IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 7, Issue 02, September 2018)

Area Efficient Design of Five Stage Pipelined RISC Processor

Shahnawaz Khan
M. Tech. Scholar
Embedded System & VLSI Design
Acropolis Institute of Technology & Research,
Indore, M.P. (India)
khanshahnawaz48@gmail.com

Prabhat Pandey
Assistant Professor
Dept. of Electronics & Communication
Acropolis Institute of Technology & Research,
Indore, M.P. (India)
prabhatpandey@acropolis.in

Abstract – **This paper proposes the design of a 32 bit RISC (Reduced Instruction Set Computer) processor. The design helps to improve the speed of the processor and to give the higher performance of the processor. It has 5 stages of pipeline viz. instruction fetch, instruction decode, instruction execute, memory access and write back all in one clock cycle. The control unit controls the operations performed in these stages. All the modules in the design are coded in VHDL.**

Keywords – **CISC, CPU, MIPS, RISC, VHDL.**

I. INTRODUCTION

There are several drawbacks & deficient effects of earlier design of Microprocessors, earlier designs was proposed in which finite state machines has been taken for controller but low power issues are not considered for encoding and cyclic controlling. If the encoding is grey, then it will consume less power, other issues of previous designs are there low speed and there controlling mechanism for instruction scheduling. In today's era of high speed systems and ubiquitous computing, the need for real-time embedded systems is always on the rise. These embedded systems must operate within stringent requirements that are often at the intersection of the conflict between speed and area. Increasing complexity of signal, image and control processing in embedded real-time applications requires very high computational power [1]. This power can be achieved by high performance programmable components like RISC or CISC processors etc. and non-programmable specific chips like ASICs or FPGA based hardware. Naturally, to enhance the speed of such systems we need to design algorithms that can computed with low running time complexity. Another way of increasing the speed of the system is to directly design high speed VLSI chips for these embedded systems. However, the present design was concerned more with demonstrating the power of multi-stage pipelining for fast instruction execution and also to demonstrate that FPGA based processors can be equally useful for real-time operating systems rather than to design just a single processor[2].

CISC processors have gained the common marketplace over the years. They support various addressing modes and various data types. The instruction length varies from instruction to instruction. They frequently access data in external memory. They are generally implemented using micro-programmed control. There is little semantic gap between instructions of CISC processor and statements in higher-level languages. Although they may be saving memory space, design is complicated and as instructions is variable in length; special hardware for boundary marking of instruction is required. Study over the years proved that simple instructions are used 80% of the time and many complex instructions can be replaced by group of simple instructions [3].

RISC processor operates on very few data types and does the simple operations. It supports very few addressing modes and is mostly register based. Most of the instructions operate on data present in internal registers. Only LOAD and STORE instructions access data in external memory. Also the instruction length is fixed and hence decoding is easier [4-5].

Parallel execution of instructions through the pipelined stages of processor improves the overall throughput of the processor but introduces some hazards in its working. Data hazards are due to sharing of destination and source resources in succeeding instructions (source for an instruction is destination for previous instruction) and they can be resolved by the method of forwarding. Structural hazards are due to common program and data memory. By implementing the pre-fetch queen in processor, structural hazards can be handled. Control hazards are introduced in non-sequential execution of an instruction and the method of flushing is used to deal with them [6-8].

The main objective for this paper is to implement a 5 stage pipelined architecture of RISC (Reduced Instruction Set Computer) CPU based on MIPS

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 7, Issue 02, September 2018)

(Microprocessor Interlock Pipeline Stages) using VHDL.

II. PROPOSED METHOD

A. Instruction Fetch Unit

The first stage in the pipeline is the Instruction Fetch. Instructions are fetched from the memory and the Instruction Pointer (IP) is updated. The function of the instruction fetch unit is to obtain an instruction from the instruction memory using the current value of the PC and increment the PC value for the next instruction as shown in Figure. This stage is where a program counter will pull the next instruction from the correct location in program memory. In addition the program counter will updated with either the next instruction location sequentially, or the instruction location as determined by a branch.

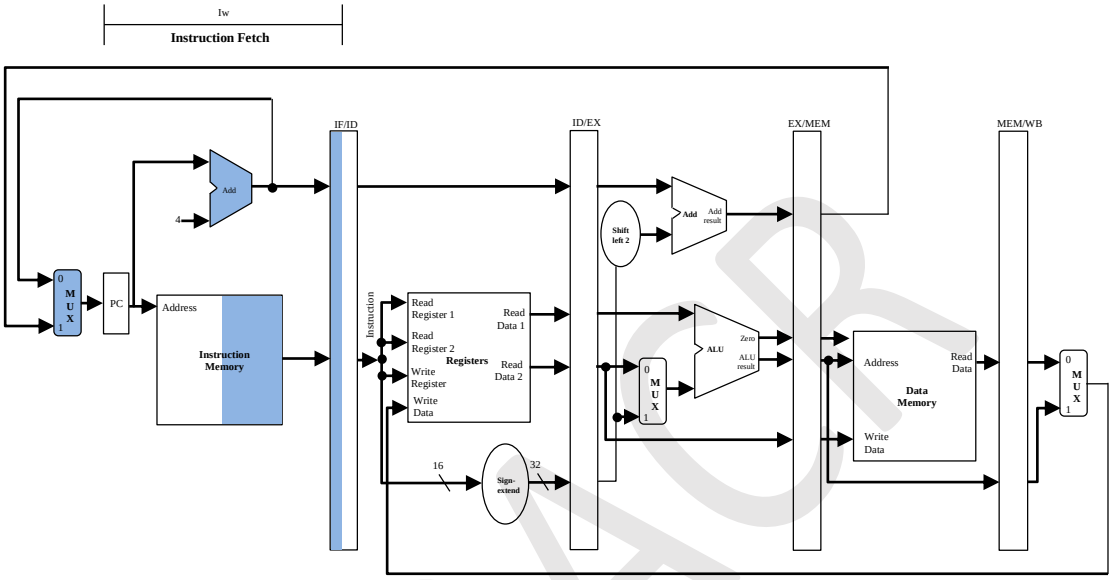

Figure 1: Instruction fetch unit in 5 stage pipelined RISC processor

The instruction fetch stage is also responsible for reading the instruction memory and sending the current instruction to the next stage in the pipeline, or a stall if a branch has been detected in order to avoid incorrect execution. The instruction fetch unit contains the following logic elements that are implemented in VHDL: 8-bit program counter (PC) register, an adder to increment the PC by four, the instruction memory, a multiplexor, and an AND gate used to select the value of the next PC. Program counter and instruction memory are the two important blocks of Instructions Fetch Unit [9].

1) Program Counters (PC)

It is an 8 bit device that is connected to the data bus and the address bus. It will hold its value unless told to do something. If the I/P is kept high the device will count, i.e. it will increment by 4.

2) Instruction Memory (IM)

The Instruction memory on these machines had a latency of one cycle. During the Instruction Fetch stage, a 32-bit instruction is fetched from the memory.

The PC predictor sends the Program Counter (PC) to the Instruction memory to read the current Instruction. At the same time, the PC predictor predicts the address of the next instruction by incrementing the PC by 4.

3) Instruction Registers (IR)

An instruction register (IR) is the part of control unit that stores the instruction currently being executed or decoded. In simple processors each instruction to be executed is loaded into the instruction register which holds it while it is decoded, prepared and ultimately executed, which can take several steps. RISC processors use a pipeline of instruction registers where each stage of the pipeline does part of the decoding, preparation or execution and then passes it to the next stage for its step. Modern processors can even do some of the steps of out of order as decoding on several instructions is done in parallel. Decoding the opcode in the instruction register includes determining the instruction, where its operands are in memory, retrieving the operands from memory, allocating processor resources to execute the command. The output of IR is available to control circuits which generate the timing signals that controls the various processing elements involved in executing the instruction.

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 7, Issue 02, September 2018)

B. Instruction Decode Unit

The Instruction Decode stage is the second stage in the pipeline. Branch targets will be calculated here and the Register File, the dual-port memory containing the register values, resides in this stage. The forwarding units, solving the data hazards in the pipeline, reside here. Their function is to detect if the register to be fetched in this stage is written to in a later stage. In that case the data is forward to this stage and the data hazard is solved. This stage is where the control unit determines what values the control lines must be set to depending on the instruction. In addition, hazard detection is implemented in this stage, and all necessary values are fetched from the register banks. The Decode Stage is the stage of the CPU's pipeline where the fetched instruction is decoded, and values are fetched from the register bank. It is responsible for mapping the different sections of the instruction into their proper representations (based on R or I type instructions) [10].

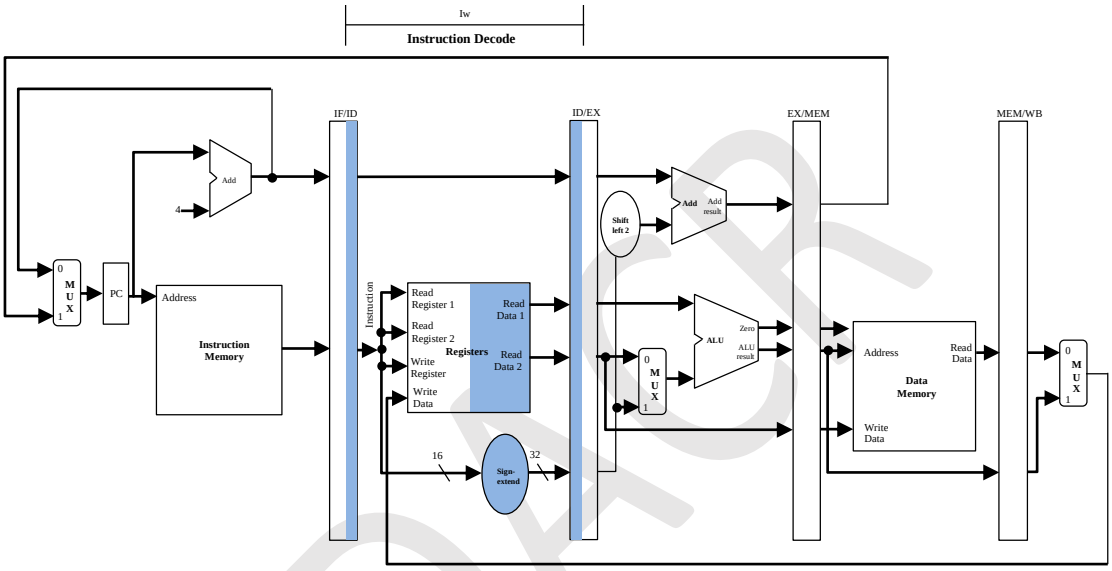

Figure 2: Instruction decode unit in 5 stage pipelined RISC processor

The Decode stage consists of the Control unit, the Hazard Detection Unit, the Sign Extender, and the Register bank, and is responsible for connecting all of these components together.

It splits the instruction into its various parts and feeds them to the corresponding components. Registers ***Rs*** and ***RT*** are fed to the register bank, the immediate section is fed to the sign extender, and the ALU opcode and function codes are sent to the control unit. The outputs of these corresponding components are then clocked and stored for the next stage The Control unit takes the given Opcode, as well as the function code from the instruction, and translates it to the individual instruction control lines needed by the three remaining stages. This is accomplished via a large case statement.

1) Control Unit

The control unit of the MIPS single-cycle processor examines the instruction opcode bits [31 – 26] and decodes the instruction to generate nine control signals to be used in the additional modules. The ***Reg Dst*** control signal determines which register is written to the register file. The Jump control signal selects the jump address to be sent to the PC. The Branch control signal is used to select the branch address to be sent to the PC. The ***MemRead*** control signal is asserted during a load instruction when the data memory is read to load a register with its memory contents. The ***MemtoReg*** control signal determines if the ALU result or the data memory output is written to the register file. The ***ALUOp*** control signals determine the function the ALU performs. (E.g. and, or, add, sbu, slt) The ***MemWrite*** control signal is asserted when during a store instruction when a registers value is stored in the data memory.

2) The ALUSrc Control Signal

It determines if the ALU second operand comes from the register file or the sign extend. The ***RegWrite*** control signal is asserted when the register file needs to be written.

3) Register Files (RF)

During the decode stage, the two register ***Rs*** & ***Rt*** are identified within the instruction, and the two registers are read from the register file. In the MIPS design, the register file had 32 entries. At the same time the register file was read, instruction issue logic in this stage determined if the pipeline was ready to

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 7, Issue 02, September 2018)

execute the instruction in this stage. If not, the issue logic would cause both the Instruction Fetch stage and the Decode stage to stall. If the instruction decoded was a branch or jump, the target address of the branch or jump was computed in parallel with reading the register file.

The branch condition is computed after the register file is read, and if the branch is taken or if the instruction is a jump; the PC predictor in the first stage is assigned the branch target, rather than the incremented PC that has been computed.

C. Execution Unit

The third stage in the pipeline is where the arithmetic- and logic-instructions will be executed. All instructions are executed with 32-bit operands and the result is a 32-bit word. An overflow event handler was not included in this project. The execution unit of the MIPS processor contains the arithmetic logic unit (ALU) which performs the operation determined by the ***ALUop*** signal. The branch address is calculated by adding the PC+4 to the sign extended immediate field shifted left 2 bits by a separate adder.

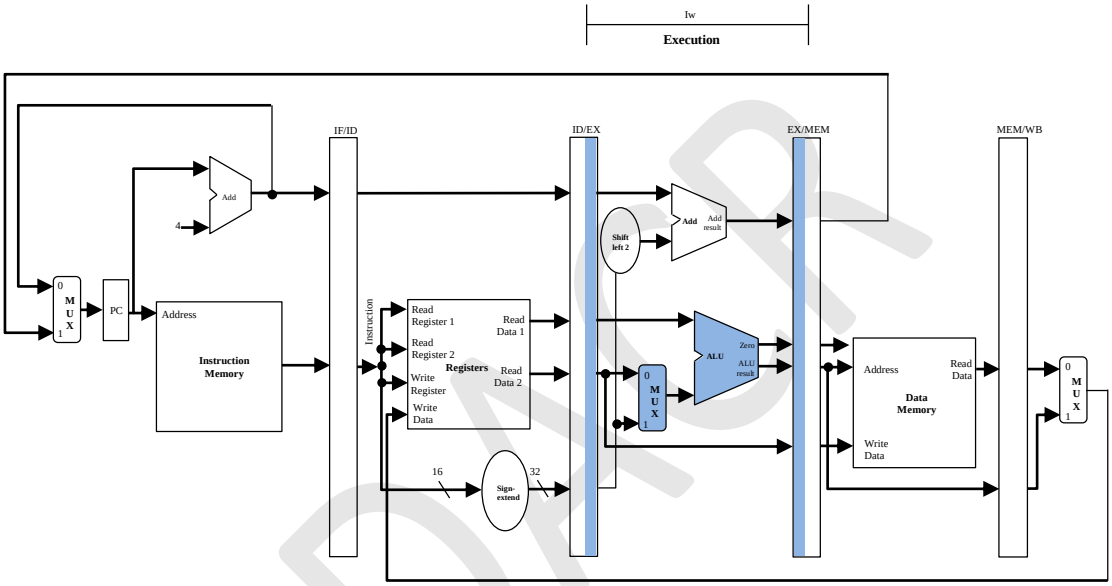

Figure 3: Execution unit in 5 stage pipelined RISC processor

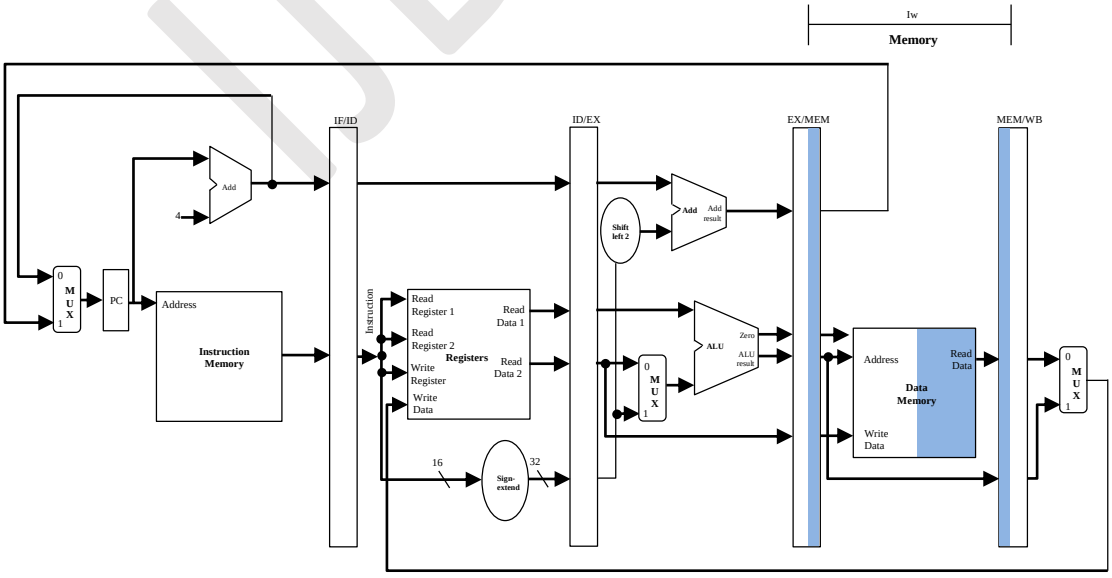

Figure 4: Memory access unit in 5 stage pipelined RISC processor

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 7, Issue 02, September 2018)

The logic elements to be implemented in VHDL includes:

1) ALU Unit

The arithmetic/logic unit (ALU) executes all arithmetic and logical operations. The arithmetic/logic unit can perform four kinds of arithmetic operations, or mathematical calculations: addition, subtraction, multiplication, and division. As its name implies, the arithmetic/logic unit also performs logical operations. A logical operation is usually a comparison. The unit can compare numbers, letters, or special characters. The computer can then take action based on the result of the comparison. This is a very important capability

D. Memory Access Unit

The memory access stage is the fourth stage of pipeline. This is where load and store instructions will access data memory. During this stage, single cycle latency instructions simply have their results forwarded to the next stage. This forwarding ensures that both single and two cycle instructions always write their results in the same stage of the pipeline, so that just one write port to the register file can be used, and it is always Available. If the instruction is a load, the data is read from the data memory

1) Data Memory Unit (DM)

The data memory unit is only accessed by the load and store instructions. The load instruction asserts the ***MemRead*** signal and uses the ALU Result value as an address to index the data memory. The read output data is then subsequently written into the register file. A store instruction asserts the ***MemWrite*** signal and writes the data value previously read from a register into the computed memory address. The VHDL implementation of the data memory was described earlier.

E. Write Back Unit

During this stage, both single cycle and two cycle instructions write their results into the register file. Figure 5 shows the write back operation.

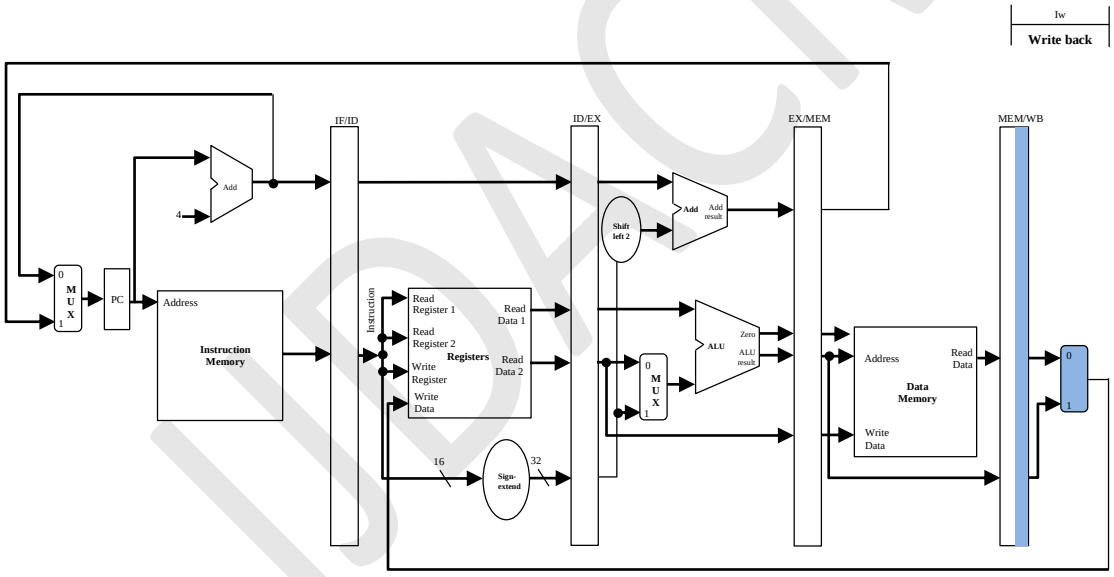

Figure 5: Write back unit in 5 stage pipelined RISC processor

F. RISC Processor Implementation

The RISC single-cycle processor performs the tasks of instruction fetch, instruction decode, execute, memory access and write-back all in one clock cycle. First the PC value is used as an address to index the instruction memory which supplies a 32-bit value of the next instruction to be executed. This instruction is then divided into the different fields. The instructions ***opcode*** field bits [31–26] are sent to a control unit to determine the type of instruction to execute. The type of instruction then determines which control signals are to be asserted and what function the ALU is to perform, thus decoding the instruction. The instruction register address fields ***rs*** bits [25–21], ***rt*** bits [20–16], and ***rd*** bits [15–11] are used to address the register file. The register file supports two independent register reads and one register write in one clock cycle. The register file reads in the requested addresses and outputs the data values contained in these registers. These data values can then be operated on by the ALU whose operation is determined by the control unit to either compute a memory address (e.g. load or store), compute an arithmetic result (e.g. add, sub), or perform a compare (e.g. branch). If the instruction decoded is arithmetic, the ALU result must be written to a register. If the instruction decoded is a load or a store, the ALU result is then used to address

IJDACR
ISSN: 2319-4863

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 7, Issue 02, September 2018)

the data memory. The final step writes the ALU result or memory value back to the register file.
Once the RISC single-cycle VHDL implementation is completed, our next task is to pipeline the RISC processor. Pipelining, a standard feature in RISC processors, is a technique used to improve both clock speed and overall performance. 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 RISC single-cycle implementation, the data path 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 RISC 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.

III. Simulation And Results

A. Results

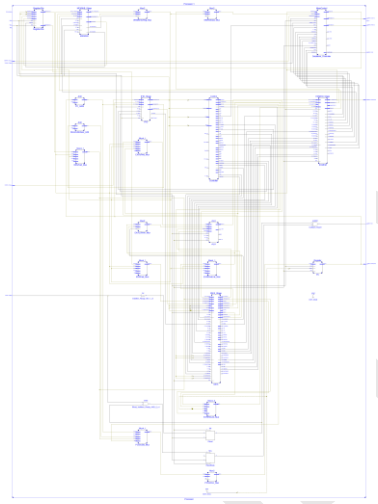

Figure 6: RTL Schematic for RISC top entity

Simulation results for 5 stage pipelined architecture of RISC has been shown in Figure 7. Simulation is carried out on Modelsim 6.5e simulator.

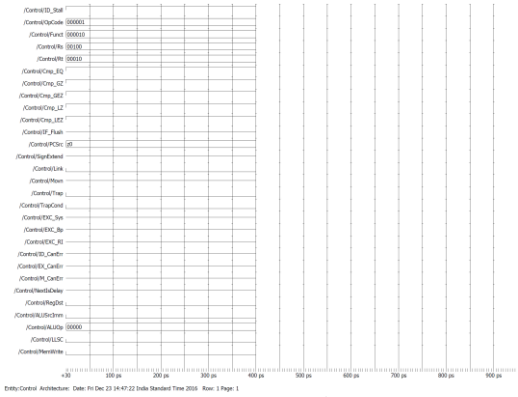

Figure 7: Simulation result of control block

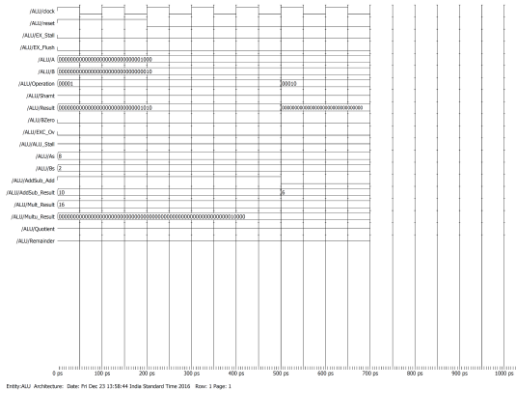

Figure 8: Simulation result of ALU block

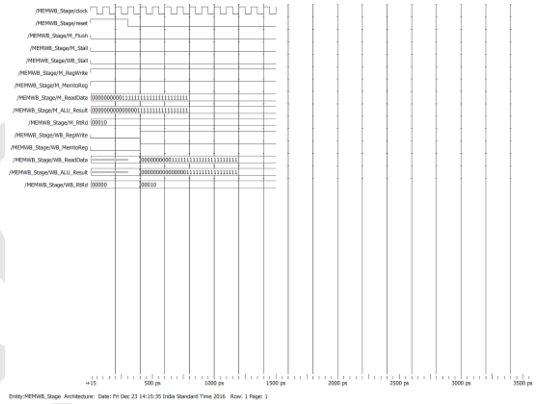

Figure 9: Simulation result of MEMWB block

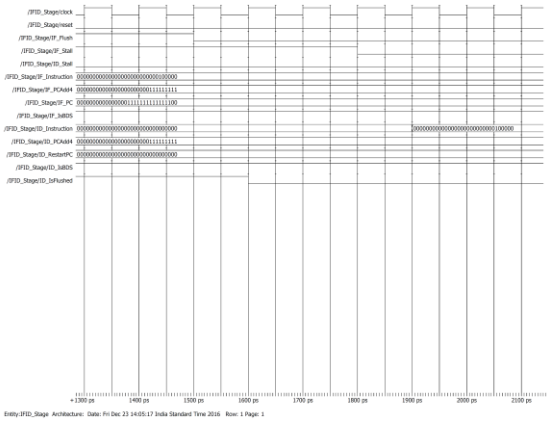

Figure 10: Simulation result of IFID block

IJDACR
International Journal Of Digital Application & Contemporary Research

International Journal of Digital Application & Contemporary Research
Website: www.ijdacr.com (Volume 7, Issue 02, September 2018)

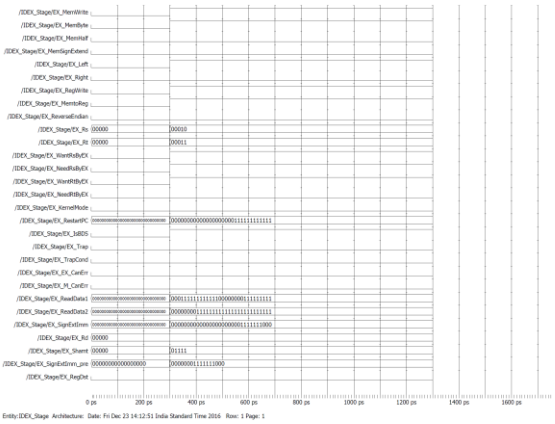

Figure 11: Simulation result of IDEX block

B. Device Utilization Summary

Selected Device: 6slx100tfgg900-3

Slice Logic Utilization

Number of Slice Registers: 1273 out of 126576, 1%
Number of Slice LUTs: 2886 out of 63288, 4%
Number used as Logic: 2886 out of 63288, 4%

Slice Logic Distribution

Number of LUT Flip Flop pairs used: 3011
Number with an unused Flip Flop: 1738 out of 3011, 57%
Number with an unused LUT: 125 out of 3011, 4%
Number of fully used LUT-FF pairs: 1148 out of 3011, 38%
Number of unique control sets: 4

IO Utilization

Number of IOs: 180
Number of bonded IOBs: 133 out of 498, 26%

Specific Feature Utilization

Number of BUFG/BUFGCTRL/BUFHCEs: 1 out of 16, 6%
Number of DSP48A1s: 3 out of 180, 1%

Timing Summary

Speed Grade: -3
Minimum period: 14.222ns (Maximum Frequency: 70.312MHz)
Minimum input arrival time before clock: 3.924ns
Maximum output required time after clock: 4.521ns

IV. Conclusion

The 5 stage pipelined architecture of 32 bit RISC Processor (MIPS) has been designed using VHDL. The synthesis is performed on Xilinx ISE 14.5 using the device 6slx100tfgg900-3 and simulations are done with ModelSim simulator. The simulation results show that the processor works perfectly. A Reduced Instruction Set computer is a microprocessor that had been designed to perform a small set of instructions, with the aim of increasing the overall speed of the processor. The RISC architecture follows the philosophy that one instruction executes in one clock cycle.

Reference

[1] Pranjali S. Kelgaonkar, Shilpa Kodgire, "Pipelined 32bit RISC MIPS Processor on Spartan-6 FPGA", International Journal of Science, Engineering and Technology Research (IJSETR), ISSN: 2278-7798, Vol. 5, Issue 8, August 2016.

[2] P. Ajith Kumar, M. Vijaya Lakshmi, "Design of a Pipelined 32 Bit MIPS Processor with Floating Point Unit", International Journal of Innovative Research in Science, Engineering and Technology, ISSN: 2319-8753, Vol. 5, Issue 7, July 2016.

[3] S. P. Ritpurkar, M. N. Thakare, G. D. Korde, "Design and Simulation of 32-Bit RISC Architecture based on MIPS using VHDL", IEEE, International Conference on Advanced Computing and Communication Systems (ICACCS -2015), Coimbatore, INDIA, January 2015.

[4] Topiwala, Mohit N., and N. Saraswathi, "Implementation of a 32-bit MIPS Based RISC Processor using Cadence", IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), pp. 979-983, May 2014.

[5] Sharda P. Katke, G.P. Jain, "Design and Implementation of 5 Stages Pipelined Architecture in 32 Bit RISC Processor", International Journal of Emerging Technology and Advanced Engineering, ISSN: 2250-2459, Vol. 2, Issue 4, April 2012.

[6] Samiappa Sakthikumaran, S. Salivahanan, V. S. Kanchana Bhaaskaran, "16-Bit RISC Processor Design for Convolution Application" IEEE-International Conference on Recent Trends in Information Technology, ICRTIT 2011.

[7] Robert S. Plachno, VP of Audio, "A True Single Cycle RISC Processor without Pipelining", ESS Design White Paper – RISC Embedded Controller.

[8] V. B. Saambhavi and V. S. Kanchana Bhaaskaran, "A 16-BIT RISC MICROPROCESSOR USING DCPAL CIRCUITS", International Journal of Advanced Engineering Technology (IJAET), Vol. 2, no. 1, pp.1-9, 2011.

[9] Samiappa Sakthikumaran et al., "A Very Fast and Low Power Incrementer and Decrementer Circuits", International Journal of Computer Communication and Information System (IJCCIS) Vol2. No.1 – 2011, pp. 200-203.

[10] Padma Devi, Ashima Girdher, Balwinder Singh, "Improved Carry Select Adder with Reduced Area and Low Power Consumption", International Journal of Computer Applications (0975 – 8887) Volume 3 – No.4, June 2010.