MANTECH Publications
Journal of Embedded Systems & Its Applications
Volume 1 Issue 1, January-June 2016

Reduction in Processing Time Using Pipelining In 16 Bit Microprocessor

Tanushree Girkar[1], Sneha Nagar[2], Prof. H.R. Singh[3]
[1]Student, [2]Assistant Professor, [3]Professor
Department of Electronics & Communications
[1]Oriental College of Engineering, [2,3] Oriental University, Indore, M.P.
E-mail: *girkartanushree2@gmail.com[1]*

Abstract

Speed of processor is the biggest issue nowadays. We live in a world where we are moving fast, hence to make our work easy, processors are designed which are getting more and more complex. This paper is about how we can reduce the overall time required for processing the input, For this purpose, we require some sort of mechanism that reduces the number of clock cycles. To this end, pipelining and subpipelinig is used. Generally, bubble pushing is used in pipelining to make the processor more efficient. The instruction sets used by the processor are very simple. The modeling of processor is done using verilog language for digital systems. The simulator used is Xilinx model simulator. Complex models are designed by behavioral modeling. The type of processor used is RISC. The processor is operational for 16 data bit, but it can be modified for 32 bits.

Keywords: *Processor Pipelining, 16 bit Processor, Staged Pipelining, Pipelining Hazards*

INTRODUCTION

The first processor was designed by defense department. And the first commercially available processor was 4004 by Intel. It consisted of 2300 transistors. It was a 4 bit processor. Today's processors contain 6 billion transistors. The speed has improved up to 3.2 GHz and 64 bits of data. The heart of processor is CPU i.e. the central processing unit. The CPU contains Registers, Control unit and ALU. You have to add input and output devices to complete

the processor. All these together create a computing system. Now the question arises that how the processor connects to the external world. There are address buses, Interface units, Input and output ports. Then there is an Address decoder system attached so that processor does not confuse between address and data. The function of address decoder is to take address lines from processor and trigger different interface units. The language which is understood by processor is assembly language that is decoded into machine code in the form of bits which is a combination of 1's and 0's.

The reduced instruction set computer, or RISC, is a microprocessor CPU design philosophy that favors a smaller and simpler set of instructions that take about the same amount of time to execute. The most common RISC microprocessors are ARM, DEC Alpha, PA-RISC, SPARC, MIPS, and IBM's PowerPC. The idea of RISC came into picture when CISC became too complex to use. People move forward to use simpler set of instructions. The features of RISC are uniform instruction encoding, homogeneous register set, simpler addressing modes and simple data types. The first and foremost thing in our processor is that one should assume it is just like a blank box and one can use it anywhere they want. So, it can be used in any application, wherever one wants. The processor we have designed has some features important to complete our goal. The reason for using RISC in our design is to make it simple. The uP16 is a very basic 5-stage pipelined processor. It has 2048 bytes of 18-bit instruction memory, 2048 bytes of 16-bit data memory and an 8 x 16-bit register file.

INTERNAL STRUCTURE OF A PROCESSOR

The architecture of the designed processor is a 16 bit RISC processor. It has total 16 Registers divided in two pairs. Every single instruction is completed in 5 cycles. An external clock is used for timing and synchronization.

Instruction Set Architecture (ISA): The ISA of this processor consists of 16 instructions with a 4-bit fixed size operation code. The instruction words are 16-bits long. The instructions that we processed are namely arithmetic, logical, memory operations, conditional operations, jump operations.

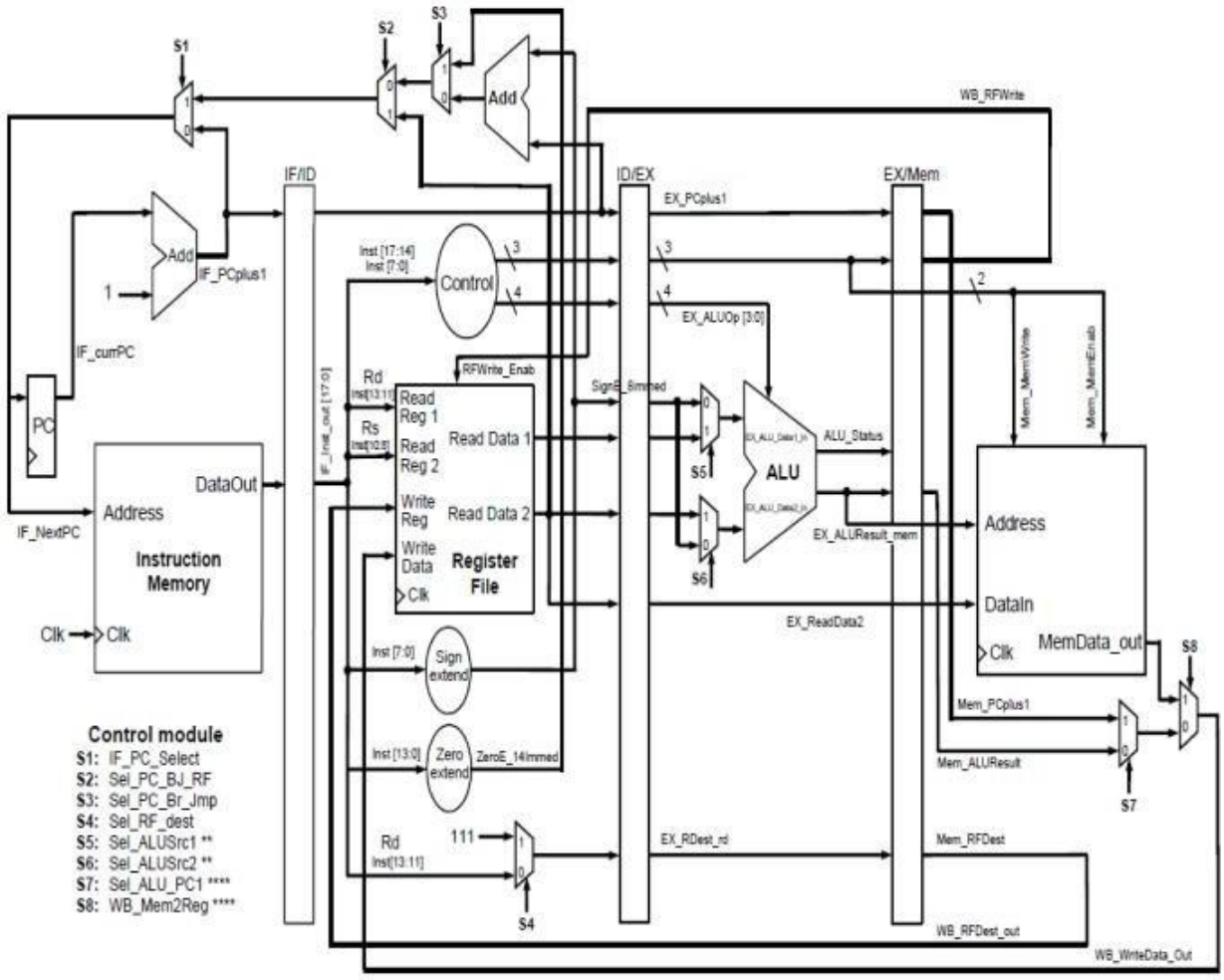

Fig. 1 microprocessor architecture

DESCRIPTION OF PROCESSOR

- ***Registers*** - There are total 8 registers in which R0, R6 and R7 are special purpose registers, while others are general purpose registers. R6 and R7 are stack pointers.
- ***Memory*** **-** This is the basic need of any processor. Two arrays of memory or RAM is used here (one for instruction memory (18bits wide as the FPGA will support 18 bits) and the other for data and stack memory (16 bit data)). Separate block RAM memory is used for both the instruction and data memory, configured as 1024x18 bit memory

and 1024x16 bit memory, respectively.

- Microprocessor Instructions: All uP16 instructions are 18-bit words stored in the instruction memory. There are only 2 different instruction formats (R-format and J-format).

PIPELINING

This concept comes in to picture after decades of 80's.If we analyze any non-pipelined architecture, and then we found some hazards in it. There are five stages for processing any bit. The stages are namely Instruction fetch, decoder, execution, memory access and write back. Now if any instruction is given to processor it has to pass through all stages to give an output. Therefore total no of clock cycles needed for16 bit instruction will be 16.Hence,we have to wait for a long time to get output and processor is called as slow processor. To speed up the work pipeline concept came in to picture. Now, the scenario will change.

A pipelined architecture is little more complex than non-pipelined. As we know that complexity always increase when design becomes small and speed becomes a major issue. We have to add some mechanism here, like some registers. Now in case of pipelining, in the first cycle t1 fetching is done and then its data is stored in register R1, thus the fetch register is free to fetch new bit. The data bit of R1 is now decoded by decoder in next cycle t2 and at the same time next data bit is fetched by fetch register. Similary, this bit of decoder is stored in register R2. This procedure is followed till write back. So total no of registers required are 5. Hence the total no. of cycles is reduced. If 32 bits of instructions are there, then total no. of cycles required will be 11. Hence we get fast output in this case.

PIPELINING HAZARDS

The problems of pipelining here are termed as hazards. This limits the performance of the system which we want. They are caused by the organization of pipelining. There are three types of hazards, data hazards, structural hazards control hazards. The structural hazard is cause by lack of resources. Because of these hazards, the normal flow of pipeline breaks and the result is formation of bubbles in the pipeline.

Structural Hazards

These are caused due to lack of sufficient no. of functional units. We can take an example of pipelining in which floating time multiplier uses 10 times clock cycle in execution stage due to unavailability of functional units in this stage, hence no. of clock cycles required are more. In order to reduce this hazard, we can add no. of functional units in it.

The first problem is overlapping of instructions. It occurs whenever some delay occurs in pipelining. Suppose the decoder becomes slow hence previous bit remains decoded in register R2 and fetch register fetches bit after bit at regular time interval. Therefore, when clock t2 arrives, some sort of overlapping occurs and hence, output differs from actual value. To reduce this problem we use the technique of bubble pushing or forwarding. We have to just add one extra clock cycle for this.

Data Hazards

The second problem that arrives is data hazards. It occurs because of data dependencies between instructions. Suppose we have five different instructions in a pipeline and first one is add instruction, then its value will get stored in R1. Subsequent instructions are subtraction, and, or, x or to be executed. Now to execute all four of these instructions properly, the value of R1 really matters. The output of add register is really important for these as that creates data dependencies. To minimize data hazards, we use forwarding operands.

Control Hazards

This hazard mainly occurs because of branch or jump instructions that can alter execution of normal flow of pipeline. It can cause higher performance penalty as compared to data hazards. Overall performance of system is degraded. Branch target address is completed at the end of ID stage. If the output at branch is wrong we have to flash & refetch from memory. Control is jumped to a new location and then one cycle is wasted since target was unknown.

CONCLUSION

A 16 bit microprocessor is being designed with pipelining to get better results. The next step is simulation of design in Xilinx tool. Till now, we understood the effects of pipelining, its hazards and solutions. The processor can solve many differential equations and logical operations. A 5 stage pipelining is used. The synchronization of

various operations is done through external clock.

REFERENCES

I. R. Uma,Research Scholar, Department of Computer Science, Pondicherry University, Pondicherry(IJERA), Design and performance analysis of 8 bit RISC processor using Xilinx tool, Vol. 2, Issue 2, Mar-Apr 2012, pp.053-058

II. Jan Gray,Designing a single FPGA optimized RISC- CPU and system-on-chip,www.fpgacpu.in

III. Dimitris Mandalidis , Panagiotis Kenterlis and John N. Ellinas, A Computer Architecture Educational System based on a 32-bit RISC Processor, International Review on Computers and Software (I.RE.CO.S.), Vol. xx, n. x

IV. D. J. Smith, HDL Chip Design, International Edition, Doone Publications, 2000

V. Rohit Sharma, Vivek Kumar Sehgal, Nitin Nitin1, Pranav Bhasker, Ishita Verma , 2009, "Design And Implementation Of 64-Bit RISC Processor Using VHDL",UKSim : 11th International Conference on Computer Modeling And Simul