TinyCSE: Tiny Computer System for Education

Ryosuke Nakamura, Yasuaki Ito, and Koji Nakano
Department of Information Engineering
Hiroshima University
Kagamiyama 1-4-1, Higashi Hiroshima, 739-8527 Japan

***Abstract*—TinyCPU is a small processor that can be implemented in various FPGAs that can be used for education and development of small embedded system. TinyCPU is so small that it is designed using Verilog HDL and the size of source code is only 427 lines. However, it does not support interrupts and peripheral controllers. The main contribution of this paper is to present TinyCSE (Tiny Computer System for Education), an extension of TinyCPU supporting interrupts and peripheral controllers. TinyCSE has controllers for external devices including keyboard, mouse, serial communication, switch, and timer. It also supports hardware interrupts from these external devices. Quite surprisingly, the code sizes of the CPU with interrupt controller and the device controllers are 515 lines and is 1339 lines in Verilog HDL, respectively. Our processor is portable and easy to understand and the function expansion is not difficult. As real-life applications, we have developed a time watch. This applications runs in 73MHz on the Xilinx Spartan-3AN family FPGA XC3S700AN using 832 out of 5888 slices (14.1%). Therefore, our tiny processing system benefits computer system education and small embedded system development.**

***Keywords*-Computer system, CPU, Verilog HDL, Education, Embedded system, Interrupt**

I. Introduction

Computer architecture is an indispensable subject for learning computer science and computer technology. To learn computer architecture, various CPU architectures have been developed as course materials [1], [2], [3]. However, it is important to learn not only CPU architecture but also whole computer system including peripheral devices.

In this study, we present TinyCSE (Tiny Computer System for Education), an extension of TinyCPU [1], [4], [5] and device controllers. TinyCPU is a small processor that can be implemented in various FPGAs. TinyCSE supports hardware interrupts and peripheral controllers that are not supported in TinyCPU. In TinyCSE, we added hardware interrupts to TinyCPU. Also, we designed controllers of external devices including keyboard, mouse, serial communication, switch, and timer. We focused on simplifying the designs and their descriptions of the external devices and the processor so that students easily can understand the whole architecture. Also, the proposed system is portable and easy to understand and the function expansion is not difficult. We confirmed the workings of our system using Spartan-3AN starter kit with a Spartan-3AN FPGA [6]. Using TinyCSE as a course material for education, students easily can learn the whole computer system including digital design, hardware description language, processor architecture, and peripheral devices.

This paper is organized as follows: Section II introduces the architecture of the original TinyCPU. In Section III, we show the extension of TinyCPU supporting hardware interrupts and peripheral devices. Section IV describes the implementation in the FPGA board and an application. Section V offers concluding remarks.

II. The Architecture of TinyCPU

TinyCPU is a simple, compact and portable processor which can be implemented in various FPGAs [1], [4], [5]. It is a pure stack architecture [7] and does not have an accumulator or a register set. Instead, it has an operation stack, which is used for all operations including store/load operations and arithmetic and logic operations. TinyCPU has seven components including state machine, 12-bit program counter, 16-bit output buffer, 16-bit ALU, 16-bit stack, and 16-bit data and 12-bit address memory. It also uses 16-bit data bus and 12-bit address bus. Every instruction of TinyCPU is a 16-bit word. TinyCPU supports 9 control instructions and 19 instructions for arithmetic and logic operations.

TinyCPU is designed using Verilog HDL. The Verilog HDL code is written as simple as possible. It consists of only 427 lines and can be synthesized for various FPGAs. Also, cross assembler and cross compiler for TinyCPU are available. However, TinyCPU does not support external devices and hardware interrupts from them. In this work, therefore, we propose an extension of TinyCPU to support hardware interrupts and peripheral controllers.

III. The Architecture of TinyCSE

This section shows our proposed TinyCSE (Tiny Computer System for Education), an extension of TinyCPU supporting hardware interrupts and peripheral controllers. The extended processor can handle hardware interrupts from external devices, including keyboard, mouse, serial communication, switch, and timer.

Figure 1 illustrates a block diagram of the architecture of our proposed CPU. The components and connections with solid lines show those of the original TinyCPU, and those with the broken lines indicate a part of the extension of the processor. The processor has 8 components including state

machine `state`, interrupt controller, 12-bit return register `rr`, 12-bit program counter `pc`, 16-bit ALU `alu`, 16-bit stack `stack`, 16-bit data and 12-bit address memory `ram`, and 16-bit data and 12-bit address I/O space `io`. It also uses 16-bit data bus `dbus` and 12-bit address bus `abus`. The inputs and outputs of the external devices are addressable by the I/O space. Our processor can communicate with external devices by specifying the address.

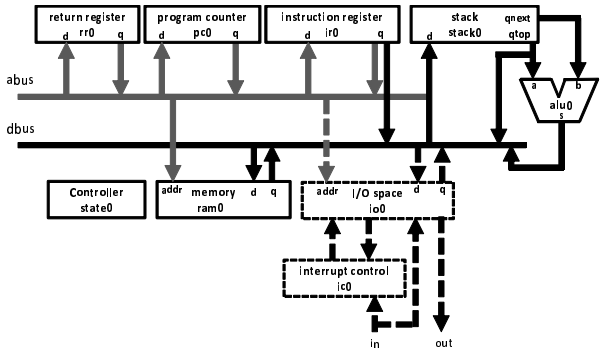

Figure 1. A block diagram of the proposed processor, showing internal architecture, bus connectivity and basic functional blocks. Solid lines show the components and connections of the original TinyCPU. Broken lines show those of the extension in this study.

A. I/O space

The original TinyCPU only supports switches and LEDs as the I/O. To support multiple peripheral devices, we extend the I/O space as port mapped I/O whose space is separated from address space. Peripheral devices are assigned to one or more ports in the I/O space. The I/O space module defines the assignment and connects the processor and peripheral devices.

B. Interrupt controller

The interrupt controller provides a software interface to the interrupt system. In our system, the controller supports hardware interrupts given by external devices. To make the controller simple, we designed it as follows.

The interrupt controller consists of an interrupt register and the interrupt signal `intr`. Peripheral devices are connected to the controller. The interrupt register indicates which device interrupts and the `intr` signal is a signal for starting the interrupt. When an interrupt occurs, the interrupt controller asserts `intr`. The controller stores which device interrupts to the interrupt register. In our interrupt process, to make the architecture simple, when an interrupt occurs, the other interrupts are ignored. In other words, multiple interrupts are not supported.

C. Extension of the state machine and instructions

To support the above I/O space and interrupt controller, we modified the state machine of the original TinyCPU. Figure 2 illustrates a state transition diagram of our proposed processor. When an interrupt occurs, the state of processor transitions into interrupt states `INSTA` and `INSTB` after the executing instruction is completed. In the interrupt states, the value of program counter is stored to the return register and the program branches into the interrupt routine by loading the address of the routine into the program counter. In the interrupt routine, interrupt process based on the device is executed.

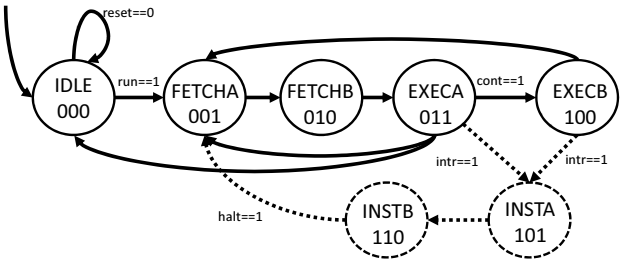

Figure 2. Our processor state transition diagram. Solid lines show the states of the original TinyCPU. Broken lines show those of the extension for the interrupt process in this study.

Also, we extended two existing instructions `IN` and `OUT`, and added two new instructions `RETURN` and `CALL`. The details of these instructions are listed, as follows.

- IN: push the value of the specified address in I/O space onto the stack top.
- OUT: pop the value of the stack top to the specified address in I/O space.
- RETURN: branch to the address stored in the return register and the interrupt signal is negated.
- CALL: store the value of the program counter in the return register and branch to the address the operand specifies.

D. External device controllers

We designed timer, VGA display, keyboard, mouse, and serial communication controllers as external device controllers. These controllers are connected to the processor and can be operated by the processor via the I/O space and the interrupt controller as shown in Figure 1. For the page limitation, the details of these controllers are omitted.

IV. Implementation in the FPGA board

We have implemented our processor in the FPGA board, Spartan-3AN starter kits (Figure 3). The Spartan-3AN starter kits FPGA boards are equipped with Spartan-3AN family FPGA XC3S700AN. The board has various peripheral devices: slide switch, push button switch, rotary switch, LED, and LCD. It also has VGA, PS/2, and serial ports as I/O ports. We have implemented our processor in the FPGA board.

Our processor and device controllers are written in Verilog HDL. Table I summarizes our processing system. The

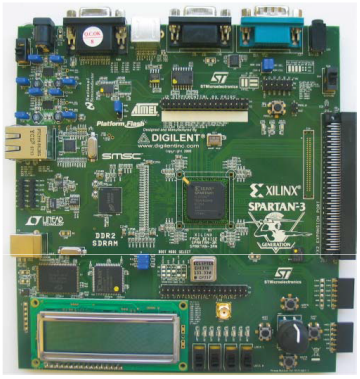

Figure 3. Spartan-3AN starter kit

code consists of 515 lines for the processor and 1339 lines for peripheral device controllers. We note that the code size of device controllers is quite large. This is because the font data in the VGA controller to display characters on the screen. Therefore, readers can find that the code size of each device is not so large.

Table I
OUR PROCESSING SYSTEM AND ITS CODE SIZE IN VERILOG HDL

	module or function	code size (lines)
Processor	definitions	58
	ALU	39
	counter	14
	state machine	27
	stack	34
	memory	27
	IO space	90
	interrupt controller	82
	top module	144
	total	515
Device controller	switch	19
	keyboard	102
	mouse	121
	serial communication	88
	timer	16
	LCD	159
	VGA (excluding font data)	88
	VGA (font data)	702
	top module	44
	total	1339
total		1854

We have used XST in ISE Foundation 13.1 for logic synthesis and analysis. According to the result, our system runs in 73MHz and uses 832 out of 5888 slices (14%), 1 out of 20 18-bit multiplier (5%), 4 out of 20 BRAMs (20%), and 42 out of 372 I/O pins (11%). Therefore, our system is simple and compact.

As real-life applications, we have developed a time watch (Figure 4). In this application, a value of a counter increases per second by the interruption from the timer controller. In addition, the button switches are used to control the behavior of the counter: start, stop, and reset counting. The value of the counter is displayed on the monitor using the VGA controller. This implementation of the real-life application result proves that our system can be used for developing a small embedded system.

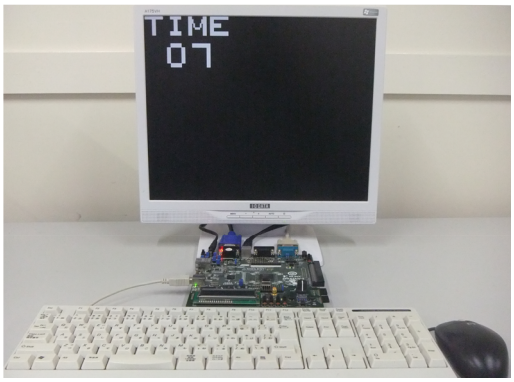

Figure 4. An example of the application (time watch)

V. CONCLUSION

We have presented TinyCSE that is an extension of TinyCPU supporting interrupts and controllers for peripheral devices. Our processor is intended to be used as a course material for computer architecture education. Our system is designed by Verilog HDL and the code consists of 515 lines for the processor and 1339 lines for controllers of peripheral devices. We have demonstrated our system using the Spartan-3AN starter kit. In the future, we plan to develop the operating system on this system.

REFERENCES

[1] K. Nakano and Y. Ito, "Processor, assembler, and compiler design education using an FPGA," in *Proc. of International Conference on Parallel and Distributed Systems*, December 2008, pp. 723–728.

[2] Y. Li and W. Chu, "Aizup – a pipelined processor design and implementation on XILINX FPGA chip," in *Proc. of IEEE Symposium on FPGAs for Custom Computing Machine*, April 1996, pp. 98–106.

[3] N. Fujieda, T. Miyoshi, and K. Kise, "A MIPS system simulator," in *Proc. of Workshop on Computer Architecture Education*, December 2009, pp. 32–39.

[4] K. Nakano, K. Kawakami, K. Shigemoto, Y. Kamada, and Y. Ito, "A tiny processing system for education and small embedded systems on the FPGAs," in *Proc. of Embedded Software Optimization*, December 2008, pp. 472–479.

[5] I. McLoughlin, *Computer Architecture: An Embedded Approach*. McGraw Hill Higher Education, 2011.

[6] *Spartan-3AN FPGA Family: Data Sheet*, Xilinx Inc., 2011.

[7] P. J. Koopman, Jr., *Stack Computers: the new wave*. Ellis Horwood, Ltd, 1989.