DESIGN & SIMULATION OF A 32-BIT RISC BASED MIPS PROCESSOR USING VERILOG

Priyavrat Bhardwaj[1], Siddharth Murugesan[2]

[1]*Department of Electrical & Electronics Engineering, Guru Tegh Bahadur Institute of Technology, New Delhi, India*
[2]*Department of Electrical & Electronics Engineering, Guru Tegh Bahadur Institute of Technology, New Delhi, India*

Abstract

This research paper presents design & simulation of a high performance five stage pipelined 32-bit Microprocessor without Interlocked Pipeline Stages (MIPS),which is a Reduced Instruction Set Computing (RISC) architecture based processor. The purpose of RISC microprocessor is to execute a minuscule batch of instructions, with the intention of proliferating the celerity of the processor. This processor was designed with5 phases of pipeline in particular Instruction Fetch (IF), Instruction Decode& Register Fetch (ID), Execution& Address Calculation (EX), Memory Access (MEM) and Write Back (WB) modules. The designing process was done using a myriad of modules which are the ALU, Control Unit, Program Counter, MUX, Instruction Memory, Data Memory, CPU, Register File, Sign Extension. The designing of this processor is developed using the Hardware Description Language (HDL) - Verilog in ModelSim simulator. The supreme aim of this paper is to develop the RTL logic design using Xilinx tool.

Keywords- *MIPS, RISC, CISC, Verilog, RTL, ModelSim, Xilinx*

----------***----------

1. INRODUCTION

Microprocessors & Microcontrollers are generally designed in the vicinity of two main computer architectures: Complex Instruction Set Computing i.e. CISC architecture and Reduced Instruction Set Computing i.e. RISC architecture. The concept of CISC is based on Instruction Set Architecture (ISA) design that redoubles performing further with several instructions utilizing changeable number of operands and an out spread variation of addressing modes in disparate locations in its Instruction Set. Thus causing them to have varying execution time and lengths thereby authoritatively mandating an intricate Control Unit, which inhabits an immensely existent region on the chip. Compared with their CISC analogue, RISC processors typically support a minuscule set of instructions. A display that juxtaposes RISC processor with CISC processor, the number of instructions in a RISC Processor is low while the number of general purpose registers, addressing modes, fixed instruction length and load-store architecture is more this in turn facilitates the execution of instructions to be carried out in a short time thus achieving higher overall performance [1].

Currently, the efficacy of the RISC processors is generally accepted to be greater than that of their CISC counterparts. Before their execution the instructions are translated into RISC instructions in even the most popular CISC processors. The attributes mentioned above accentuate the design strength of RISC in the market for embedded systems known as "system-on-a-chip (SoC)"[12]. The premier micro processors exhibiting reduced instruction set are SPARC, ARM, MIPS and IBM's PowerPC. RISC processor typically has load store architecture. This denotes there are two instructions for accessing memory which are a load instruction set to load data from the memory and store instruction set to Write Back (WB) the data into memory without any instructions [1].

2. MIPS INSTRUCTION SET ARCHITECTURE

The instruction set can be categorized under three classifications in the MIPS ISA, these are: Register type (R), Immediate type (I) and Jump type (J). Each instruction starts with a 6-Bit Opcode. Alongside these opcode, Register type (R)define 3 (three) registers, Immediate ype (I) instructions define2 (two)registers and a 16Bitevaluation; Jump type (J) instructions have an opcode of 26Bit[1].

The following table demonstrates the three formats used for the MIPS core instruction set architecture:

Table 1

Type	-31- -0- format (bits)					
R	Opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
J	Opcode (6)	rs (5)	rt (5)	immediate (16)		
I	Opcode (6)	address (26)				

2.1 Register (R) Type Instructions:

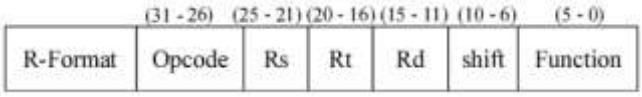

Fig. 1. Register-Type Instructions

The instruction format illustrated in Fig. 1.is that of Register (R) Type. In it the Opcode is represented by the last 6 bits. The 3 register types on which the operations are executed are Rs, Rt and Rd which are represented in the above illustration by 15-bits that follow the Opcode. The starting or the source registers are Rs andRt while the ending or target register is Rd. Succeeding 5-bits are the shift sum which betokens the number of bits that are to be moved. The final 6-bits represent the function field points to the function which are to be executed on the registers.

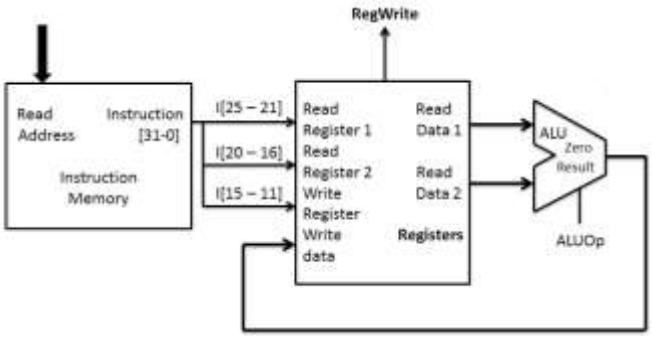

Fig. 2. Register Type Instructions Data Path

Fig. 2 illustrates Register (R) type Instruction's Data path [1]. Its main usage is in performing mathematical operations such as addition and subtraction. E.g. add Rd, Rs, Rt. In this the value received by Rd is the signed addition of Rs and Rt i.e. Rs plus (+) Rt.

2.2 Immediate (I) Type Instructions:

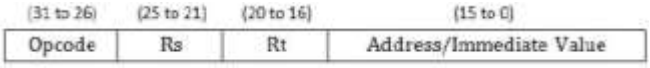

Fig. 3. Immediate Type Instructions

Immediate (I) type instructions are demonstrated in Fig.3. The four fields portrayed in this type of arrangement represent - the Opcode, which is of 6-bitthat is utilized to select the Instruction type, storing of data is done in the Source Register and Target Registers which are Rs and Rt respectively. Each are of 5-Bit. The final 16-bit Address/Immediate Valuefield wielding prompt data.

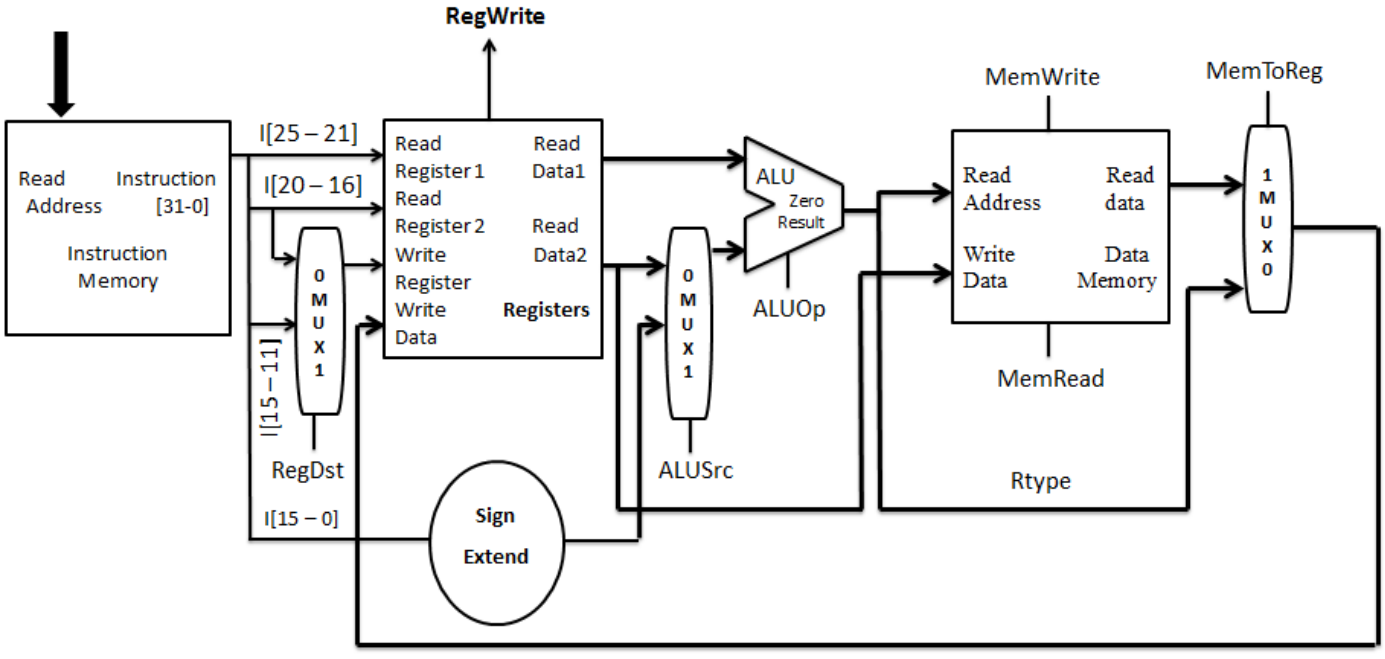

Fig. 4. Data Path for Immediate (I) type instructions

Fig. 4 delineates the data path for Immediate (I) type instruction [1]. It demonstrates the dichotomy of Rt register which can be used both as a source and a destination. The immediate value received by sign extend is represented by the last 16-bits which is then sent to the Arithmetic and Logical Unit for playing out the desired function.

2.3 Jump (J) type Instructions (Branch Format)

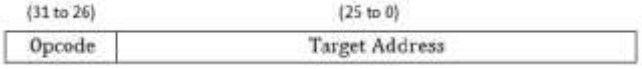

Fig. 5. Jump Type Instructions

The instruction format shown in Fig. 5. is that of Branch Type [1]. The two fields illustrated in this arrangement type are the Opcode which is of 6-bit, utilized to choose the kind of instruction organization and Ending or Target address of 26-bit, utilized to determine where the address has to be branched.

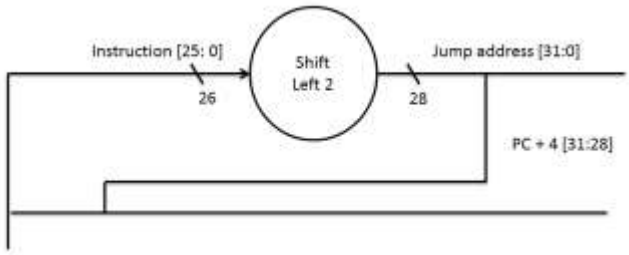

Fig. 6. Data Path for Jump Type Instructions

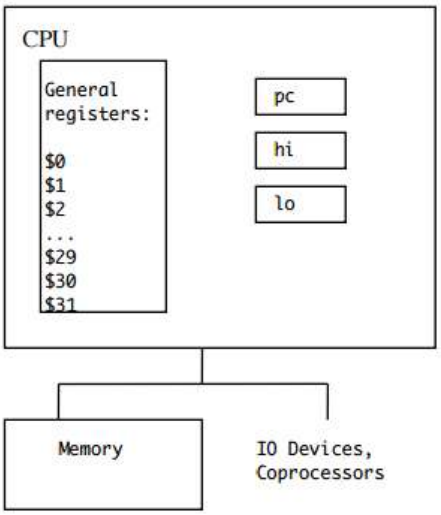

Fig. 7. Basic MIPS Architecture

The data path for Jump (J) type instruction is illustrated in Fig. 6.[1]. The figure demonstrates that a 32-bit jump (J) address is obtained when the last 4 bits of PC + 4 are attached to the shift left by 2 values of a 26-bit instruction captured out from MEM. In addition, it jumps to the destination by omitting any alternate instruction.

3. MIPS ARCHITECTURE

The accompanying outline demonstrates the fundamental architecture of a MIPS-based framework:

Microprocessor without Interlocked Pipeline Stages (MIPS) is a RISC (Reduced Instruction Set Computing) architecture. Pipelined MIPS has five stages which are IF, ID, EX, MEM and WB. Pipelining means several operations in single data path at the same instant. Pipelining is used to enhance the capabilities of the RISC processor which is the reason for its utilization in this type of computer architecture. A multicycle CPU comprises of countless tasks. So if one task occurs, rather than waiting for the process to finish, at the same time another task is initiated in the same data path simultaneously without interfering with the previous task. The processes is thus divided into different pipelined stages. Following every clock a new operation is instigated in the pipeline stage to which the process is being fed to. The triggering is done without causing any interruptions to the past process. This makes simultaneous utilization of all stages in the data path possible. This thusly can increment the throughput of MIPS.

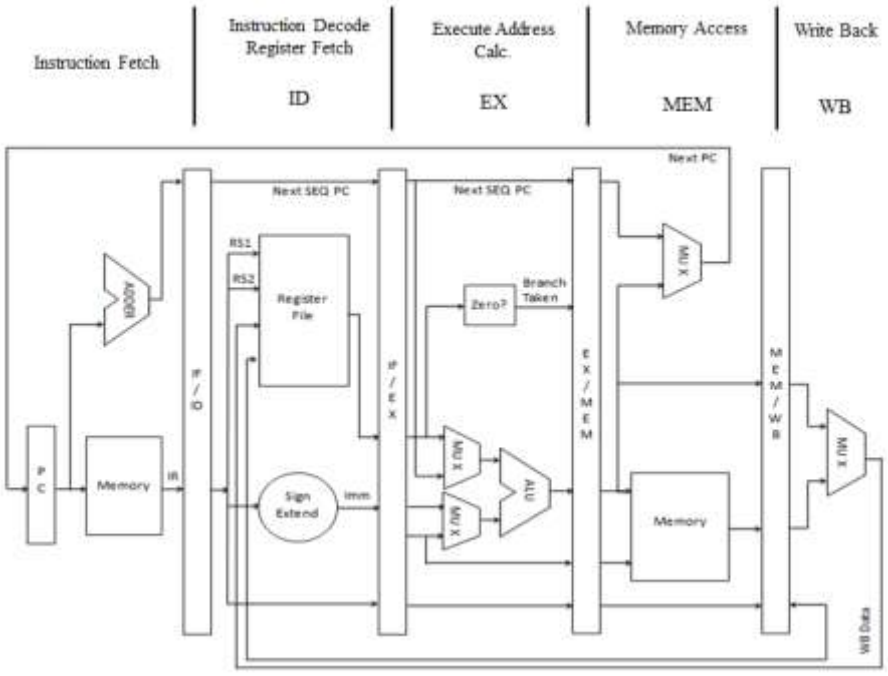

Fig. 8. 5-Stage Pipelined MIPS

MIPS processor has been executed utilizing five pipeline stages, which are Instruction Fetch (IF), Instruction Decode (ID), Execution (EX), Memory access (MEM) and Write Back (WB).The isolation of these stages is achieved by special registers known as pipeline registers. The aim of these registers is to isolate the stages of the instructions so that there is no inadmissible information because of various directions being executed all the while. They are named in the middle of each of these: IF/ID Register, EX/MEM Register and MEM/WB Register. The data path demonstrated in Fig. 8. is that of the MIPS pipelined processor.

3.1 Instruction Fetch (IF)

The command relayed to the Program Counter (PC) to fetch the instruction from the cache memory is what instigates the primary pipelining operation of the IF stage. The storage of PC and Instruction for the successive clock cycle is done in the IF/ID pipelined register as RAM (Random Access Memory)

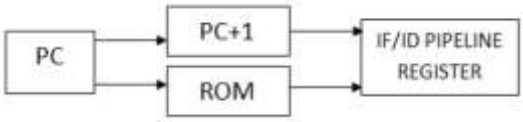

Fig. 9. IF Stage representation

IF stage for the most part relies on upon PC's represent value. On the basis of the PC value the processor gets the instructions from the cache and followed by which the Program Counter value is incremented by 1. Thus, the IF/ID register receives this information followed by which the information is relayed to the decoder unit. The Instruction Fetch (IF) stage operation has been represented in Fig. 9 [22].

3.2 Instruction Decoder (ID)

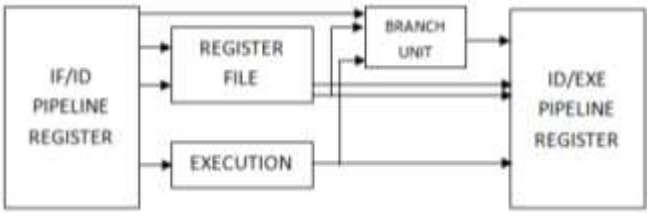

Fig. 10. ID Stage representation

The Opcode is relayed to the decoder unit at the instant when the instruction is obtained from the IF stage. Instruction Decoder ID stage directs the controlling command to the various units of the MIPS processor examining the Opcode of the instructions. Thus the procurement of data from the MIPS registers is carried out by the Read register. The Branch unit is likewise incorporated into Instruction Decoder (ID) stage. The Input data of ID stage is received from IF stage as shown in Fig. 8.This decoding stage includes four different instructions: Register (R) type, Immediate (I)type, Jump (J) type and Input/Output (I/O) type instructions. Depending upon these instructions the function will be performed utilizing above mentioned formats. Fig. 10. indicates Instruction Decode (ID) stage operation [22].

3.3 Execute (EX)

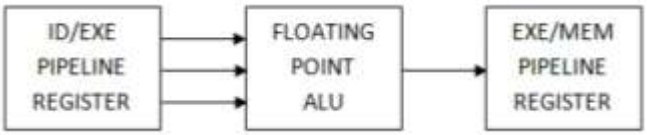

Fig. 11. Execution Stage (EX) representation

Following the Instruction Decoder(ID), the instructions are sent to execute stage(EXE or EX). Execute (EX) stage performs Arithmetic and Logical Unit (ALU) processes. Execution of operations is the fundamental aspect of Execute (EX) stage, for instance arithmetic operations such as addition and difference and OR & AND. In particular, EX/MEM pipelined register receives the result upon the execution of specific instructions (i.e. FP ALU). Execute stage representation is shown in Fig. 11. [22].

3.4 Memory Access &Input/Output (MEM)

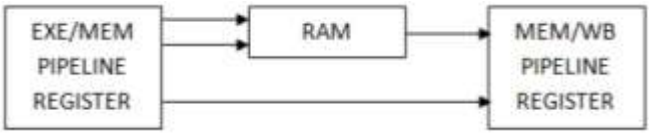

Fig. 12. Memory Access representation

The storing and loading of values along with inputting and outputting data from the processor is the primary function of the memory access (MEM) stage. The outcome will be dispatched to the WB stage in a scenario where the instruction is neither memory nor IO instruction. After the result is figured the primary function is to store the data values in the destination register. The Memory Access (MEM) stage operation is demonstrated in Figure 12 [22].

3.5 Write Back (WB)

Fig. 13. Write Back representation

As per Fig. 13., the Write-Back (WB) operation is the final stage of the RISC based MIPS architecture which composes the result, store information and input data from and to the register file [22]. Writing the data that has been fetched from the MIPS register to the target register is the main aim of this stage.

4. RESULT

4.1 RTL Design

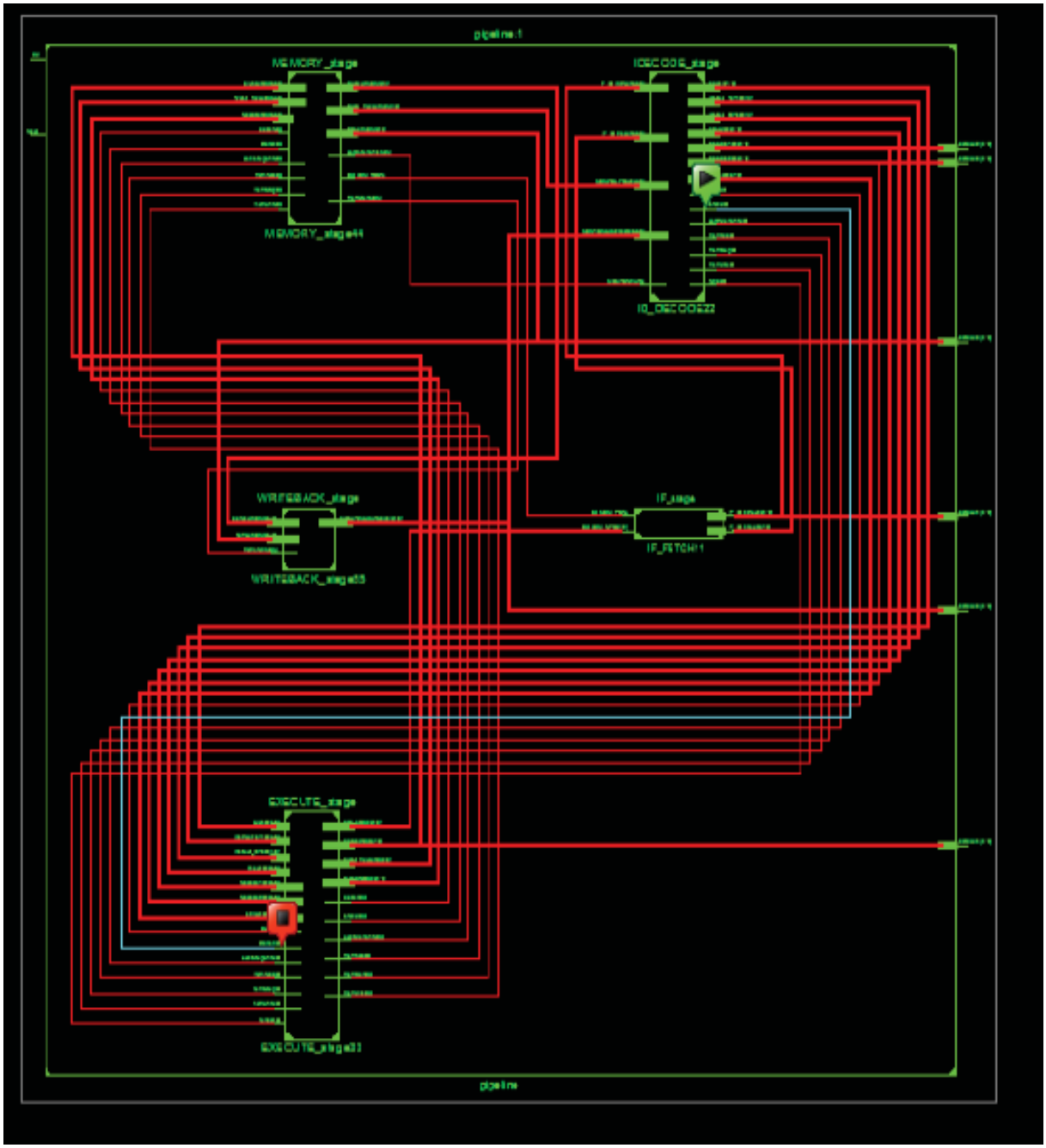

Fig. 14. RTL Design view of RISC based MIPS Processor

As per the design, the Write-Back (WB) operation writes back the result, stores the information and inputs the information to the Register file and vice-versa. For instance, the result is given by ADD Rd, Rs, Rt instructions. It is analogous to the Feedback operation in various engineering systems. The Register Transfer Logic as illustrated in Fig. 14. is that of 32-Bit RISC based MIPS Processor [2]. It contains Instruction decoder (ID) unit, Instruction Fetch (IF) unit, memory unit (MEM) and execution unit (EX). In this process, the program counter (PC) is utilized while fetching the Opcode from Instruction Fetch (IF) stage and sending the code to Instruction Decode (ID) stage. Instruction Fetch (IF) stage receives the Opcode from the ID unit which is then sent for execution in the EX stage. The instruction configuration chosen is dependent on the Opcode. The execution of the instruction in the EX stage occurs in accordance with the assigned Opcode. Storage of Opcode to the memory and fetching it from the memory is the primary task of the memory unit.

4.2 Simulation Result

The Resulting output waves of 5-staged pipelined MIPS RISC processor is shown below:

Fig. 15 (a). Output Wave Simulation

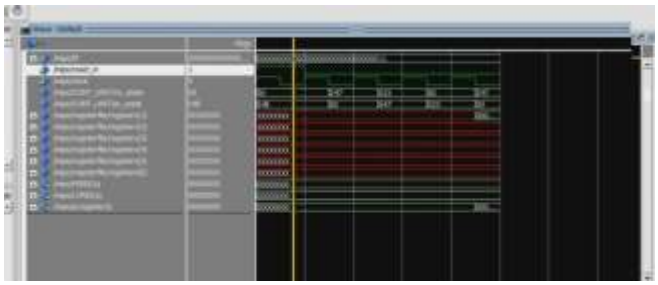

Fig. 15 (b). Output Wave Simulation

Fig. 15 (c). Output Wave Simulation

As illustrated in the waveforms, Fig. 15 (a), (b) and (c) demonstrates simulation result for R-type, I-type & J-type instruction formats.

5. CONCLUSION

This research paper outlines a 32-bit Microprocessor without Interlocked Pipeline Stages (MIPS) based RISC processor is executed effectively with pipelining. In a five stage pipelining system the execution of each direction occurs in a single clock cycle. This design demonstrates the usage of MIPS based CPU equipped for taking care of different Register type, Jump type and Immediate type of instructions and each of these classifications has a diverse configuration.

REFERENCES

[1] Mohit N. Topiwala, N. Sarawathi, "Implementation of a 32-bit MIPS Based RISC Processor using Cadence", International Conference on Advanced Communication Control and Computing Teclmologies (ICACCCT), 2014 IEEE.

[2] Pranjali S. Kelgaonkar, Prof. ShilpaKodgire, "Design of 32 Bit MIPS RISC Processor Based on Soc", International Journal of Latest Trends in Engineering and Technology (IJLTET), January 2016.

[3] Ramandeep Kaur, Anuj, "8 Bit RISC Procesor Using Verilog HDL", Int. Journal if Engineering Research and Applications, March 2014.

[4] PreetamBhosle, Hari Krishna Moorth, "FPGA Implementation of low power pipeline 32-bit RISC Proessor", International Journal of Innovative Technology and Exploring Engineering (IJITEE), August 2014.

[5] Gautham P, Parthasarathy R, Karthi, Balasubramanian, "Low Power Pipelined MIPS Processor Design", in the proceedings of the 2009, 12th international symposium, 2009 pp. 462-465.

[6] Neenu Joseph, Sabarinath S, "FPGA based Implementation of High Performance Architectural level Low Power 32-bit RISC Core", 2009 IEEE.

[7] "Computer Organization and Design- the hardware/software interface", 3rd edition by David A. Patterson and John L. Hennessy, pp. 370-412.

[8] Mrs. RupaliBalpande, Mrs. RashmiKeote, "Design of FPGA based Instruction Fetch & Decode Module of 32-bit RISC (MIPS) Processor", International Conference on Communication Systems and Network Technologies, 978-0-7695-4437-3/11, 2011 IEEE.

[9] Harpreet Kaur, Nitika Gulati, "Pipelined MIPS with Improved Datapath", IJERA, Vol. 3, Issue 1, January – February 2013, pp. 762-765.

[10] Sharda P. Katke, G.P. Jain,"Design and Implementation of 5 Stages Pipelined Architecture in 32 Bit RISC Procesor", IJETAE, Volume 2, Issue 4, April 2012, pp. 340-346.

[11] PejmanLotfi-Kamran, Ali-Asghar, ZainalabedinNavabi, "Dynamic Power Reduced of Stalls in Pipelined Architeture Processors", International Journal of Design, Analysis and Tools for Circuits and Systems, Vol. 1, No. 1, June 2011.

[12] Neeraj Jain, "VLSI Design and Optimized Implementation of a MIPS RISC Procesor using XILINX Tool", International Journal of Advancd Research in Computer Science and Electronics Engineering (IJARCSEE) Volume 1, Issue 10, December 2012.

[13] Mamun Bin IbneReaz, MEEE, Md. Shabiul Islam, MEEE, Mohd. S. Sulaiman, MEEE, A Single Clock Cycle MIPS RISC Processor Design using VHDL, ICSE2002 Proc. 2002, Penang, Malaysia, 0-7803-7578-S/02/S, 2002 IEEE.

[14] Kui YI, Yue-Hua DING, 32-bit RISC CPU Based on MIPS Instruction Fetch Module Design, 2009 International Joint Conference on Artificial Intelligence, 978-0-7695-3615-6/09, 2009 IEEE.

[15] Rohit Sharma, Vivek Kumar Sehgal, Nitin Nitin1, Pranav Bhasker, IshitaVerma, Design and Implementation of a 64-bit RISC Processor using VHDL, UKSim 2009: 11th International Conference on

Computer Modelling and Simulation, 978-0-7695-3593-7/09, 2009 IEEE.
[16] Pravin S. Mane, Indra Gupta, M. K. Vasantha, "Implementation of RISC Processor on FPGA", 1-4244-0726-5/06, 2006 IEEE.
[17] Shuchita Pare, Dr. Rita Jain, 32Bit Floating Point Arithmetic Logic Unit ALU Design and Simulation, Dec 2012, IJETECS. Bai-ZhongYing, Computer Organization, Science Press, 2000.11.
[18] Wang-AiYing, Organization and Structure of Computer, TsinghuaUniversity Press, 2006.
[19] Wang-Yuan Zhen, IBM-PC Macro Asm Program, Huazhong Universityof Science and Technology Press, 1996.9.
[20] MIPS Technologies, Inc. MIPS32™ Architecture For ProgrammersVolume II: The MIPS32™ Instruction Set June 9, 2003.
[21] Zheng-WeiMin, Tang-ZhiZhong, Computer System Structure (Thesecond edition), Tsinghua University Press, 2006.
[22] Mr. Sagar P. Ritpurkar, Prof. Mangesh N. Thakare, Prof. Girish D.Korde, "Review on 32Bit MIPS RISC Processor using VHDL",International Conference on Advances in Engineering & Technology –2014 (ICAET-2014), PP 46-50, IOSR.
[23] PowerPC 755/745 RISC Microprocessor 2013 datasheet, e2vsemiconductors SAS.
[24] AM1806 ARM Microprocessor 2014 datasheet, TEXASINSTRUMENTS.
[25] AM1808 ARM Microprocessor 2014 datasheet, TEXASINSTRUMENTS.
[26] ADSP-BF592 Blackfin Embedded Processor 2013 datasheet, AnalogDevices.
[27] Sitara AM335x ARM Cortex-A8 Microprocessors (MPUs) 2013datasheet, TEXAS INSTRUMENTS.
[28] MPC7457 RISC Microprocessor 2013 datasheet, FreescaleSemiconductor.
[29] STM32F205xx, STM32F207xx 2013 datasheet, STMicroelectronics.