

Using FPGA for Computer Architecture/Organization Education

Yamin Li and Wanming Chu

Computer Architecture Laboratory
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu, 965-80 Japan

yamin@u-aizu.ac.jp, w-chu@u-aizu.ac.jp

Abstract

In this paper, we introduce hardware exercises for Computer Architecture/Organization Education at the University of Aizu, Japan. Particularly, we discuss a pipelined RISC processor design and implementation on Xilinx FPGA chip.

1. Courses for Computer Architecture / Organization Education

The courses concerning Computer Architecture/Organization Education [1] [2] for undergraduates at the University of Aizu include Computer Architecture, Computer Organization I, and Computer Organization II. The Computer Architecture and Computer Organization I focus on the issues of uniprocessor computer architecture and organization. The Computer Organization II focuses on the issues of parallel computer architecture and organization (array processors and multiprocessor systems).

2. Laboratories for Computer Architecture / Organization Education

The laboratories for these courses consist of software exercises and hardware exercises. In the software exercises, we ask undergraduates to write some applications with assembly language, to run them at architecture simulators, and to evaluate the architectures and the applications.

In the hardware exercises, we ask undergraduates to design a simple non-pipelined processor, a pipelined RISC processor, and a multiprocessor computer. All of the hardware exercises are performed at Cadence environment for

schematic capture and functional simulation. The pipelined processor design and the multiprocessor computer design are implemented on Xilinx FPGA chips and measured with Logic Analyzer.

The exercise courses are helpful to undergraduates not only to understand the principle of Computer Architecture/Organization and the operations of pipelined processor and multiprocessors, but also to master the design methodologies and the use of measuring instruments.

We will discuss the pipelined processor design and implementation on Xilinx FPGA chips in next section.

3. Laboratory Design with Cadence and Xilinx Tools

In the University of Aizu, there are more than sixty sets of exercise equipments for Computer Architecture/Organization Education. One set consists of a SUN workstation with installed Cadence/Xilinx design tools [3], an evaluation board with mounted Xilinx xc4006PC84 FPGA chip [4] [5], and a Logic Analyzer.

We have developed a RISC pipelined processor (Aizup) and implemented it on the Xilinx FPGA chip for the course of Computer Organization I [10]. The Aizup processor pipeline has four stages and deals with data dependency and control dependency. Some techniques for performance optimization, such as data forwarding, delay branch, and instruction reorganization, can be demonstrated in the design.

The contents of the exercise include design of datapath and control unit, circuit schematic, functional simulation, netlist generation, pin assignment, placement and routing, Intel mcs-format file generation, file download and FPGA chip program, and measurement of the chip. Because the

Table 1. Instruction set and pipeline operations

Instruction Format	Operations	IF	DC	EX	WR
NOP 0000 00 00	No Operation	IR←M[PC] PC←PC+1			
ADD 0001 RD, RS RD RS	RD←RD + RS Update Z	IR←M[PC] PC←PC+1	A←RD B←RS	C←A + B Z←((A + B)==0)	RD←C
SUB 0010 RD, RS RD RS	RD←RD - RS Update Z	IR←M[PC] PC←PC+1	A←RD B←RS	C←A - B Z←((A - B)==0)	RD←C
OR 0011 RD, RS RD RS	RD←RD or RS Update Z	IR←M[PC] PC←PC+1	A←RD B←RS	C←A or B Z←((A or B)==0)	RD←C
AND 0100 RD, RS RD RS	RD←RD and RS Update Z	IR←M[PC] PC←PC+1	A←RD B←RS	C←A and B Z←((A and B)==0)	RD←C
XOR 0101 RD, RS RD RS	RD←RD xor RS Update Z	IR←M[PC] PC←PC+1	A←RD B←RS	C←A xor B Z←((A xor B)==0)	RD←C
MOV 0110 RD, RS RD RS	RD←RS Update Z	IR←M[PC] PC←PC+1	A←RD B←RS	C←B Z←(B==0)	RD←C
LD 0111 RD, RS RD RS	RD←M[RS]	IR←M[PC] PC←PC+1	A←RD B←RS	C←M[B]	RD←C
ST 1000 RD, RS RD RS	M[RS]←RD	IR←M[PC] PC←PC+1	A←RD B←RS	M[B]←A	
ADDI 1001 RD, n RD n	RD←RD + 000000n Update Z	IR←M[PC] PC←PC+1	A←RD B←000000n	C←A + B Z←((A + B)==0)	RD←C
SUBI 1010 RD, n RD n	RD←RD - 000000n Update Z	IR←M[PC] PC←PC+1	A←RD B←000000n	C←A - B Z←((A - B)==0)	RD←C
SR0L 1011 N N	R0←R0 or 0000N Update Z	IR←M[PC] PC←PC+1	A←R0 B←0000N	C←A or B Z←((A or B)==0)	R0←C
SR0H 1100 N N	R0←N0000 Update Z	IR←M[PC] PC←PC+1	A←R0 B←N0000	C←B Z←(B==0)	R0←C
BZ 1101 N N	if (Z) PC←PC+(s)N	IR←M[PC] PC←PC+1	if (Z) PC←PC+(s)N		
BNZ 1110 N N	if (!Z) PC←PC+(s)N	IR←M[PC] PC←PC+1	if (!Z) PC←PC+(s)N		
BRA 1111 N N	PC←PC+(s)N	IR←M[PC] PC←PC+1	PC←PC+(s)N		

undergraduates have mastered the use of Cadence in Logic Circuit Design and Computer Architecture courses, they can finish the exercise within one semester, say 45 hours.

3.1. Aizup Architecture

In order to be able to implement the pipelined processor associated with instruction memory and data memory on a single Xilinx xc4006 FPGA chip, we select the 8 bits as the width of instructions and data. Most of the instructions have two operands, one is a register operand (RD) and the other is a register operand (RS) or an immediate. The operation result is written back to RD. The four pipeline stages are IF (Instruction Fetch), DC (DeCode and operand fetch), EX

(EXecution or memory access), and WB (Write Back).

Table 1 lists the instruction operation codes, the operations in each stage, and the immediate extensions. Some registers are needed for the pipeline operations. They are PC (Program Counter), IR (Instruction Register), A/B (holding two source operands), and C (holding the result of operations). There is also a single bit register, Z, for storing the zero tag, that will be evaluated by conditional branch instructions.

3.2. Aizup Organization

Figure 1 shows the organization of Aizup. The major cells include ALU, register file, adders, instruction memory,

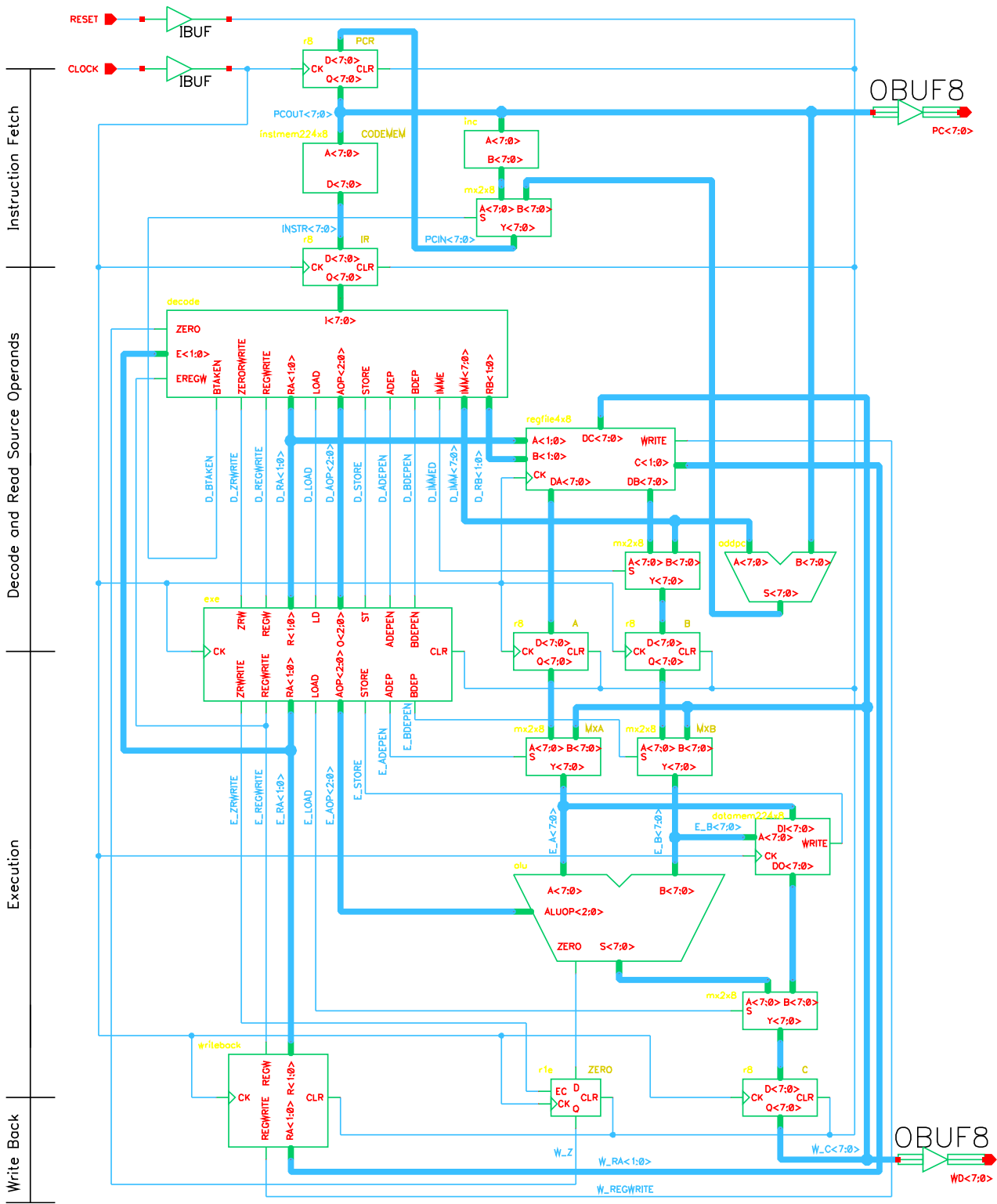


Figure 1. The top schematic of Aizup

data memory, decoder, pipeline registers, and multiplexors. The four pipeline stages are marked in the left side of the figure.

In Aizup, the register-to-register instructions perform $RD \leftarrow RD \text{ op } RS$, and two bits are used for addressing the register file. It means a 4×8 bits register file with two read ports and a write port is needed. It can be implemented with four 8-bit registers plus a pair of 4-to-1 multiplexors each 8 bits wide for read ports and a 2-to-4 decoder for write control.

For the data dependency, we designed a special data path to solve this problem. A dependent detection block can detect the dependencies (ADEPEN and BDEPEN). When a data dependency is detected, the source data for ALU operation is passed from C via multiplexor, instead of from A or B.

The dependency detection takes place in EX stage. In our processor design, we move it to DC stage, and use pipeline registers to transfer to EX stage. There are good points of the movement. First, doing the detection in DC stage will save the gates because some common logic can be shared with other decode circuits. Second, the time required by EX stage will be shortened because the ADEPEN and BDEPEN are available immediately at the beginning of EX stage.

As for the control dependency, we adopt the delay branch method. In our processor model, the branch target address is evaluated in DC stage. It means that one delay cycle is introduced and an additional adder is needed for address evaluation. We can use this method to demonstrate the optimization by reorganizing the instruction codes.

3.3. Aizup Control Unit

The control unit generates all control signals for controlling datapath. We summarize the control signals in Aizup as following.

1. IF stage

- BTAKEN (Branch Taken). If branch is taken, BTAKEN should be high to select the branch target address for instruction fetching in the next clock cycle.

2. DC stage

- AA<1:0> (A Address). It indicates the register number of RD. RD is the destination register but it is also the source 1 register ($RD \leftarrow RD \text{ op } RS$). Most of the instructions put the number of RD in

a fixed position in the instruction format, but the SROL and SROH instructions are special cases. The number of RD is always 0 for these instructions.

- AB<1:0> (B Address). It indicates the register number of RS. RS is the source 2 register. Most of the instructions put the number of RS in a fixed position in the instruction format, but some instructions put immediate in that position.
- IMMES (Immediate Selection). When the source 2 operand is an immediate, IMMES should be high for selecting the immediate, not register operand.
- IMM<7:0> (Immediate). It is an 8-bit immediate. Based on instructions, it can be generated with different extension methods. The IMM<7:0> is used not only for the source 2 operand of ALU operations, but also for the calculation of branch target address. Generally speaking, the IMM<7:0> does not belong to control unit, it belongs to datapath.

3. EX stage

- AOP<2:0> (ALU Operation Control). It can be generated easily based on instructions.
- ADEPEN (A Dependent). It is the selection signal for ALU operand A. When it is high, the forwarding data of register C is selected. Otherwise, the data of register A is selected.
- BDEPEN (B Dependent). It is the selection signal for ALU operand B. When it is high, the forwarding data of register C is selected. Otherwise, the data of register B is selected.
- STORE (Store). It is the memory-write control signal. When it is high, a memory-write happens.
- LOAD (Load). When LOAD is high, the register C will be written with the loaded data from data memory.
- ZRWRITE (Zero Register Write). When ZRWRITE is high, zero tag register will be updated.

4. WB stage

- AA<1:0> (A Address). It is the same signal as AA<1:0> of DC stage. Here it is the number of destination register only.
- REGWRITE (Register Write). It is the register-write control signal. When it is high, the data of register C will be written into register file.

All of the control signals are generated at DC stage. But some of them are used in EX and WB stages. To cope with this, the pipeline registers are used to put them to the corresponding stages.

The processor circuits are designed with Xilinx XC4000 library. The functional simulation is done at Cadence environment. By following Xilinx Design Flow, the bitstream file is generated and translated to Intel mcs-format. The translated file is transferred to evaluation board via RS-232C. After we program the xc4006PC84 FPGA chip, the Aizup can be measured with Logic Analyzer.

4. Discussion and Future Work

The University of Aizu attaches importance to exercise classes for most courses, especially for the courses of Computer Architecture/Organization Education. The time ratio of lectures and exercises for the courses of Logic Circuit Design, Computer Architecture, Computer Organization I, and Computer Organization II is 1:2, i.e., one unit for lecture and two units for exercise.

Most of the undergraduates have shown their interest in the hardware exercise classes and the use of Cadence/Xilinx design tools. Some undergraduates have joined research projects and they can design and simulate the real circuitry with various libraries.

The FPGA has shown its flexibility in rapid prototyping. We can implement our design on the chip and verify it in hours or a few days. And one more important feature of FPGA is the re-programmability. This feature is more suitable for education purpose. We can implement and verify various designs on the same chip.

The Aizup model that we designed and implemented does not contain any floating-point instruction. We would like to put the floating-point unit in the new version of Aizup. We have designed a Multiple-Threaded Multiple-Pipelined processor [8] [9] including floating-point unit using Toshiba TC180/183E/C ASIC library [6] [7]. The floating-point unit can perform addition, subtraction, multiplication, division, square root, and conversion between integer and floating-point numbers. We are now investigating the possibility of implementation of the floating-point unit on FPGA chips for advanced exercise course.

We are also investigating the use of external memory modules so that we can demonstrate the external memory access and instruction/data cache operations. The new Aizup will include integer unit, floating-point unit, instruction cache, data cache (with different structures and replacement algorithms), and interface to system bus. According to the experience on design and implementation of Aizup, we

believe that the circuitry of new Aizup will require more than one Xilinx xc4006PC84 FPGA chip.

In a new evaluation board design, multiple FPGA chips will be mounted on the board that is not only for the new Aizup design but also for multiprocessor design for Computer Organization II course. In this exercise course, we will demonstrate multiprocessor computer. The exercise will concern bus arbitration, cache coherence, and interconnection networks.

References

- [1] J. Hennessy and D. Patterson, *Computer Architecture, A Quantitative Approach*, Morgan Kaufmann Publishers, Inc., Second Edition, 1996.
- [2] D. Patterson and J. Hennessy, *Computer Organization & Design: The Hardware/Software Interface*, Morgan Kaufmann Publishers, Inc., 1994.
- [3] XACT Development System - XACT User Guide, Xilinx, April, 1994.
- [4] *The Programmable Logic Data Book*. Xilinx, 1994.
- [5] XACT Development System - XACT Libraries Guide, Xilinx, April, 1994.
- [6] Toshiba TC180C Series Cell Library, Vol. 1, Toshiba, 1993.
- [7] Toshiba TC180C/E*TC183C/E Series Micro-cell Library, Toshiba, 1994.
- [8] Y. Li and W. Chu, "Design and Performance Analysis of A Multiple-threaded Multiple-pipelined Architecture," *Proc. of the Second International Conference on High Performance Computing*, New Delhi, India, December 1995.
- [9] Y. Li and W. Chu, "A New Non-Restoring Square Root Algorithm and Its VLSI Implementations," 1996, to be submitted.
- [10] Y. Li and W. Chu, "Aizup - A Pipelined Processor Design and Implementation on XILINX FPGA Chip," *The Fourth Annual IEEE Symposium on FPGAs for Custom Computing Machines (FCCM'96)*, Napa, California, April, 1996.