Research and Design of a simple CPU with Quartus II

Bo Liu*

College of Information Engineering, Chaohu University, Chaohu, China

*Corresponding author e-mail: 623321161@qq.com

Abstract. Since the architecture of a CPU is the core technology in the computer industry, in this paper, we aim to carry on a research on the architecture CPU by really design a simple CPU with Quartus II which is a hardware design tool produced by Altera company. The design procedure of this simple CPU includes the overall structure planning, Instruction set design, and CPU hardware design by Verilog HDL. Furthermore, In order to test the validity and correctness of this CPU, we designed an assembly language program which was compiled manually and loaded to the storage of the CPU, the result of the test shows that the whole procedure of the design is correct and can be implemented authentically.

1. Introduction
As the core technology of a computer, CPU design plays an import role in the computer manufacturing industry. However, the implementable procedure of the CPU design is not well known even by some computer professionals. With the help of Quartus II, design an executable CPU on your own is not impossible.

2. Specific methods and procedure of the design
Basically, no matter how simple a CPU we design, it has some necessary components and connections among these components. These components include instruction register (ir), control logic (control), arithmetic unit (ALU), and front register (A and B) of ALU, program storage (irom), program counter (PC), data storage (dram), address register (mar), accumulator (da), and output register (out), etc [1].

2.1. Research and design of instruction system
The instruction system is the focus of CPU design, on one hand, instruction system is based on the overall structure of the CPU, on the other hand, it also depends on the specific tasks and application scenarios. Make it simple, this CPU will be applied in the simple arithmetic tasks. Therefore, the instruction set can be determined according to the given tasks.

For our CPU, every instruction designed is composed by 16 bits, which is named machine code in professional term, and the high 5 bits are used to distinguish different functions of various instructions, the rest 11 bits are used to distinguish different operands. Table 1 presents the instructions we designed both in the type of machine code and the type of corresponding assembly code.

2.2. How to make the instructions work?
As we all know that the instructions cannot work without the corresponding logic control circuit, but how to design the logic control circuit? Generally speaking, Quartus II, which is a hardware design
software program produced by Altera company, is usually used to design the logic control circuit. And Verilog HDL, which is a popular hardware design language, allows people to design hardware in the way they write software code [1, 2].

Table 1. Instructions designed (machine code and the corresponding assembly code).

| number | machine code | Assembly code |
|--------|--------------|---------------|
| 1      | 00001        | Lda           |
| 2      | 00010        | Sdal          |
| 3      | 00011        | Sdah          |
| 4      | 00100        | Sub           |
| 5      | 00101        | Out           |
| 6      | 00110        | Jmp           |
| 7      | 00111        | Call          |
| 8      | 01000        | Str           |
| 9      | 01001        | Jn            |
| 10     | 01010        | Jz            |
| 11     | 01011        | Add           |
| 12     | 01100        | Ret           |
| 13     | 01101        | Mult          |
| 14     | 01110        | Divi          |
| 15     | 11111        | Stp           |

Therefore, with the help of Quartus II and Verilog HDL, we are able to design our own CPU, figure 1 shows part of the Verilog HDL code we wrote for this CPU, in this code, we borrowed the storage already designed by Quartus II [1, 3], and we can see every instruction works orderly according to the clock period.

Figure 1. Part of the CPU code designed by Verilog HDL.
3. Experimental design and testing

3.1. Design the testing program

It is very important to design a good testing experiment for the CPU designed. Therefore, an assembler code for Arithmetic operations is designed, in order to test the whole instructions designed above, this assembler code needs to use as much the whole instruction set as possible.

Table 2 shows the testing assembler code we designed. This assembler program is designed to figure a arithmetic operation (32 ×45), It uses 4 variables (c, a, b, and result) and 3 labels (begin, Loop, and exit), the program address is from 0, and the variables addresses are given randomly (The address of c is 1, a is 2, b is 3, and result is 4). Therefore, the machine code and the corresponding hexadecimal code can be figured out manually.

| Address | label | variable | Assembly code | machine code | Hexadecimal code |
|---------|-------|----------|---------------|--------------|------------------|
| 0       | begin:| sdal 0   | 0001000000000000 | 1000        |
| 1       | c     | str result | 0100000000000001 | 4004        |
| 2       | a     | sdal 1   | 0001000000000000 | 1001        |
| 3       | b     | str c    | 0100000000000010 | 4001        |
| 4       | result| sdal 32  | 0001000000100000 | 1020        |
| 5       |       | str a    | 0100000000000001 | 4002        |
| 6       |       | sdal 45  | 0001000000101101 | 102D        |
| 7       |       | str b    | 0100000000000000 | 4003        |
| 8       | Loop:| lda b    | 0001000000000001 | 0803        |
| 9       |       | Jz exit  | 0101000000000000 | 5010        |
| 10      |       | sub c    | 0010000000000000 | 2001        |
| 11      |       | str b    | 0100000000000001 | 4003        |
| 12      |       | lda result| 0000100000000000 | 0804        |
| 13      |       | add a    | 0101100000000000 | 5802        |
| 14      |       | str result| 0100000000000000 | 4004        |
| 15      |       | jmp Loop | 0011000000000000 | 3008        |
| 16      | exit:| out result| 0010100000000000 | 2804        |
| 17      |       | stp      | 1111100000000000 | F800        |

Table 2. Testing assembler code.

3.2. Test the CPU designed

In order to test the CPU designed, the testing assembler program must be run on the CPU, and the test result is also needs to be observed expeditiously. With the help of Quartus II, this test experiment can be conducted successfully. But, how to run the program with Quartus II?

First of all, the testing code (hexadecimal code) should be loaded to the storage cpu.mif which is a storage file made by Quartus II, figure 2 shows the storage file, and the hexadecimal codes are already loaded to this file.
Secondly, timing simulation is carried out and the result demonstrates the correct answer of $32 \times 45$, the answer is 1440 in decimal, and 05A0 in hexadecimal (see figure 3) which is obviously correct.

4. Conclusion
This paper has, through a specific design example and the testing experiment, presented the CPU design methods and finally demonstrates the validity of the whole design process.

Acknowledgments
This work was financially supported by University natural science research fund of Anhui province (KJ2018A0461).

References
[1] Yongjiang Jiang, Design and manufacture CPU and single chip microcomputer, M. Poster & telecom press, Beijing, 2014, pp. 46-66.
[2] Altera Company, Quartus prime introduction using verilog designs, 2016.
[3] J. Gray, “Building a RISC System in an FPGA: Part 1: Tools, Instruction Set, and Datapath; Part 2: Pipeline and Control Unit Design; Part 3: System-on-a-Chip Design”, Circuit Cellar Magazine, #116-118, March-May 2000.