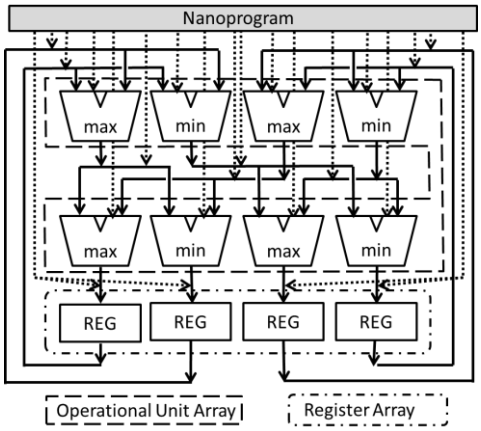

Fig. 7. Nanoprogram controlled sub-sorting operation.

In order to illustrate our image of using the two-level microprogram, Fig. 7 shows an example sorting operation controlled by the nanoprogram. The operational unit array, intra-FU network and register array are controlled by nanoinstructions of horizontal type to sort 4 data items in REGs. The bit-length of the operation may be flexibly adapted to the application's requirement. The microprogram coordinates the nanoprogram controlled sub-sorting operations to sort the whole data.

In summary, the two-level control scheme has advantages of 5 items described in section II (b). The proposed two-level controlled parallel and reconfigurable architecture share these advantages and, further, it has additional advantage for reconfigurable architecture by providing multi-nanoprograms of horizontal type which replace many complex sequence control circuits, necessary for usual FPGA design. Thus, following the Wilkes' original motivation of providing a systematic way to design the control part of a computer, our two-level control scheme allows the user to utilize reconfigurable, fine-grained parallel processing capability by describing micro- and nano-programs.

IV. Conclusion

First, we reviewed the key technologies of microprogramming. Then, we focused on the utilization of two-level microprogramming scheme combined with multiprocessor parallelism. Based on our experience by the development of the two-level microprogrammed multiprocessor machine, called MUNAP, we proposed a new two-level controlled, parallel reconfigurable architecture in order to ease the development of applications on such architecture. The proposed architecture is expected to realize flexible control of many fine-grained operational units by distributed nanoprograms under single microprogram stream. Further, the architecture will reduce the difficulty of designing applications on reconfigurable hardware by replacing a lot of sequence control circuits with nanoprograms.

Note that our proposed architecture is independent of its implementation details. It may be realized by using a *hard* integrated circuit or a *soft* FPGA. In any case, it isolates the programming view from the detailed implementation view and, at the same time, allows the flexibility of the underlying hardware. Thus, our scheme will be much more palatable to the application developers than the use of an HDL and logic compiler scheme.

Acknowledgment

We would like to thank the reviewers for their helpful comments. This work was supported by JSPS KAKENHI Grant Numbers 25330055 and 15K00068.

References

[1] M.V. Wilkes, "The Best Way to Design an Automatic Calculating Machine," Report of Manchester University Computer Inaugural Conference, pp.16-18, 1951.

[2] S.S. Husson: Microprogramming: Principles and Practices , Englewood Cliffs, N.J., Prentice-Hall, 1970.

[3] A.B. Tucker and M.J. Flynn, "Dynaamic Microprogramming : Processor Organization and Programming," Comm. ACM, Vol.14, No.4, 240-250, 1971.

[4] R.W. Cook and M.J. Flynn, "System Design of a Dynamic Microprocessor," IEEE Trans. Computers, Vol.C-19, No.3, pp.213-222, 1970.

[5] A.M. Abd-Alla, D.C. Karlgaard, "Heauristic Synthesis of Microprogrammed Computer Architecture," IEEE Trans. Computers, Vol.C-23, No.8, pp.802-807, 1974.

[6] T.G. Rauscher and A.K. Agrawala, "Dynamic Problem-Oriented Redefinition of Computer Architecture via Microprogramming," IEEE Trans. Computers, Vol.C-27, No.11, pp.1006-1014 , 1978.

[7] A. Opler, "Fourth Generation Software," Datamation, Vol.13, No.1, pp.22-24, 1967.

[8] A.K. Aglawala, T.G. Rauscher, Foundations of Microprogramming, New York:Academic, 1976.

[9] D. A. Patterson, D. R. Ditzel, "The case for the reduced instruction set computer," ACM SIGARCH Computer Architecture News. 8 (6): 25 -33, 1980.

[10] J.A. Fisher, "Very Long Instruction Word architectures and the ELI-512," Proc. of the 10th International Symposium on Computer Architecturepp. 140–150, 1983.

[11] T. Agerwala, "Microprogram Optimization: A Survey," IEEE Trans. Computers, Vol.C-25, pp.962-973, 1976.

[12] T. Baba and H. Hagiwara, "The MPG System: A Machine-Independent Efficient Microprogram Generator, IEEE Trans. On Computers, Vol.C-30, No.6, pp.373-395, 1981.

[13] D. MacGregor, D. Mothersole, and B. Moyer, "The Motorola MC68020," IEEE Micro, Vol.4, No.4, pp.101-118, 1984.

[14] R.F. Rosin, G. Frieder, and R.H. Echhouse, "An Environment for Research in Microprogramming and Emulation," Comm. ACM, Vol.15, No.8, pp.748-760, 1972.

[15] T. Baba, K. Ishikawa, and K. Okuda, "A Two-Level Microprogrammed Multiprocessor Computer with Non-numeric Functions," IEEE Trans. on Computers, Vol.C-31, No.12, pp.1142-1156, 1982.

[16] T. Baba, Microprogrammable Parallel Computer –MUNAP and Its Applications-, The MIT Press, p.290 , 1987.

[17] A. Putnam, et al., "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," ISCA2014, pp.13-24, 2014.

[18] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks," FPGA'15, pp.161-170, 2015.

[19] B. Leber, B. Geib, and H. Litz, "High Frequency Trading Acceleration Using FPGAs," FPL'11, pp.317-322, 2011.

[20] PK Gupta, "Xeon+FPGA Platform for the Data Center," ISCA2015. https://www.ece.cmu.edu/~calcm/carl/lib/exe/fetch.php?media=carl15-gupta.pdf