Design and development of a 5-stage Pipelined RISC processor based on MIPS

Ankitha C.S[1], Dr. Kiran V[2]

[1]MTECH 1st year, Dept. of ECE, RV college of Engineering, Karnataka, India
[2]Associate Professor, Dept. of ECE, RV college of Engineering, Karnataka, India

---***---

Abstract - The MIPS-based RISC processor has a extensive applications due to its outstanding performance as well as lesser power consumption. Here, is the concept of a MIPS processor built on a pipeline is suggested, using the forwarding and stopping processes. The majority of modern processors feature a pipelined architecture. The design of pipelined processors enables the execution of several instructions at a single clock pulse. The processors are built with numerous stages to produce this phenomenon of multiple processing. Under a shared clock pulse, these stages operate as a single entity to produce the effect of multiple processing. A five stage pipelined RISC processor typically has the following stages: instruction fetch, instruction decode, execution, memory, and Wright back stages. Each of these steps functions as a separate processor coupled to the preceding one so that each of them processes one instruction and sends it to the following step in the sequence with each clock pulse. A 32 bit RISC processor with a 5-stage pipeline has been designed in the proposed project. By considering a minimal set of instructions from the instruction set architecture of the popular RISC processor MIPS32 for implementation. The behavioral model of the processor is written in Verilog HDL and the operation of the processor and the pipeline structure are studied and verified.

Key Words:* *MIPS Processor, Pipeline, RISC, Verilog.

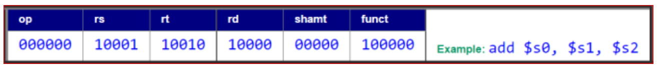
1. INTRODUCTION

There are numerous types of processors available today. The majority of them are created using a hardware descriptive language, such as Verilog or VHDL. There are two different types of processors: RISC and CISC based. As a replacement for CISC, RISC processors are thought to be the most efficient CPU architecture.It comprises of streamlined instructions that perform fewer tasks yet more quickly. The ability of RISC processors to use thread level parallelism and instruction level parallelism to enhance the register set and internal parallelism is its core feature. Additionally, it enables the CPU to execute instructions at a faster rate. Smartphones, tablets, computers, and many other intelligent gadgets use RISC architecture. RISC processor components not only perform better but are typically less expensive to develop and manufacture since they use fewer transistors.Because of the previously described factors, system on chips (SoCs) are the ideal fit for RISC architectures. MIPS architecture designs remain constant, however they could change according on whether the pipeline is single-cycle or multi-cycle implemented [2]. Routers and other similar tiny computing devices employ MIPS.

The MIPS design has three sorts of instruction sets: Register type , Immediate type and jump type

1.1 Register type (R type)

Here the opcode is indicated by the last 6 bits. The next 15 bits stand for the three registers Rs, Rt, and Rd, which are used for operations. Source registers are Rs, Rt, and destination registers are Rd, respectively. The next five bits are the shift amount, which indicates how many bits need to be shifted. The function field in the last six bits identifies the operation to be executed on the registers. Fig. 1 illustrates the structure of a R type instruction.

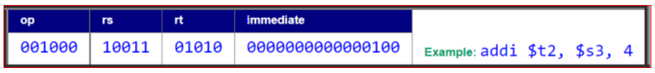


op	rs	rt	rd	shamt	funct	
000000	10001	10010	10000	00000	100000	Example: add $s0, $s1, $s2

Fig. 1. Format for R type instructions.

1.2 Immediate type (I type)

When an instruction needs to operate on both a register value and an immediate value, an I instruction is utilized. The source and destination register operands, rs and rt, are each 5 bits in size. The immediate value is 16 bits long (0 to 15). This value may be stated in two's complement depending on the instruction and is typically used as the offset value in various instructions. Fig. 2 depicts the I type Instruction format.

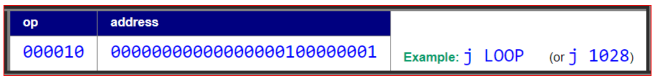


op	rs	rt	immediate	
001000	10011	01010	0000000000000100	Example: addi $t2, $s3, 4

Fig.2. Structure of I type instruction.

1.3 Jump type (J type)

The first 5 bits of jump type Instruction structure specify the type of jump operation that needs to be carried out. The branch offset is represented by the 2's complement in the remaining 26 bits. Instruction field for J-type is shown in Fig. 3.



op	address	
000010	00000000000000000100000001	Example: j LOOP (or j 1028)

Fig. 3. Instruction field for a jump-instruction

2. RELATED WORK

The Harvard architecture, which Von Neumann introduced, was used in the construction of the processor. It has distinct program memory and data memory in the CPU. Clock gating, an effective low power approach, is employed at the architecture level to reduce RISC Core's power usage. Low power usage reduces heat dissipation, extending battery life and improving device dependability [3]. Multiple executions of a single instruction can be achieved

3. METHODOLOGY

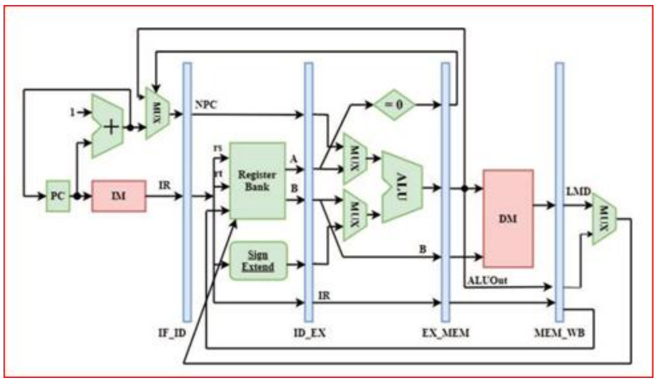


Fig.4. Block diagram of MIPS processor's architecture

Five steps of the pipelining procedure have been completed using this specified methodology. They are instruction fetch stage (IF), instruction decode stage (ID), execute stage (EX), memory access stage (MEM), and write back stage (WB). Utilizing bottom-up design methodology, the processor's sub-module was initially created, coded, and tested. At first when each of the sub-modules had been generated and were confirmed to be entirely functioning, these are incorporated into a top module to produce the MIPS processors. A broad set of MIPS instructions were then executed on the CPU to assess its functionality and timing. The instructions are executed, intentionally and methodically by moving through each stage. Each phase completes its assigned responsibilities and adds to the final project. Also registers separating each stage of the pipeline operation aid in buffering. One clock cycle is used to represent the time provided for each stage.

3.1 Various Pipelining Stages

3.1.1 IF-stage:

At this point, the CPU reads instructions from memory at the address represented by the value of the Instruction Pointer (IP) or Program Counter (PC). The instruction's Opcode is received from the Instruction Memory which stores all of the instructions required for the process.

3.1.2 Instruction Decode:

The decoder stage's objective is to transmit and decode the fetched instructions to the control unit. The ID stage receives the data from the IF stage. The Opcode is acquired by the decoder unit via the fetch stage. In accordance with the Opcode and its related functions, ID stage regulates different components and modules of the design. Four different sorts of instructions make up the decode stage: Register-type, Immediate-type, Jump-type, and I/O type instructions. Control unit's actions are carried out in accordance with the Opcodes of these instructions. One among the crucial components as well as the brain of the processor and ID phase is controller, which receives the instruction's Opcode of size 6-bits as input and generates the Execute instruction as the output and that is sent to the Arithmetic and Logical Unit. The Execute Command specifies a particular activity that ALU must carry out. When a threat is identified, the controller stops all writing operations to the register file / memory. Instruction decode stage includes the Sign Extend unit. This module converts the final 16 bits of the immediate instruction—the immediate data—into a 32-bit value. The Immediate Instruction's MSB bit value is added to the remaining bits to perform sign extension.

3.1.3 Execute:

In the EX-stage, calculations are performed. The ALU is the main part of the EX module. The central component of a processor is the arithmetic logic unit. It is in charge of all tasks that need to be done. ALU processes the operations based on the Execute Command input. The ALU performs shift operations both logical and arithmetic, as well as arithmetic and logical operations including ADD, SUB, AND, OR, NOR, and XOR.

3.1.4 Memory Access:

The loading and unloading of values into and out of registers is the main activity and function in the memory stage. They are also in charge of retrieving and transmitting the data from the memory module. The main job of the data value is to be stored in the appropriate destination registers in accordance with the instruction.

3.1.5 Write-back (WB):

In this stage the calculated or retrieved value will be written back to the register specified in the instructions. It should be noted that two distinct stages are simultaneously accessing the register file: the decode stage is reading two source registers, and the WB stage is writing the destination register of a previous instruction.This phase is in charge of the outcome, information storage, and data input into and out of the register. Pipeline Registers or Buffers—separate types of memory—divide these pipelined phases. Such buffers are employed to divide the phases, preventing problems from occurring when several instructions are carried out simultaneously without any data dispute. Since the purpose of the WB stage is to write data to the destination register. For instance, the R1 register is produced by the ADD R1, R2, and R3 instruction memory to speed up programs.

4. RESULTS

Simulation results for different assembly programs loaded onto the processor.

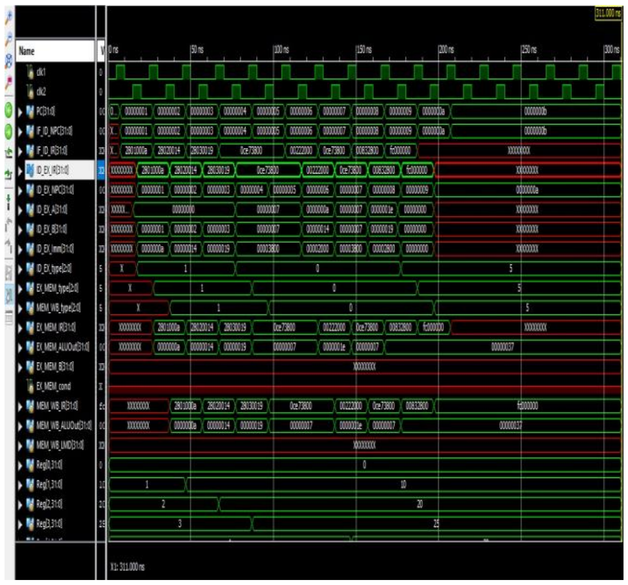


Fig.6 Simulation result for the program to verify register manipulation

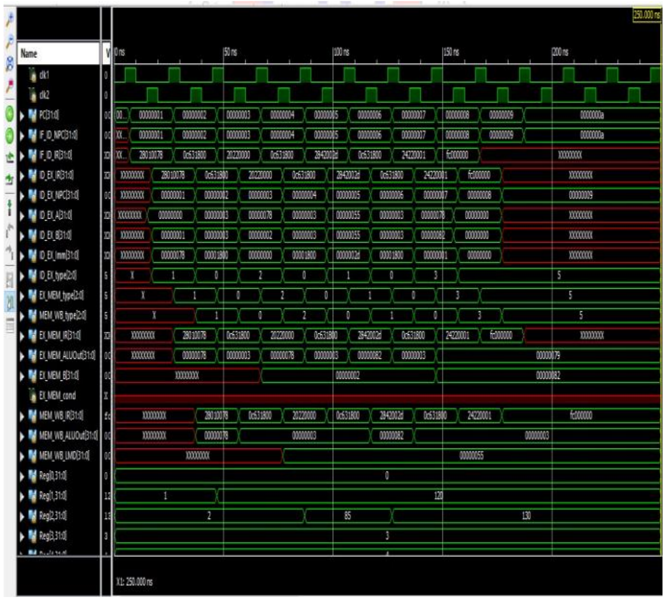


Fig.8 Simulation result for the program to verify register manipulation

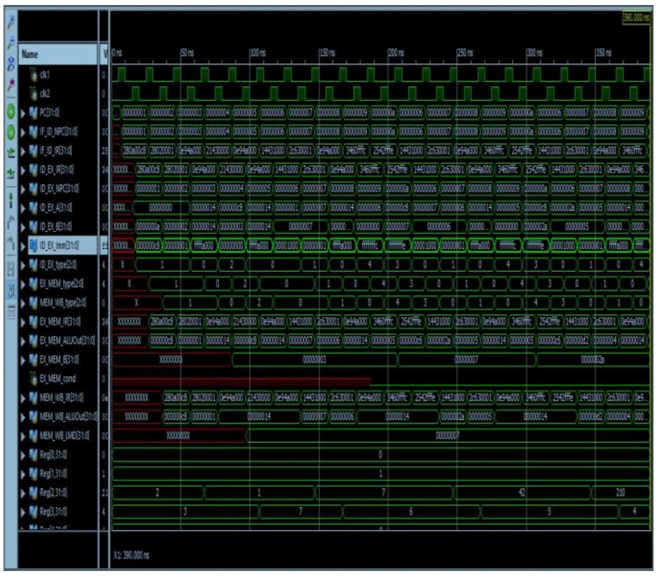


Fig.9 Simulation result for the program to verify register manipulation

5. CONCLUSION

In the proposed project, a 32-bit MIPS-based RISC processor is successfully designed, modelled, and simulated. With a pipeline design and Verilog HDL, it is effectively executed. It is determined that the functionality is correct by comparing the simulation's results to those that were predicted.

REFERENCES

[1] Preetam Bhosle, Hari Krishna Moorthy, "FPGA implementation of low power pipelined 32-bit RISC processor", International Journal of Innovative Technology and Exploring Engineering (IJITEE), August 2012.

[2] J.Poornima, G.V.Ganesh, M.Jyothi, M.Sahithi, A.Jhansi Rani B. Raghu Kanth, "Design and implementation of pipelined 32-bit Advanced RISC Processor for Various D.S.P Applications", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (1) , 2012.

[3] Navneet kaur, Adesh Kumar, Lipika Gupta, "VHDL Design and Synthesis of 64 bit RISC Processor System on Chip (SoC)", IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 3, Issue5 (Nov. – Dec. 2013),

[4] M. N. Topiwala and N. Saraswathi, "Implementation of a 32-bit MIPS based RISC processor using Cadence," 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, 2014.

[5] N.Alekya ,P.Ganesh Kumar, "Design of 32-Bit Risc CPU Based on MIPS", JGRCS Volume 2,No 9,September 2011

[6] Sagar Bhavsar, Akhil Rao, Abhishek Sen, Rohan Joshi, "A 16-bit MIPS Based Instruction Set Architecture for RISC Processor", International Journal of Scientific and Research Publications, Volume 3, Issue 4, April 2013

[7] Bhardwaj, Priyavrat & Murugesan, Siddharth, "Design & simulation of a 32-bit RISC based MIPS processor using verilog", IJRET, 2016.

[8] Anand Nandakumar Shardul,"16-Bit RISC Processor Design for convolution Application", International Journal of Advancements in Research & Technology, Volume 2,Issue 9,September-2013.

[9] K. P. K and V. Prakash A. M, "Designing and Implementation of 32-bit 5 stage Pipelined MIPS based RISC Processor Capable of Resolving Data Hazards," 2021 IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC), 2021, pp. 1-6, doi: 10.1109/ICMNWC52512.2021.9688435.

[10] Balpande, Rupali S.,and Rashmi S. Keote. "Design of FPGA Based Instruction Fetch & Decode Module of 32-bit RISC Processor", 2011 International Conference on Communication Systems and Network Technologies, 2011