ASAver.1 : An FPGA-Based Education Board for Computer Architecture/System Design

Hiroyuki OCHI

Department of Computer Engineering
Faculty of Information Sciences
Hiroshima City University
Asaminami-Ku, Hiroshima, 731-31, Japan
Tel: +81-82-830-1550 Fax: +81-82-830-1792
e-mail: ochi@ce.hiroshima-cu.ac.jp

Abstract— This paper proposes a new approach that makes it possible for every undergraduate student to perform experiments of developing a *pipelined* RISC processor within limited time available for the course. The approach consists of 4 steps; at the first step, modeling of pipelined RISC processor is simplified by avoiding structural hazard and by ignoring other hazards, and in the succeeding steps, students learn difficulties of pipelining by themselves. An educational FPGA board *ASAver.1* and results of feasibility study are also shown.

I. Introduction

With the recent advances of integrated circuit technologies, very large and sophisticated systems can be implemented practically, and it is strongly desired to cultivate researchers/engineers with the ability of designing a huge and unique system. Various new approaches on hardware education are eagerly proposed, prepared, and put into practice, including [2, 3, 4, 7].

KUE-CHIP2 [2] developed by Ochi et al. is a simple 8-bit microprocessor of accumulator architecture with full control/observe facilities, which is useful at an introductory course of computers to help students understand how microprocessor works, by observing its behavior step-by-step. The KUE-CHIP2 board has many switches and LED displays to control and observe the KUE-CHIP2 microprocessor. In addition, the board has connectors to interconnect external circuitry to replace the on-board KUE-CHIP2, and is used for evaluating KUE-CHIP2-compatible circuitry designed by undergraduate students during the succeeding course of experiment on logic and architecture design, using devices such as TTL.

With these features, hardware education using KUE-CHIP2 has been very successful for many years. Because of the current trend in microprocessor architecture, from accumulator machine toward pipelined RISC processor, however, requirement for changing the target architecture of the course to the RISC architecture has been increased. To make it possible, a new approach should be developed, because pipelined RISC processors has difficulties for students at every steps, including understanding, designing, debugging, and running.

Recently, reconfigurable (e.g., SRAM-based) FPGAs come to attract the attention of educators, because (1) a system as complex as a 16-bit microprocessor can be realized within a single FPGA device, (2) FPGAs can be reconfigured quickly and makes debug/tuning process easy, and (3) FPGA-based education requires almost no running cost (except license fee for CAD tools,) as it can be reconfigured unlimited number of times. Sueyoshi developed an educational FPGA board *KITE* [3] which has a Xilinx XC4010-PG191 device, main memory, many switches and LED displays, and a host computer interface to read and write the main memory. For introductory course, a default configuration data is loaded into the FPGA and it works as the KITE processor, an observable 16-bit accumulator machine. At an advanced laboratory for undergraduate students, every student designs a microprocessor and implements it to the FPGA on the board by loading the generated configuration data. Nishimura et al. also developed an educational FPGA board for their students' experiments, and currently it is used to implement a 16-bit non-pipelined RISC microprocessor *PICO-16* [4].

The goal of this paper is

1. To consider a new education plan that enables undergraduate students to understand and develop 16-bit *pipelined* RISC microprocessors by themselves within limited time available for the course, and

2. To develop a new educational board which is useful for coherent education from introduction of computer to hardware and system, including the above experiment.

This paper proposes a 4-step approach that makes it possible to perform experiments of developing a pipelined RISC processor by every undergraduate student within

ASP-DAC '97
0-89791-851-7$5.00 © 1997 IEEE

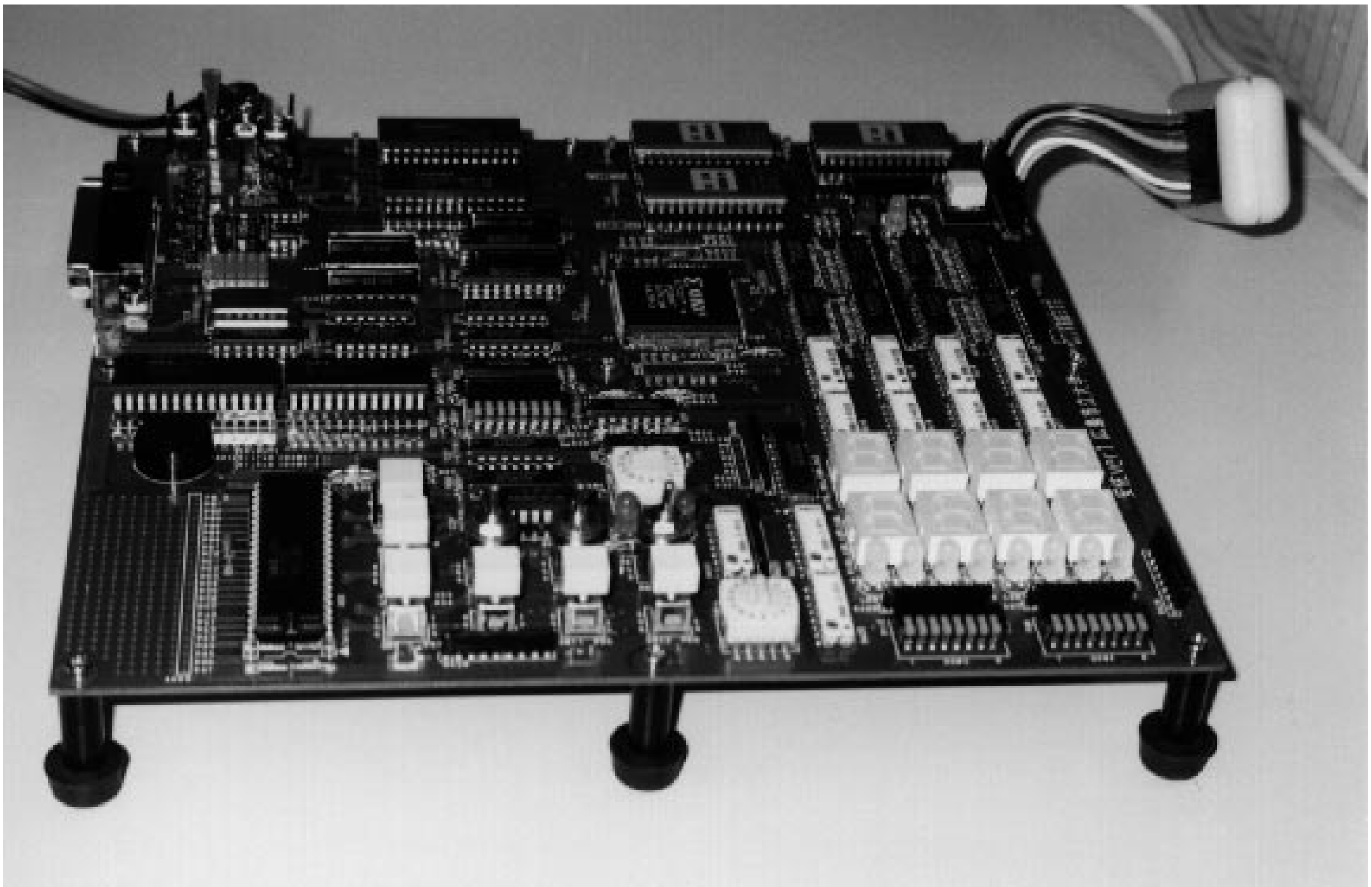


Fig. 1. Photograph of ASAver.1 Board

limited time for the course. The uniqueness of the proposed approach is in its strategy of guiding students toward complete understanding of pipelining. The approach consists of clear and completed steps, and at every step, students can verify their achievement by themselves and attain satisfaction. At the first step, every student implements by himself/herself a pipelined RISC processor which is based on a given, very simple model; The given processor model has separate buses for instruction and data memory to avoid structural hazard, while it completely ignores data and control hazards to make implementation easy. Although it is such a "defective" processor, we can test its functionality and even run application programs by giving object code containing sufficient number of `NOP` instructions to avoid hazards. In the second step, the developed processor is observed by himself/herself, and the defects of the processor are fixed by his/her own way in the third step. Finally, every student is requested to present how he/she improved the processor.

This paper also describes a new educational FPGA board *ASAver.1* (Fig. 1.) This board is versatile and useful. For the use at introductory course, it is designed to be as simple as possible (i.e., an FPGA, a single memory bus, switches, and LED displays) and can work stand-alone. For the advanced experiments based on the proposed approach, it has capability to have separate buses for instruction and data memory, by adding another memory board. For computer system experiments, it has sockets and interfaces for optional I/O devices.

To verify that the proposed approach is feasible for laboratory for undergraduate students, I implemented a prototype of a 16-bit pipelined RISC processor, referred to as *ASAP-0*, on the ASAver.1 board. The processor ASAP-0 is a pipelined (5-stage, 1-CPI) RISC processor with eight 16-bit general purpose registers, a 16-bit program counter, and a zero flag, with 10 essential instructions, described by 310 lines of Verilog-HDL. It is implemented to an FPGA device using Cadence Synergy and Xilinx Xact on Sun SPARCstation 20 workstation with about 1.5-hour runtime. The ASAP-0 processor implemented here is so compact that even it can be implemented within a so-called 6,000-gate-class FPGA device (XC4006 [9].) This means that the ASAver.1 board which has a so-called 15,000-gate-class FPGA device (XC5215 [8]) can provide sufficient amount of hardware resources for students to

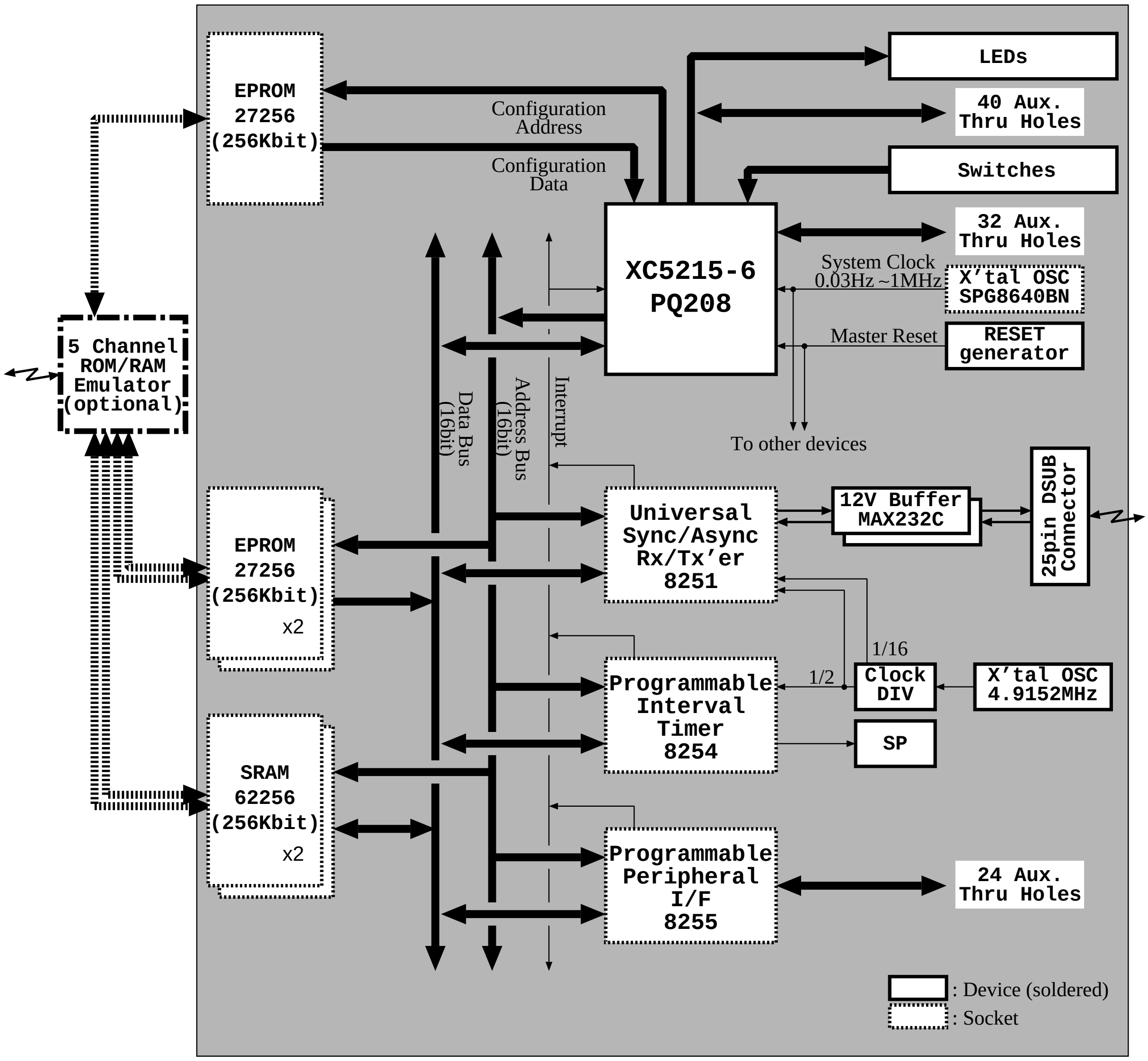


Fig. 2. Block Diagram of ASAver.1 Board

improve their designs.

In the next section, the description of equipments including the ASAver.1 board will appear. In Section 3, education approach of the computer architecture experimental course is proposed, and how the ASAver.1 board is used is described. In Section 4, the result of the feasibility study is presented. In Section 5, the education plan in our department is described.

II. Education Environment

A. ASAver.1 Board

The ASAver.1 board is a single-board system for experimenting and developing a computer with an FPGA-based processor.

The ASAver.1 board has the following features:

- A Xilinx XC5215-6 PQ208 FPGA device, which is said that it can realize a logic circuit of approximately 15,000 gates [8].
- Eight general purpose 7-segment LEDs with hexadecimal-to-7-segment decoder/drivers and 8 general purpose LEDs with drivers, which can be driven by programmable I/O pins of the FPGA device.
- 27-bit general purpose switches (DIP switches, toggle switches, a rotary DIP switch, and push button switches,) which can be monitored by programmable I/O pins of the FPGA device.

- A clock oscillator for the processor, whose frequency is easily varied by a rotary DIP switch from 0.03 Hz to 1 MHz. To apply higher frequency, optional 4-pin crystal oscillator can be inserted to the socket.
- 16-bit address bus and 16-bit data bus, which allows 16 bit × 65536 word memory space.
- Two 32 KB EPROM socket and two 32 KB SRAM socket, which are used for the main memory.
- A serial I/F device (i8251 compatible) with ready-made RS232c interface circuitry (including a baud-rate clock oscillator, ±12V buffers, and a 25-pin connector,) which enable us to perform experiments using RS232c serial port without any additional hardware.
- An interval timer device (i8254 compatible,) which is useful for generating interrupt signal and/or driving a speaker to buzz.
- A parallel I/F device (i8255 compatible) with universal board area, which enable us to connect optional peripherals such as a printer or something like stepping motors.
- 32 auxiliary through holes, which enable us to access 32 unused programmable I/O pins of the FPGA device.
- A connector for downloading configuration data from a host computer, and an EPROM socket for stand-alone configuration.

The block diagram of the board is shown in Fig. 2. Memory devices (except the configuration ROM) and I/O devices are connected to the FPGA via a 16-bit address bus and a 16-bit data bus, and thus this board is ready for use as a single board microcomputer. There are 32 programmable I/O pins of the FPGA which are completely unused, and we have access for those pins via 32 auxiliary through holes. These I/O pins increase the flexibility of the board.

B. ROM/RAM Emulator

To perform experiments efficiently, it is strongly desired to have facility to read and modify the contents of main memory. The KITE board [3] and the PICO-16 board [4] has a built-in circuitry to realize such function. In my case, I have developed a separate board, the 5-channel ROM/RAM emulator, in order to keep the ASAver.1 board to be an simple stand-alone computer so that it can be understood clearly by students.

This emulator emulates up to 5 ROM or RAM devices simultaneously. It is actually a Z80-based microcomputer with an RS232c interface to be accessed by a host computer, and the memory contents can be monitored and manipulated by the host computer.

III. Designing Pipelined RISC Processor

A. Previous Activities

In the standard course of computer architecture, it is desired for students to understand a pipelined RISC processor, namely the DLX architecture [1], and, if possible, a chance to develop and evaluate a real pipelined RISC processor should be given. There has been a project to develop a pipelined RISC processor in Universities [5, 6], however, it is considered to be the laboratory for graduate students.

Tomita and Nakashima [7] proposed a unique approach for undergraduate students to understand and develop a pipelined RISC processor. The target instruction set architecture is a 16-bit RISC architecture, named *SIMPLE*. At first, 5-CPI (non-pipelined) version of SIMPLE architecture is considered or designed, and succeedingly, 4-CPI, 3-CPI, and 2-CPI versions are discussed, and finally 1-CPI version is developed. It seems very interesting, however, it seems to take too long time to complete the whole steps.

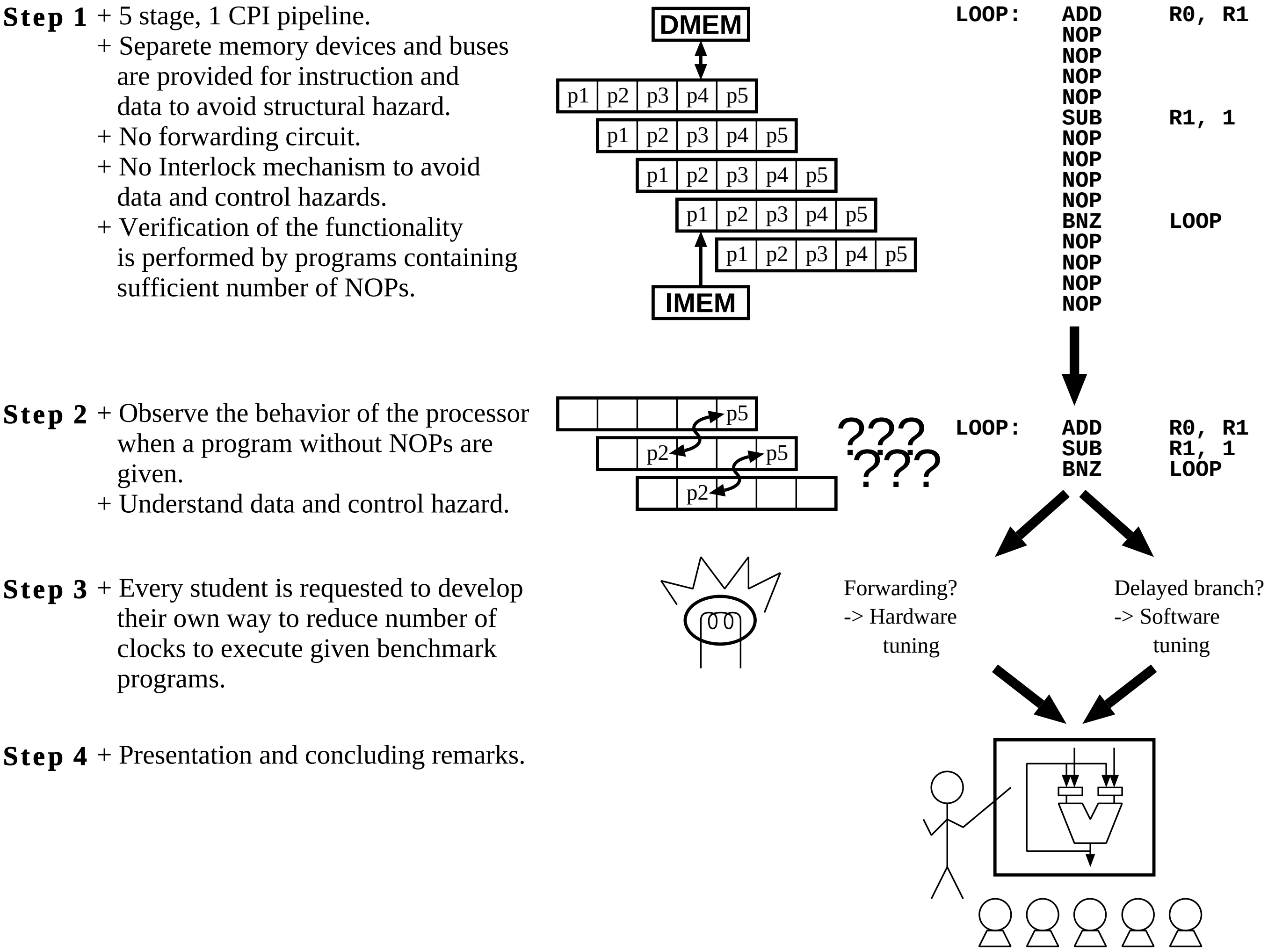
B. Proposed Approach

To make it possible to perform experiments of developing a pipelined RISC processor within limited time of the course, I will propose a novel 4-step approach. The outline of this approach is illustrated in Fig. 3.

Step 1. Design and implement a pipelined (1-CPI) RISC processor on an FPGA. The processor is based on a given, very simple model. To avoid structural hazard, separate memory devices and buses are provided for instruction and data. This memory architecture models the cache architecture of most RISC processors which has separate instruction cache and data cache. On the other hand, data hazard and control hazard are not considered, i.e., such complicated circuitry as forwarding path, interlock logic, and so on are not implemented at this step in order to make design simple. Although it is such a "defective" processor, we can test its functionality and even run application programs by giving object code containing sufficient number of `NOP` instructions to avoid hazards.

Step 2. Carefully observe the behavior of the processor when a program with reduced number of `NOP` instructions are given, and confirm the conditions when the processor works correctly. This experiment makes clear the difficulty of pipelining, and shows real examples of data hazard and control hazard.

Step 3. Evaluate the developed processor by given benchmark programs, and challenge to develop their own way to reduce number of clock cycles to execute them. Some students may improve software (e.g., object code scheduling, utilization of delayed branch,)



Step 1 + 5 stage, 1 CPI pipeline.
+ Separete memory devices and buses are provided for instruction and data to avoid structural hazard.
+ No forwarding circuit.
+ No Interlock mechanism to avoid data and control hazards.
+ Verification of the functionality is performed by programs containing sufficient number of NOPs.

Step 2 + Observe the behavior of the processor when a program without NOPs are given.
+ Understand data and control hazard.

Step 3 + Every student is requested to develop their own way to reduce number of clocks to execute given benchmark programs.

Step 4 + Presentation and concluding remarks.

```
LOOP:   ADD     R0, R1
        NOP
        NOP
        NOP
        NOP
        SUB     R1, 1
        NOP
        NOP
        NOP
        NOP
        BNZ     LOOP
        NOP
        NOP
        NOP
        NOP
```

```
LOOP:   ADD     R0, R1
        SUB     R1, 1
        BNZ     LOOP
```

Fig. 3. Approach for Understanding Pipelined RISC Processor

and others may improve hardware (e.g., forwarding, branch prediction.)

Step 4. Make presentation, and concluding remarks are given.

To provide separate memory devices and buses for instruction and data, on-board memory and I/O devices of the ASAver.1 board are used as data memory, and an additional memory board is attached as an instruction memory. This daughter board is connected to the 32 auxiliary through holes on the ASAver.1 board (Fig. 4.)

This approach is expected to have following merits:

- Given from teachers is a very simple model of a pipelined RISC processor, which enables students to understand it easily and clearly. In addition, difficulties of pipelining are discovered at Step 2 by students themselves, rather than taught by teachers, using real machine developed at Step 1.
- Development of processor in Step 1 is as easy as development of a non-pipelined RISC processor, because the only difference is the number of hardware resources to avoid structural hazard. Easiness of development and easiness of running application programs (just by inserting `NOP` instructions) let students accomplish easily.
- The goal of Step 3 is clear, and there exist various methods and options to get to the goal. Every student can apply his/her preferred method (even his/her major interest is not hardware design.) They might even find the importance of HW/SW co-design by themselves.
- For those students who try hardware improvement at Step 3, migration from Step-1-processor to Step-3-processor is not very difficult nor laborious, because the framework of the processor does not change drastically. Even if this step cannot be finished com-

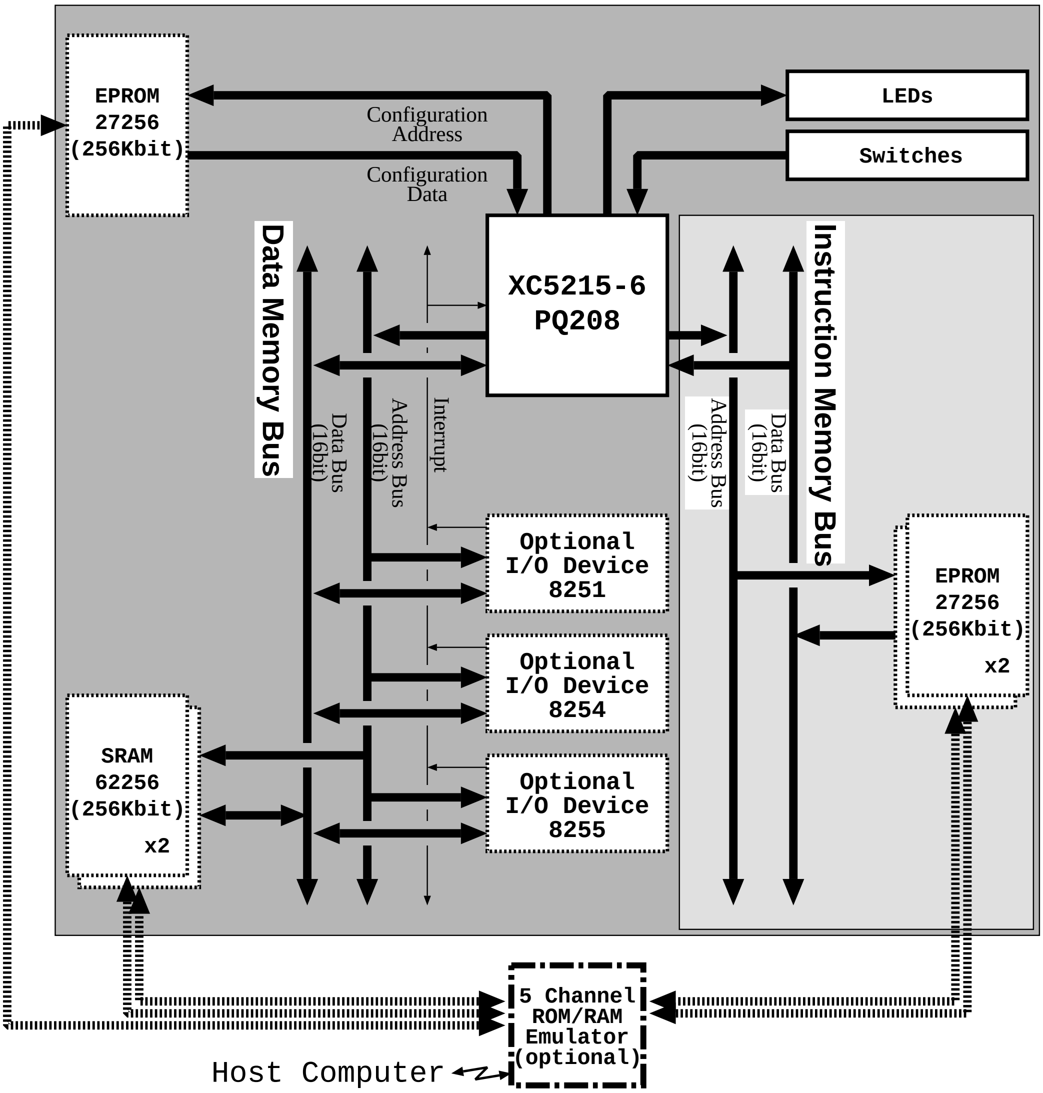


Fig. 4. Block Diagram of ASAver.1 Board for the Experiment (shaded portion in below right shows the daughter board)

pletely within limited time for the course, it is expected that there is a sufficient educational effect gained at Step 2.

- In Step 4, it is expected that most major topics on pipelined RISC processors appear in the students' presentation. Step 4 is expected to have a great educational effect, because they have just finished the experiment and hence they have great interest on pipelining.

Some difficult issues are carefully excluded from the proposed approach, including structural hazard and interrupt. These issues should be covered by lectures.

IV. Feasibility Study

To evaluate the feasibility of the proposed approach for undergraduate students, I have implemented three prototypes of 16-bit pipelined RISC processors on the ASAver.1 board. These processors, referred to as ASAP-0/f0, ASAP-0/f1, and ASAP-0/f2, has eight 16-bit general purpose registers (`gr`,) a 16-bit program counter (`pc`,) and a zero flag (`z`,) and has 10 essential instructions. Instruction set summary and data path diagram are shown in Table I and Fig. 5, respectively.

ASAP-0/f0 does not have any forwarding path; this implementation corresponds to Step 1 of the proposed approach. On the other hand, ASAP-0/f1 and ASAP-0/f2 have simple forwarding paths which are specified by dash lines in Fig. 5. Control hazard is not considered in

TABLE I

Instruction Set Summary of ASAP-0

Mnemonic	(MSB) Instruction Code (LSB)				Verilog-HDL Pseudocode

LD Ra, d(Rb)	00	Ra	Rb	d	// LoaD from memory
					gr[Ra] <= mem[gr[Rb]+d];
					pc <= pc+1;

ST Ra, d(Rb)	01	Ra	Rb	d	// STore to memory
					mem[gr[Rb]+d] <= gr[Ra];
					pc <= pc+1;

LI Rb, d	10	001	Rb	d	// LoaD Immediate
					gr[Rb] <= d;
					pc <= pc+1;

LHI Rb, d	10	011	Rb	d	// Load High Immediate
					gr[Rb] <= d<<8 | gr[Rb]&255;
					pc <= pc+1;

B d	10	111	000	d	// Branch pc <= pc+1+d;
BNE d	10	111	100	d	// Branch Not Equal pc <= (!z) ? pc+1+d : pc+1;

ADD Rd, Rs	11	Rs	Rd	0000 0000	// ADD arithmetic
					gr[Rd] <= gr[Rd]+gr[Rs];
					z <= ~|(gr[Rd]+gr[Rs]);
					pc <= pc+1;

NOR Rd, Rs	11	Rs	Rd	0010 0000	// NOR bitwise
					gr[Rd] <= ~(gr[Rd]|gr[Rs]);
					z <= ~|(~(gr[Rd]|gr[Rs]));
					pc <= pc+1;

NOP	11	111	111	1111 1110	// No OPeration pc <= pc+1;

HLT	11	111	111	1111 1111	// HaLT
					run <= 0;
					pc <= pc+1;

these prototypes.

ASAP-0/f0, ASAP-0/f1, and ASAP-0/f2 were described in Verilog-HDL (Table II.) Every description was compiled into schematic by Cadence Synergy 2.2, and implemented into a target FPGA (XC4013-6 PQ208[1]) by Xilinx Xact 5.2.0 on a Sun SPARCstation 20 workstation with 64 MB main memory. At last, the designed pipelined microprocessors implemented on the ASAver.1 board successfully executed an application program which plays music using the on-board interval timer device. I could easily prepare the program by just inserting sufficient amount of NOP instructions to the assembly listing.

Usage of an XC4013-6 PQ208 device is summerized in Table III. There are still a lot of unoccupied resources within an XC4013 device and, in fact, the entire circuitry of ASAP-0/f0 was successfully implemented even within an XC4006 device, which is a so-called 6,000-gate-class device [9], having just 256 CLBs (i.e., 512 4-input function generators and 512 flip flops are available.) This means that sufficient freedom for improvement of the processor is left for students, because the ASAver.1 board is designed to have a so-called 15,000-gate-class FPGA device XC5215 which has 1,936 4-input function generators and 1,936 flip flops [8].

Running time of Synergy and Xact are summarized in Table IV. Running time of Xact (especially, running time of routing step) sharply increases as CLB usage exceeds about 70%. It is expected that running time will be greatly reduced when we use a little larger target device, namely, XC5215.

V. Education Course of Our Department

Laboratories and exercises closely related to computer hardware in the undergraduate course of our department is listed as follows:

Computer Engineering Lab. 1 (for 2nd grade students) Experiments on digital circuits (computer hardware introductory experiments using the KUE-CHIP2 board, and basic logic design experiments using TTL devices) and analog circuits are performed.

[1] I had to use XC4000 series device for this feasibility study, because Synergy of newer version that supports XC5200 series devices are not yet installed in our workstation. I did not use the special mode to use a 4-input look-up table as a 16-bit SRAM which is available in XC4000 devices [9], because it is not available in XC5200 series devices [8].

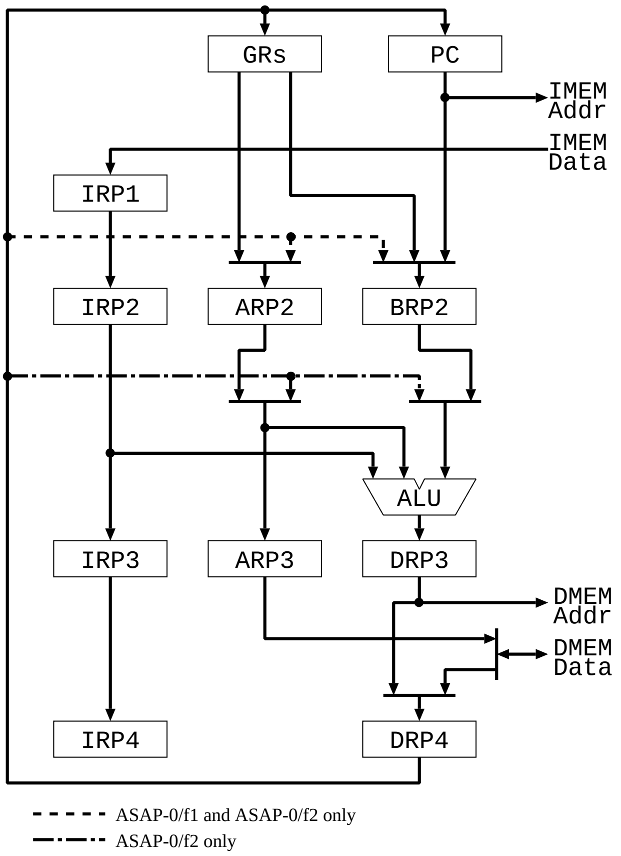


Fig. 5. Data Path Diagram of ASAP-0

TABLE II
Summary of Verilog-HDL Description

	ASAP-0/f0	ASAP-0/f1	ASAP-0/f2
Top Level Module	137 lines	150 lines	159 lines
Phase Signal Gen.	62 lines	62 lines	62 lines
ALU	25 lines	25 lines	25 lines
PC	21 lines	21 lines	21 lines
GR	65 lines	65 lines	65 lines
Total	310 lines	323 lines	332 lines

TABLE III
Usage of an XC4013-6 PQ208 device

	ASAP-0/f0	ASAP-0/f1	ASAP-0/f2
Occupied CLBs	$\frac{374}{576}$ =64%	$\frac{399}{576}$ =69%	$\frac{425}{576}$ =73%
F and G Function Generators	$\frac{281}{1,152}$=24%	$\frac{286}{1,152}$=24%	$\frac{324}{1,152}$=28%
H Function Generators	$\frac{73}{576}$ =12%	$\frac{75}{576}$ =13%	$\frac{77}{576}$ =13%
CLB Flip Flops	$\frac{302}{1,152}$=26%	$\frac{302}{1,152}$=26%	$\frac{302}{1,152}$=26%
3-State Buffers	$\frac{272}{1,248}$=21%	$\frac{304}{1,248}$=24%	$\frac{304}{1,248}$=24%

TABLE IV
Running time of CAD tools on SPARCstation 20

	ASAP-0/f0	ASAP-0/f1	ASAP-0/f2
Synergy	37 min.	38 min.	38 min.
Xact	62 min.	72 min.	171 min.
Total	99 min.	110 min.	209 min.

Logic Simulation (for 3rd grade students) Design and simulation using Verilog-HDL language are performed. The logic simulator Cadence Verilog-XL and the synthesizer Cadence Synergy on 60 Sun SPARCstation 5 workstations are available.

Computer Engineering Lab. 3 (for 3rd grade students) Design and implementation of a basic 8-bit microprocessor is performed using an FPGA device Xilinx XC4010-PG191. Yet another laboratory room is used, where they can use 30 SPARCstation 5 workstations with Xilinx Xact together with Verilog-XL and Synergy, and 15 logic analyzers.

Computer Engineering Lab. 4 (for 3rd grade students) At first, ASAver.1 board is used as a ready-made one-board microcomputer[2] to perform experiments such as communication using RS232c, to make students familiar to the instruction set architecture and the usage of the board. Then, design and implementation of a 16-bit pipelined RISC microprocessor is performed, as described in this paper. The same laboratory room as the Lab. 3 is used.

During the above education, students are expected to gain not only practical technique of hardware design but also ability of developing a unique, very large hardware system.

VI. Conclusion

In this paper, an effective and practical education approach for understanding and developing pipelined RISC processor at laboratory for undergraduate students is proposed. Then, description of ASAver.1 board and its usage for the laboratory are presented. This paper also descrives the result of prototyping experiments, and it is shown that the proposed approach is feasible for the laboratory for undergraduate students, and that a 16-bit pipelined RISC microprocessor ASAP-0 can be implemented on an existing FPGA device. Experiment in the real laboratory will be shown at the conference.

The essential points of the proposed approach are summerized as follows :

[2] By giving EPROMs containing configuration data, designed by the teaching staff, which implements a non-pipelined RISC microprocessor on the FPGA device on the ASAver.1 board.

1. At the first step, modeling of pipelined RISC processor is simplified by avoiding structural hazard and by ignoring other hazards.

2. Students learn pipelining by themselves through incremental implementation and evaluation.

This approach seems more effective to understand data hazard and control hazard, rather than learning hazards at lectures and textbooks. Through this experiment, it is expected that students will gain not only the knowledge on pipelined RISC processor, but also their own way of understanding hardware, and their own methodology and interest on development of something new.

The ASAver.1 board is versatile and useful for laboratories, and are expected to be widely used for coherent computer enginnering education from introduction followed by hardware, software, and system.

Acknowledgment

The author would like to thank Associate Professor Hideharu Amano of Keio University and Associate Professor Shin-ichiro Mori of Kyoto University for offering valuable materials on the laboratory carrying out for their students. The author would like to thank Professor Toshinori Sueyoshi of Kyushu Institute of Technology for giving some valuable knowhow related to the KITE board. The author would like to thank Mr. Hiroyuki Kanbara of Advanced Software Technology and Mechatronics Research Institute of Kyoto (ASTEM RI) and Associate Professor Tetsuo Hironaka of Hiroshima City University for their valuable suggestions and comments on the ASAver.1 board. Thanks are also due to the staffs of the Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University, for their encouragements and supports. Among them, the author would like to thank Associate Professor Ryuichi Takahashi for his great efforts for introducing the CAD environment and for carrying out "Logic Simulation" and "Computer Engineering Lab. 3," both of which are indispensable for the author's study.

This research was supported in part by a grant from Special Research Fund of Hiroshima City University.

References

[1] J. L. Hennessy and D. A. Patterson : *"Computer architecture : a quantitative approach,"* Morgan Kaufmann Publishers, Inc., 1990.

[2] H. Kanbara and H. Yasuura : *"KUE-CHIP2 : A microprocessor for education of LSI design and computer hardware,"* Proc. Synthesis and System Integration of Mixed Technologies (SASIMI'95,) pp.233–240, 1995.

[3] T. Sueyoshi : *"Application of FPGA to education for computer science,"* Journal of Information Processing Society of Japan, vol.35, no.6, pp.519–529, 1994, in Japanese.

[4] K. Nishimura, T. Kudo, and H. Amano : *"Educational 16-bit microprocessor PICO-16,"* Proc. third Japanese FPGA/PLD design conference and exhibit, pp.589–595, 1995, in Japanese.

[5] T. Nakagawa, A. Yamaga, W. Nagaura, M. Iwaihara, K. Murakami, and H. Yasuura : *"Designing educational microprocessor QP-DLX with full synthesis,"* Proc. 2nd Asian Pacific Conf. Hardware Description Language (APCHDL'94,) pp. 143-146, 1994.

[6] K. Nakagaki, M. Ouchi, K. Inoue, B.O. Apduhan, M. Kuga, and T. Sueyoshi : *"Design and Implementation of the Educational Microprocessor DLX-FPGA Using VHDL,"* Proc. 2nd Asian Pacific Conf. Hardware Description Language (APCHDL'94,) pp. 147-150, 1994.

[7] S. Tomita and H. Nakashima : *"Computer Hardware,"* Shoko-Do, (Oct. 1995,) in Japanese.

[8] Xilinx, Inc. : *"Xilinx XC5200 logic cell array family technical data v3.0,"* Xilinx, Inc., 1995.

[9] Xilinx, Inc. : *"The programmable logic data book,"* Xilinx, Inc., 1994.