Advances in Engineering Research, volume 123
2nd International Conference on Materials Science, Machinery and Energy Engineering (MSMEE 2017)

Design and Implementation of Parallel Pipeline Processor

Hongyu Fan[1,a)], Hui Zhang[1,b)], Nan Jiang[1,c)],and Dongwei Guo[1,*)]

[1]*School of Computer Science and Technology, Jilin University, Changchun 130012, China.*

[a)]13166833675@163.com
[b)] 15754310743@163.com
[c)] jiangnan123_ok@163.com
[*)] guodw@jlu.edu.cn.com

Abstract. In order to further improve the speed of instructions' execution on the basis of the current multi-cycle processor and pipeline processor which is commonly used by MIPS processors, we implemented a simple dual-pipeline processor. To show its superiority, we followed the implementation of this three processors, and then carried out the same test and analysis. The final result indicates that the execution speed of the pipeline processor is 2.837 times fast than the multi-cycle processor, while the dual-pipeline processor is running 32.8% faster than the normal pipeline.

Keywords: Design, Parallel Pipeline, Processor

INTRODUCTION

Reduced instruction set system is widely used in embedded system, but with the complication of demand, improve the speed of instruction execution is also increasingly important. This paper based on the MIPS architecture and use Modelsim software [1,2]to simulate the instructions' execution

Multi-cycle processors is typical in early period, although the control logic is relatively simple, only need to maintain a state machine and fewer control signals, it can't take full advantage of each component thus the time efficiency is low. Pipeline processor improve the utilization of components to a large extent, and it needs to take various data related problems and jump instructions into account, In order to ensure that this complex logic can be implemented correctly, more control signals and components are used. while with the rapid increase in demand, the time limit of pipeline processor is exposed, the only way to break through this bottleneck is to increase the frequency or use multiple lines in parallel**,** so we attempt to use dual-pipeline. Instruction-related issues become more complex in dual-pipeline, so after the design and unremitting efforts we successfully achieve a dual-pipeline processor, making the speed of instruction execution significantly improved.

IMPLEMENTATION PROCEDURE

To improve the internal parallel capabilities, MIPS architecture requires instruction and data storage separately, that is, using Harvard architecture. The process of implementation goes through three general stages: simple multi-cycle processor, pipeline processor and dual-pipeline processor.

The realization of the multi-cycle processor

Multi-cycle processor is the mainstream MIPS processor in early time, the instruction execution process is divided into five stages(Instruction Fetching, Instruction Decoding, Execution, Memory, Write Back)[3], as Fig. 1 shows.

FIGURE 1. The multi-cycle processor performs the illustration.

The process of execution for different instructions may be different, in order to determine which is the stage of the next cycle, it is necessary to maintain a state machine within its controller**.** Meanwhile, the processor is divided into several sub-modules to ensure functional modularity[4], which includes: fetch module, bit expander, arithmetic logic unit, controller, general-purposed register, memory. The program counter is used to make sure that the instructions are executed in a correct logical order. The control signals and the enable signals ensures that each instruction have the correct operating authority towards functional parts.

The realization of the pipeline processor

The design of the pipeline processor is based on the multi-cycle processor, and the execution of each instruction have no change. The differences are the additional data and control signals and several modules[5], through more complex and meticulous logic to improve the efficiency of components, Fig. 2 shows the basic process of the five-stage pipeline. There is intermediate stage between each of the two phases[6,7], so a total of four intermediate modules are added to deliver signals and data of the current instruction, preventing these values from being covered by the follow values. Fig. 3 shows the Function module distribution and cooperation.

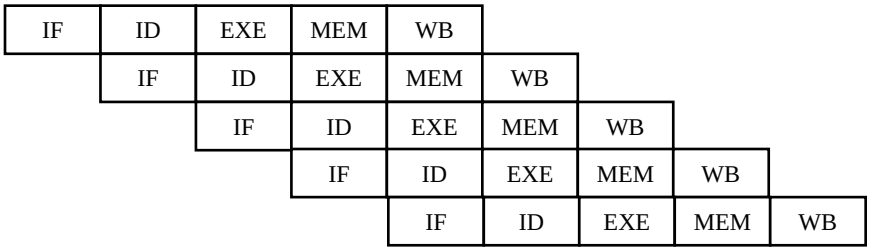

FIGURE 2. Five-stage pipeline instruction execution icon.

The jump prediction and hazard detection module are added. This two modules solve two kinds of read after write hazard and problems from jump instructions together:

(1) When the result of one instruction's execution cycle is entered as another instruction's source operand, the hazard detection module would give signals and supplies the result directly to the next instruction's to ensure that the next instruction can have the correct source operand input.

(2) When variable from memory is entered as another instruction's source operand, it is necessary to add the bubble delay execution to guarantee the correctness of result.

(3)IF one instruction have a jump operation, this two modules provide corresponding control signal to the pipeline to give up the results of two instructions behind it, while modifying the instruction counter to ensure that the next incoming instruction is the location after the jump.

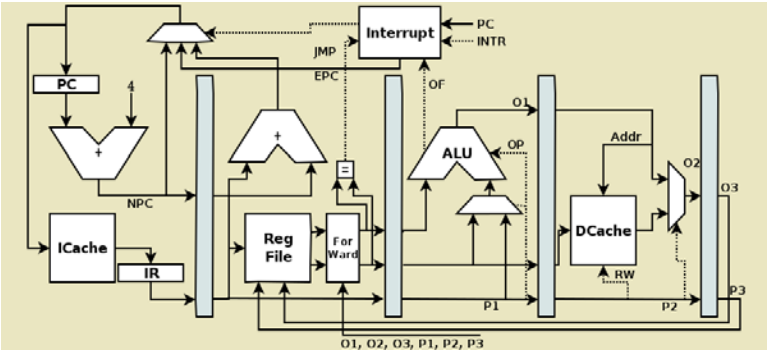

FIGURE 3. Multi - cycle pipeline module structure diagram.

The realization of the dual-pipeline processor

This is based on the pipeline processor and further improved from it, compared to the single pipeline, the structure of each pipeline does not change. The difference is the use of dual-pipeline structure[8], which can improve the speed of instruction execution. Among this two pipelines, one is a major assembly line, the other is a auxiliary pipeline with lower priorities.

There's a pre-fetch module, used to check hazards and conflicts between two instructions:

(1) There is no conflicts or hazards, this two instructions can enter two pipelines at the same time respectively, the process will not interfere with each other.

(2) If conflicts exist, the current instruction enter the main assembly line, meanwhile auxiliary pipeline add a empty command to keep balance between two pipelines.

A more complex problem is that hazards exists between the two pipelines, pre-fetch module need to fetch and check the saved four previous adjacent instructions to ensure a reasonable order of distribution. In addition, to reduce the possibility of emergency and complexity of related problems, all kinds of jump instructions and those who need to access memory [9]are entrusted to the main line.

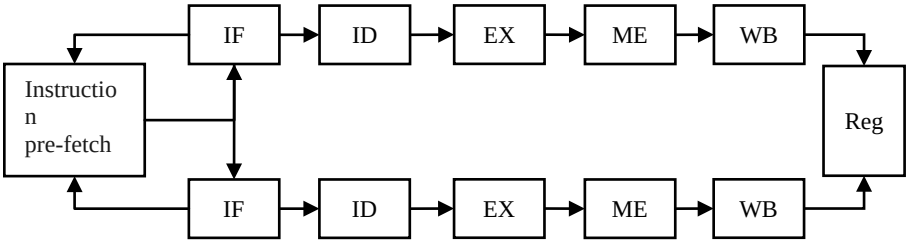

FIGURE 4. Double pipeline instruction execution diagram.

TEST RESULTS AND ANALYSIS

Instruction test

MIPS instructions are 32-bit word and expressed in hexadecimal number., we carry out a test towards 50 kinds of instructions. According to the actual situation in the distribution of various types of instructions, we use MARS software to generate 1420 instructions, covering the various MIPS instructions, arithmetic and logic operation class instructions account for about 80% among them. These instructions were then tested in multi-cycle processor, pipeline processor, and dual- pipeline processor, and the results were recorded and analyzed.

Test results are and comparison

The two-column acceleration ratio in the table gives the relative effect of the first two processors.

TABLE 1. Three Processor Efficiency Comparison Tables.

Processor name	Number of cycles	Time/us	Speedup ratio	Speedup ratio
Multi-cycle processor	5680	227.2	1	
Pipeline processor	2002	80.1	2.837	1
Dual pipelined processor	1507	60.3	3.769	1.328

We use the number of cycles to measure the efficiency of three different processors. Table 1 are results after recording and calculating. After repeated debugs and tests, the numerical results are recorded[10], some of the simulation tests are shown in Fig. 5. Each row represents one processor, Number of cycles are came from the

execution of 1420 instructions, Time is actual cost spent by these cycles. According to the data, we can intuitively see that the speed of this three processors improves one by one, the upgrade is obvious, towards the same instruction set ,number of cycles from about 5700 down to 2000 cycles, and then down to about 1500 cycles. The table involves the concept of the Acceleration Ratio, the number of cycles in which the processor 1 executes these instructions are n_1 , and the counterpart of processor 2 is n_2 .so processor 2's Acceleration Ratio towards processor 1 can be calculated as below:

$$Acceleration\ \ Ratio = \frac{n_2}{n_1} \tag{1}$$

When comparing the multi-cycle processor and the single pipeline, it can be easily seen that the speed improvement is 2.837 times. Because of the data hazards among instructions and delay caused by the jump instructions, the process will produce some empty instructions which can slow down the pipeline, so this result is a apparent enhancement. When comparing the single pipeline processor and dual-pipeline processor, the Acceleration Ration is 1.328, because the use of dual-pipeline, the interaction between the instructions is more unpredictable, and the impact between the pipeline will be generated, so this improvement is also impressive and hard-won.

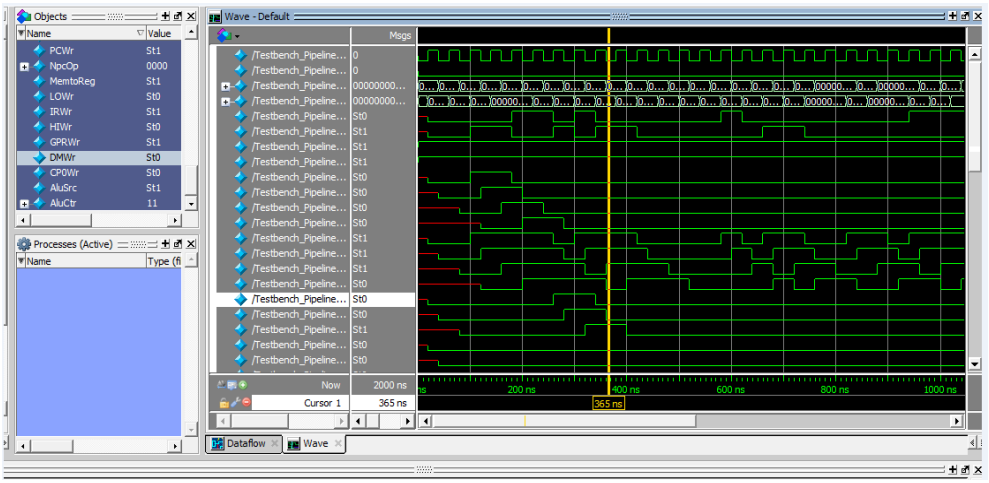

FIGURE 5. Part of the simulation test

CONCLUSION

No matter what kind of processor, the structure of coding is clear, but it costs a lot of time and efforts to debugs and tests. the difficulty lies in the hazard and forwarding modules, handling all kinds of situations in a comprehensive way is complicated, but all in all it's worthwhile. The dual-pipeline has improved efficiency significantly, but there's many aspects which can be improved in further research. Adding more control signals and intermediate registers to ensure the correctness of the results is a relatively successful approach to reduce the cost of time. However, the preservation and transmission will take more space and logical costs. Additionally, this plan is limited, the cooperation between modules and communications between pipelines remain promoted. The next step is to learn how to perform more functions with fewer components to achieve further increase at the design phase.

ACKNOWLEDGMENTS

The corresponding author of this paper is Professor Dongwei Guo*, thanks to Professor Guo's guidance and encouragement. Thanks to all of the participants because their perseverance is of vital importance and thanks for their contribution to the composing of this paper. Finally we are grateful to Undergraduate Training Program for Innovation and Entrepreneurship of Jilin University for its financial support(2016B53469).

* Mail Address: guodw@jlu.edu.cn

REFERENCES

1. The Verilog Language Reference Manual. 1995.
2. D. Sweetman, See MIPS Run, second ed, 2005, pp.1-28.
3. J. B. Peatman, Digital Hardware Design, McGraw-Hill, New York, 1980.
4. K. Hwang, Computer Arithmetic: Principles, Architecture and Design, Wiley, New York, 1978.
5. P. M. Kogge, The Architecture of Pipelined Computers, McGraw-Hill, New York, 1981.
6. L. M . kwiatkowski, C. Verhoef, Science of Computer Programming, 2015.
7. D. Hrris, S. Harris, Digital Design and Computer Architecture, second ed, 2013, pp.-622.
8. M.J. Flynn, Computer Architecture : Pipelined and Parallel Processor Design, Jones and Bartlett, Boston, MA. 1995.
9. H. G. Cragon, The Cache Memory book, Academic Press, San Diego, CA, 1998.
10. E. L. Lamie, Real-Time Embedded Multithreading Using ThreadX and MIPS, 2009.