International Journal of Electronics Communication and Computer Engineering
Volume 2, Issue 1, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Designing & Simulation of 8-Bit MIPS RISC Processor

Akhilesh Singh Thakur
M. Tech Scholar, NIIST, Bhopal

Dr. Ravi Shankar Mishra
HOD, EC, NIIST, Bhopal

Prof. Puran Gaur
EC, NNIST Bhopal

Abstract— **This paper targets the implementation of a MIPS (Microprocessor without Interlocked Pipeline Stages) RISC (Reduced Instruction Set Computer) Processor via VHDL (Very high speed integrated circuit Hardware Description Language) design. The goal of this paper is to enhance the simulator based approach by integrating some hardware design & simulating them in pipelined (3 level) & non-pipelined modes so as to assess the performance of the processor in each of the modes.**

Index Terms— MIPS, RISC, PIPELINING, VHDL

I. INTRODUCTION

This paper all about designing & simulation of MIPS RISC processor with & without pipelining processes so as to compare the performance of the processor. As we all have an idea that pipelining is a process in which processing speed can enhanced by conducting parallel sessions of "Instruction Fetch", "Instruction Decode" & "Instruction Execution" [1]. In this way one can increase the processing speeds of a processor by saving clock cycles.

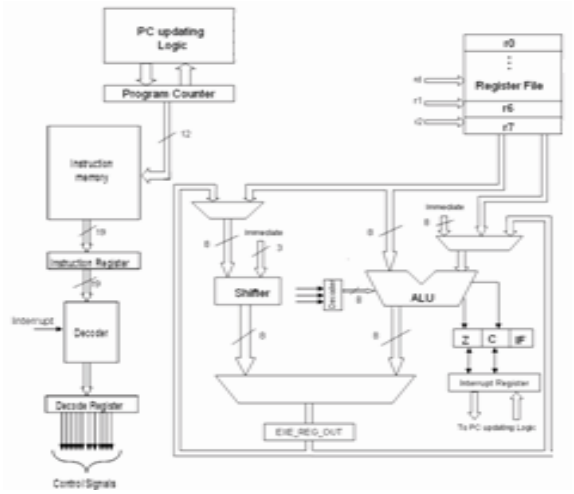


Fig.1 MIPS Single Cycle Processor **[2]**

The processor which has been simulated in this paper has an 8-bit, 7 Registers architecture [see Fig. 1].

II) THE MIPS INSTRUCTION SET ARCHITECTURE

Akhilesh singh Thakur is M.Tech Scholar in NIIST, Bhopal
R.S.Mishra is a professor in NIIST, Bhopal.
Puran Gaur is a Assistant Professor in NIIST, Bhopal.

As mentioned before MIPS is a RISC microprocessor architecture. The MIPS Architecture defines 8, 20-bit general purpose registers (GPRs). Register $r0 is hard-wired and always contains the value zero. The CPU uses byte addressing for word accesses and must be aligned on a byte boundary divisible by four (0, 4, 8, …). MIPS only has four instruction types as shown in **Fig. 2**. Table 1 provides a description of each of the fields used in the three different instruction types.

00	fn	rd	r1	r2	XXXXX

Fig. 2(a)

Fig. 2(a) gives the instruction format for arithmetic & logical operations in which data can fetched from the 2 registers namely, r1 & r2. The first two bits '00' indicate that it is an arithmetic or logical instruction. The field 'fn' indicates the kind of arithmetic or logical operation to be carried out. The field 'rd' indicates the location of the destination register. The fields r1 and r2 indicate the location of the operands. During the write back cycle, the data from the execution register is written into the destination register(rd).

01	fn	rd	r1	Const

Fig. 2(b)

Fig. 2(b) gives the instruction format for arithmetic & logical operation for **immediate** data operand. The first two bits '01' indicate that the instruction is arithmetic or logical. The field 'fn' indicates the kind of arithmetic or logical operation to be carried out. The field 'rd' indicates the location of the destination register. Here one of the operands is a value from the register file and the other is an immediate value.

III) MIPS SINGLE-CYCLE PROCESSOR VHDL IMPLEMENTATION

The MIPS implementation as with all processors consists of two main types of logic elements: combinational and sequential elements. Combinational elements are elements that operate on data values, meaning that their outputs depend on the current inputs. Such elements in the MIPS implementation include the arithmetic logic unit (ALU) and adder. Sequential elements are elements that contain and hold a state. Each state element has at least two inputs and one output. The two inputs are the data value to be written and a clock signal. The output signal provides the data values that were written in an earlier clock cycle. State elements in the MIPS implementation include the Register File, Instruction Memory, and Data Memory as seen in **Fig. 1** [11]. While many of logic units are straightforward to design

and implement in VHDL, considerable effort was needed to implement the state elements.

Once we determined how to declare the state elements of the MIPS processor it was time to implement the rest of the logic devices in VHDL. Because the final task is to pipeline the single-cycle implementation of the MIPS processor, we decided to modularize the single-cycle implementation into the five different VHDL modules to be fully utilized later in the pipelined implementation of the MIPS processor. The five modules are: Instruction Fetch, Instruction Decode, Control Unit, Execution, and Data Memory as shown in **Fig. 3** [3]**.** With the decision to use five different modules to implement the single-cycle MIPS processor, the VHDL design becomes a two-level hierarchy. The top-level of the hierarchy is a structural VHDL file that connects the
all five components of the single-cycle implementation, while the bottom-level contains the

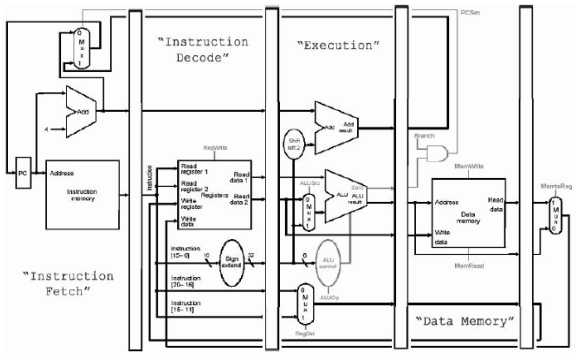
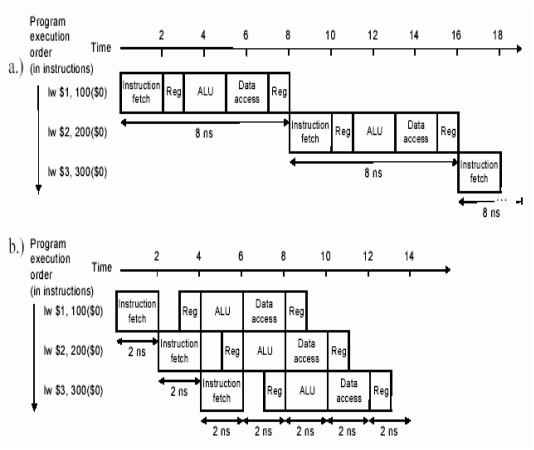


Fig. 3 Modularized MIPS Single Cycle **[1, 3]**

behavioral VHDL models of the five different components. Appendix C contains the top-level structural VHDL code for the MIPS single-cycle processor.

IV) MIPS PIPELINED PROCESSOR VHDL IMPLEMENTATION

Pipelining, a standard feature in RISC processors, is a technique used to improve both clock speed and overall performance [1]. Pipelining allows a processor to work on different steps of the instruction at the same time, thus more instruction can be executed in a shorter period of time. For example in the VHDL MIPS single-cycle implementation above, the datapath is divided into different modules, where each module must wait for the previous one to finish before it can execute, thereby completing one instruction in one long clock cycle. When the MIPS processor is pipelined, during a single clock cycle each one of those modules or stages is in use at exactly the same time executing on different instructions in parallel. **Fig. 4** shows an example of a MIPS single-cycle non-pipelined (a.) versus a MIPS pipelined implementation [15].
(b.) The pipelined implementation executes faster, keep in mind that both implementations use the same hardware components**.**



The MIPS pipelined processor involves five steps; the division of an instruction into five stages implies a five-stage pipeline:

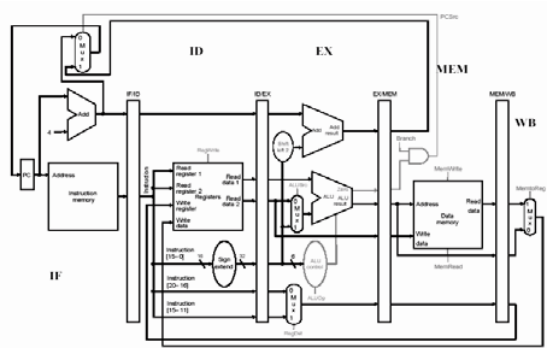
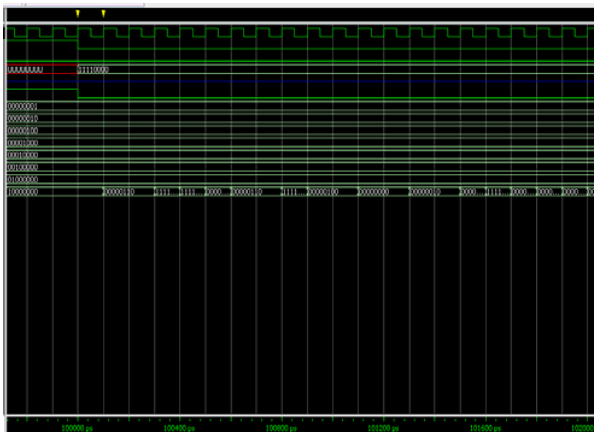


Fig. 5 MIPS Pipelined Processor Data path **[1, 3, 4]**

1. Instruction Fetch (IF)
2. Instruction Decode (ID)
3. Execution (EX)
4. Data Memory (MEM)
5. Write Back (WB)

Fig. 6 ModelSim XE III Wave Editor & Simulator Screenshot **[8]**



V. RESULT & DISCUSSION

This paper work depicts the functioning of MIPS processor in register direct & immediate mode. Although the ModelSim simulation shown in **Fig. 6** appears to be the same but the pipelined & non-pipelined modes can easily differentiated

from the synthesis report of Xilinx 9.2i Editor, shown in **Fig. 7 & 8**. Fig. 7 shows the synthesis report of the MIPS non-pipelined operation, whereas Fig. 8 depicts the MIPS pipelined operation. The summarized performance is compiled in Fig. 9. The VHDL designs of the MIPS processor were all simulated to ensure that the processors were functional and operational. **Fig. 6** shows a screenshot of the ModelSim XE IIIe Waveform Editor and Simulator results for the instructions shown in **Fig. 6**. The first two rows depict the global clock and reset signals. The following rows are executed during the Instruction Fetch stage of the MIPS pipelined processor. The signals are the PC value, used to index the instruction memory and the 19-bit instruction that was index out of the instruction memory. Please note that these values correspond to those shown in **Fig. 6**. **Fig. 7** shows the summary of device utilization & timing constraints observed in non-pipelined & pipelined modes. The figure proves the utility of pipelining in RISC microprocessor operation. The synthesis is done for Xilinx XC3S400 FPGA package (PQ208 family). The 3 level pipelining namely Instruction fetch, Instruction decode & Operation execution, has improved the timing from **15.543 ns** to **8.837ns** (non-pipelined v/s pipelined). The conclusion is data input can fed at a rate of **113.164 MHz.**

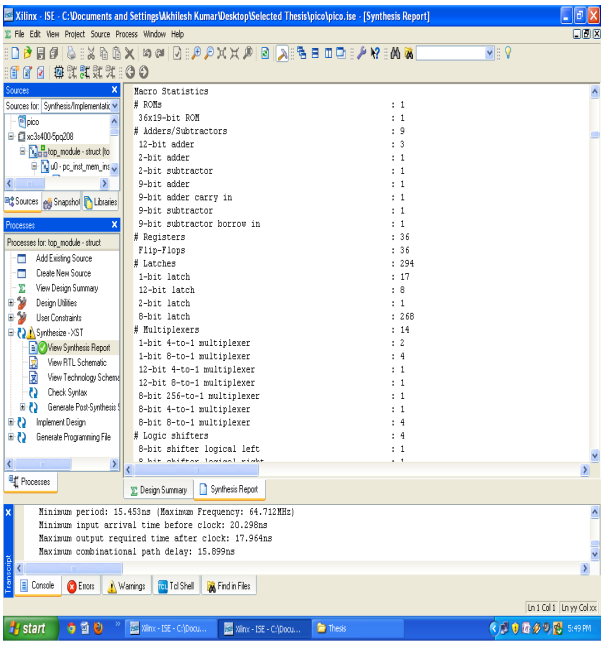


*Fig.7 XILINX 9.2i Synthesis Report of Non-Pipelined MIPS **[6, 7, 9]***

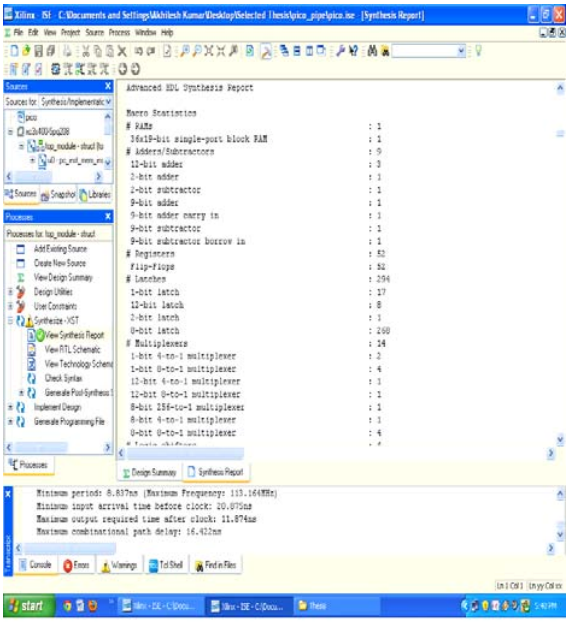


*Fig.8 XILINX 9.2i Synthesis Report of Pipelined MIPS **[6, 7, 9]***

Components	Non- Pipelined	Pipelined
Minimum Delay	15.543 ns	8.837 ns
Maximum i/p Frequency	64.712 MHz	113.164 MHz
36 X 19 bits RAM	1	1
Adders/Subtractors	9	9
Registers	36	52
Latches	294	294
Multiplexers	14	14
Logic Shifters	4	4
XORs	9	9

*Fig.9 Non-Pipelined v/s Pipelined MIPS Summary **[6, 7, 9]***

VI. REFERENCES

[1] Patterson, D. A., Hennessy, J. L., Computer Organization and Design: The Hardware/Software Interface, 2nd edition, Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[2] MIPS Technologies, MIPS19™ Architecture For Programmers Volume I: Introduction to the MIPS19™ Architecture, rev. 2.0, 2003.

[3] Diab, H., Demashkieh, I., "A reconfigurable microprocessor teaching tool", IEEE Proceedings A, vol. 137, issue 5, September 1990.

[4] Takahashi, R., Ohiwa, H., "Situated Learning on FPGA for Superscalar Microprocessor Design Education", IEEE Proceedings of the 16th Symposium on Integrated Circuits and System Design, 2003.

[5] Land. B, Electrical Engineering 475 Microprocessor Architectures, http://instruct1.cit.cornell.edu/Courses/ee475/

[6] Gray, J., Designing a Simple FPGA-Optimized RISC CPU and System-on-a-Chip, 2000.

[7] Brown, S., Vranesic, Z., Fundamentals of Digital Logic with VHDL Design, McGraw-Hill Publishers, 2002.

[8] Larus, J. R., "*MODELSIM S20: A MIPS R2000 Simulator*", 1993.

[9] IEEE. IEEE Standard VHDL Language Reference Manual. IEEE, New York, NY, 2002. IEEE Standard 1076-2002.