An Application Specific Java Processor with Reconfigurabilities

Shinji Kimura Hiroyuki Kida† Kazuyoshi Takagi‡ Tatsumori Abematsu* Katsumasa Watanabe

Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0101, JAPAN
†: Presently NIIT at Nara; ‡: Presently Nagoya University; *: Presently SHARP Corporation

Abstract— The paper presents an application specific Java processor including reconfigurabilities, which is a DLX like pipeline processor with 5 stages and executes Java byte codes directly. Reconfigurabilities are the key technologies for application specific operations. We have introduced two reconfigurabilities: (1) a mechanism to override the control signals for a specific instruction, (2) external components can be attached with the same input and output ports as the internal ALU.

I. Introduction

With the recent advance in the VLSI technology, embedded systems can include network interfaces and communication capabilities. Java language is one of major programming languages for such applications.

Java programs are converted to Java byte codes and executed by the Java virtual machine ([1], [2]). The Java virtual machine is implemented as interpreters or byte code compilers on many platforms, and Java byte codes have high portability. Java processors are direct implementations of the Java virtual machine, and are suitable for executing Java byte codes under limited hardware resources such as in embedded systems.

The paper presents a pipeline Java processor with reconfigurabilities for installing application specific instructions and application specific hardware modules.

II. Java Virtual Machine and Java Byte Code

The Java virtual machine is a virtual computer which executes Java byte codes directly using the internal stack. Each instruction has a one byte operation code and data operands with variable length (0 to 4 bytes). Up to 256 instructions can be specified and 201 instructions are specified at present ([2]).

Instructions are classified into 3 categories: (1) stack operations: push constants, load and push, pop and store, duplication of stack; (2) arithmetic operations for operands on the stack: addition, subtraction, multiplication, division, shift, logical operation, etc.; (3) execution control operations: jump, conditional branch, call and return for Java methods.

Basic data types manipulated by the Java virtual machine are int (32 bit), long (64 bit), float (32 bit) and double (64 bit). The internal stack with the 32 bit width is used to store intermediate results. The depth of the stack is not specified.

III. Reconfigurable Java Processor

A. Architecture

We have designed a reconfigurable pipeline Java processor (R-Java processor) for embedded systems. The processor has a pipeline structure with 5 stages like the DLX ([3]): IF stage, ID stage, EX stage, MEM stage and WB stage. The processor also includes a memory control unit, a data forwarding unit and an internal stack as shown in Fig. 1.

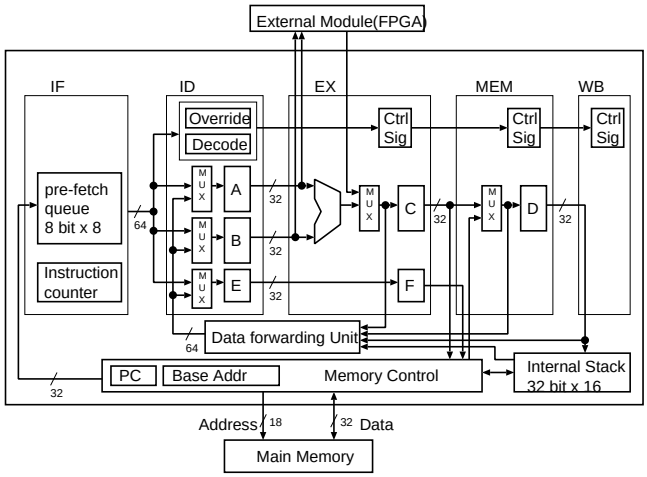

Fig. 1. Architecture of Java Processor

The width of the internal stack and other data path is 32 bit. The depth of the stack is 16 since the intermediate results are usually not so large. To cope with the stack overflow and underflow, we have installed a control mechanism to transfer data in the stack to and from the external memory. Logically, the depth of the stack is infinite.

Because of the limitation on the hardware resources, we have only implemented 117 instructions among 201 instructions. Un-implemented instructions invokes the internal trap and are executed by the operating software.

B. Execution Control

Instructions are fetched from the memory to the prefetch queue. 4 bytes of data are fetched with one clock. IF unit decides the end of each instruction and transfer the instruction to the ID stage. Note that the maximum length of an instruction is 5, and the length of the prefetch queue is set to 8 bytes.

ID unit generates 30 bit control signals and transfers the signals to the following units:

- type of operation (5 bits) to EX unit,
- information (14 bits) to Data forwarding unit,
- memory access information (2 bits) to MEM unit,
- stack push information (5 bit) to WB unit,
- Jump information (1 bit) to PC control unit, and
- information (3 bits) to prefetch instruction control.

In Fig. 1, “Ctrl Sig” denotes the transfered signals.

C. Reconfigurability in the Execution Control

The pipeline stages are controlled by signals generated by ID unit, and the interpretation of an instruction can be changed

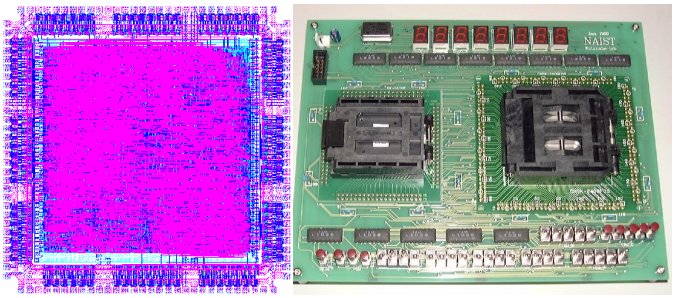

Fig. 2. Layout and a test board of Java LSI chip.

by overriding the generated control signals.

We have implemented the override module as in Fig. 1. The module has registers to keep overrode instructions and overrode control signals. For the module, we introduce special instructions for setting values for special registers. The module compares the executed instruction and the overrode instruction, and transfers overrode control signals when matched.

This reconfigurability is useful to install new instructions, such as un-specified instructions at present and instructions on using application specific hardware modules.

D. Interface to External Hardware Modules

On embedded systems, application specific hardware modules would be attached for speed-up. Thus input ports for ALU in the processor are passed to an external hardware module and the output port of ALU is multiplexed with that of the external module. We have also implemented a mechanism to stall the pipeline for multi-cycle external modules.

To cope with the cost for application specific modules, we recognize that the external module should include the reconfigurability such as in Field Programmable Gate Arrays (FPGAs). In our present state, we cannot include FPGA modules within the LSI and FPGA modules are attached outside the Java processor at the board level.

IV. Implementation and Evaluation

We have designed our reconfigurable Java processor using VHDL and implemented via VDEC Rohm LSI with 4.5 mm $\times$ 4.5 mm size, 0.6 μm process, 2 PolySi layers and 3 metal layers. The VHDL source includes about 4000 lines, and the LSI chip includes about 110000 transistors (28000 elements). Fig. 2 shows the chip layout. [1]

We have also implemented a test board for VDEC Rohm LSI as shown in Fig. 2. The Java LSI is in the left side socket and there is an FPGA in the right side socket. All I/O pins of the LSI are passed to FPGA. FPGA works as memory units, connections between LSI pins and peripherals such as switches and 7 segment diodes, and external hardware modules.

We have evaluated the correct behavior of the Java processor with 50 MHz clock on this board. Note that any VDEC-Rohm LSI chip with the same package can be tested on this board.

The execution time has been measured using the repetition of the following butterfly operation on integer variables in the Discrete Fourier Transform (DFT).

[1] The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center(VDEC), the University of Tokyo with the collaboration by Rohm Corporation and Toppan Printing Corporation.

TABLE I
CPU seconds for executing the butterfly operations

Processor type	clock cycle	10^6	25×10^6
R-Java wo ext. module	50 MHz	0.8	20.0
R-Java with ext. module	50 MHz	0.4	10.0
Pentium II Interpreter	412 MHz	1.8	27.4
Pentium II JIT	412 MHz	0.8	1.4

```
dr = s * iy + c * ry;   di = c * iy - s * ry;
rx += dr; ry = rx - dr; ix += di; iy = ix - di;
```

After the compilation, the function consists of 32 Java instructions with 8 pipeline hazards, and is executed with 40 clocks. On the other hand, we can construct a hardware module computing the above operation with 120 ns delay on Altera FPGA FLEX10K70. With this module, the function consists of 13 Java instructions with 1 pipeline hazard and 6 clock wait for the external module execution; total 20 clocks.

We have obtained 2 times speed-up for the DFT operation with the external module. Note that the invocation of the external module and the data acquisition from the external module are implemented as the instruction override mechanism.

We show the execution time of the above butterfly operation on our Java processor and other general processors. We have varied the number of repetitions of the above operation with 1 million and 25 million. Table I shows the result.

Note that R-Java 50 MHz with an external module is faster than the software interpreter on Pentium II 412 MHz and is competitive with Pentium II Just In Time Compiler. By JIT, the Java instructions are reduced to only 10 Pentium instructions.

Just In Time mechanism would also be useful in the R-Java processor. R-Java processor loads 8 instructions and we have a chance to execute several instructions as 1 instruction. This would be one of future works.

V. Conclusion

We have presented a reconfigurable Java processor for embedded systems. The processor has a pipeline modules with 5 stages and executes Java byte codes directly. The processor includes a mechanism to change the execution control for one (but any) specified instruction. The processor also includes I/O interfaces to use an external reconfigurable module.

Acknowledgments

We would like to thank members of Watanabe-lab at NAIST for their discussions. The work is supported in part by the NAIST IS fund 1998, by Grants in Aid for Scientific Research in Aid for Scientific Research from the Ministry of Education, Science, Sports and Culture (09245101, 11480068), and by a grant from Fujitsu Laboratory.

References

[1] T. Lindholm and F. Yellin. *The Java Virtual Machine Specification*. Addison Wesley, 1996.

[2] B. Venners. *Inside the Java Virtual Machine*. McGraw-Hill, 1998.

[3] D. A. Patterson and J. H. Hennessy. *Computer Architecture (2nd Ed.), A Quantitative Approach*. Morgan Kaufmann, ISBN 1-55860-329-8, 1996.