International Journal of Electrical, Electronics and Data Communication, ISSN: 2320-2084 Volume-5, Issue-6, Jun-2017
http://iraj.in

MICRO-ARCHETECTURE DESIGN OF RISC V MICROPROCESSOR USING VHDL

VINAY REDDY NARAYANA

Electronics and Communication Engineering Department, Assistant Professor, PESIT-BSC
E-mail: vinayreddynarayana@gmail.com

Abstract: The freedom of compatibility with old designs and the use of microprocessor technology led to a renaissance in computer design which emphasized both architectural innovation and efficient use of technology improvements. The paper proposes a micro-architecture design of a 32 bit RISC V microprocessor, the proposed micro-architecture is meritorious compared to the earlier versions since the RISC-V processor ISA keeps the source (rs1 and rs2) and destination (rd) registers at the same position in all formats to simplify decoding, also immediates are packed towards the leftmost available bits in the instruction and have been allocated to reduce hardware complexity. In particular, the sign bit for all immediates is always in bit 31 of the instruction to speed sign-extension circuitry. The prominent features which make RISC V better than other ISAs are RISC-V ISA is very easy to decode and all instructions in it are easy to schedule and do hazard checking.

Optimization is done by pipelining the design in order to improve the processor Clock cycle Per Instruction, however most of the time there are data dependencies which will increase the CPI. In order to improve the CPI, a data forwarding unit is incorporated in the execution stage to solve the same. The Instruction set architecture used for the design is the RISC - V *(Version – 2.0000)* from the University of Berkeley (UCB). The entire project phase comprises micro-architecture design of RISC – V ISA (Integer computational and control transfer instructions) processor and the elimination of the structural hazard and the data hazards. Elimination of the structural hazard is done by designing a separate memory for the instructions and data. Data hazards are eliminated by the design of the forwarding unit which overcomes the problem of data dependencies.

Keywords: CPI, ISA, RISC V

I. INTRODUCTION

The proposed paper is in the aspect of micro-architecture design of an RISC V processor with the prime motive to build a System on Chip suitable for embedded microprocessor system. One of the major challenges in any processor design is the reduction of the CPI (Clock cycle Per Instruction). Hence the objective serves to have –

- Design of the RISC V processor
- Pipelining of the processor to reduce CPI
- Elimination of the pipeline hazards

The architecture chosen in the project is a RISC architecture, implementing a processor with a simplified instruction set design provides several advantages over implementing a comparable CISC design:

- **Speed**. Since a simplified instruction set allows for a pipelined, superscalar design RISC processors often achieve 2 to 4 times the performance of CISC processors using comparable semiconductor technology and the same clock rates.
- **Simpler hardware**. Because the instruction set of a RISC processor is so simple, it uses up much less chip space; extra functions, such as memory management units or floating point arithmetic units, can also be placed on the same chip. Smaller chips allow a semiconductor manufacturer to place more parts on a single silicon wafer, which can lower the per-chip cost dramatically.

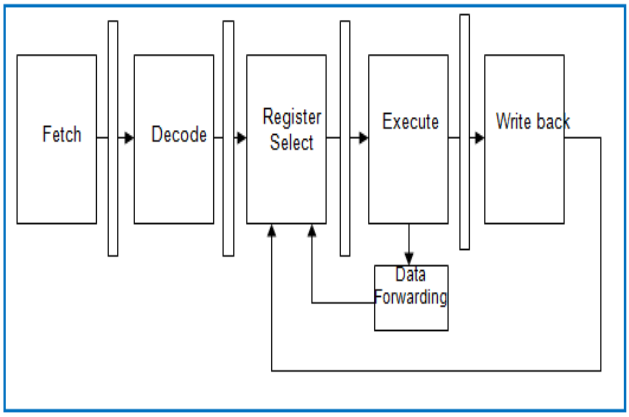

Figure 1: Abstract model of the 5 stage RISC V processor

II. INSTRUCTION SET FOR THE RISC V PROCESSOR DESIGN

The RISC-V ISA is defined as a base integer ISA which is very similar to that of the early RISC processors except with no branch delay slots and with support for optional variable-length instruction encodings. The base is carefully restricted to a minimal set of instructions sufficient to provide a reasonable target for compilers, assemblers, linkers, and operating systems. Each base integer instruction set is characterized by the width of the integer registers and the corresponding size of the user address space. This project focuses on the base

integer variant - RV32I i.e., 32 bit instruction registers.

III. PROPOSED ARCHITECTURE

Instruction type : R type instruction	
1.	ADD rd, rs1, rs2
2.	SUB rd, rs1, rs2
3.	AND rd, rs1, rs2
4.	OR rd, rs1, rs2
5.	XOR rd, rs1, rs2
6.	SLL rd, rs1, rs2
7.	SRL rd, rs1, rs2
8.	SRA rd, rs1, rs2
9.	SLT rd, rs1, rs2
10.	SLTU rd, rs1, rs2
Instruction type : I type instruction	
11.	ADDI rd, rs1, imm
12.	ANDI rd, rs1, imm
13.	ORI rd, rs1, imm
14.	XORI rd, rs1, imm
15.	SLLI rd, rs1, imm
16.	SRLI rd, rs1, imm
17.	SRAI rd, rs1, imm
18.	SLTI rd, rs1, imm
19.	SLTUI rd, rs1, imm
Instruction type : Unconditional branch, U (LUI) type instruction	
20.	LUI rd, imm
Instruction type : Unconditional branch, U (AUIPC) type instruction	
21.	AUIPC rd, imm
Instruction type : Unconditional branch, UJ (JAL) type instruction	
22.	JAL rd, imm
Instruction type : Unconditional branch, I (JALR) type instruction	
23.	JALR rd, rs1, imm
Instruction type : Conditional branch, SB type instruction	
24.	BEQ rd, rs1, imm
25.	BNE rd, rs1, imm
26.	BLT rd, rs1, imm
27.	BGE rd, rs1, imm
28.	BLTU rd, rs1, imm
29.	BGEU rd, rs1, imm

The RISC V processor is a pipelined design which comprises of five stages, the chapter covers all the five stages in detail. Finally the overall integrated module is explained, the five stages of the designed RISC V processor are

- Fetch stage
- Decode stage
- Register select stage
- Execute stage
- Write back stage

4.1 Fetch Stage Design

Program Counter: The PC is designed using a simple D FF and the purpose of the PC is to store the address of the current instruction. The register used for the PC is a 32 bit register which works on the rising edge of the clock.

PC adder: This adder is a 32 bit adder which is used to increment the current address by 4, it is a combinational design which adds 4 to the current address and gives it to the PC as the next address of the instruction.

Instruction Memory: The instruction memory is designed to hold the instructions of the program, also referred as the program memory is a ROM memory. The size of the memory is 2^{32} X 2^{32} bits in size. The enable signal is made high in order to read the instruction from the memory.

Address selector: The address selector is a 2:1 multiplexer which is used to select if the next address would be a branch address or the current address incremented by 4. The select signal to the multiplexer comes from the execute module once the branch address is calculated by the adder present in the execute unit.

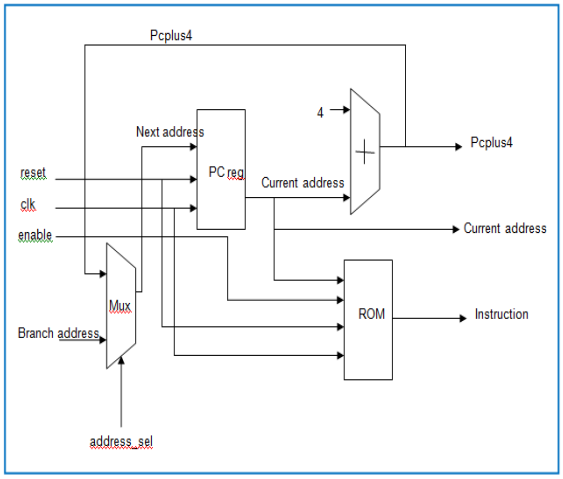

Figure 2: Fetch stage of the RISC V Processor design

4.2 Decode Stage Design

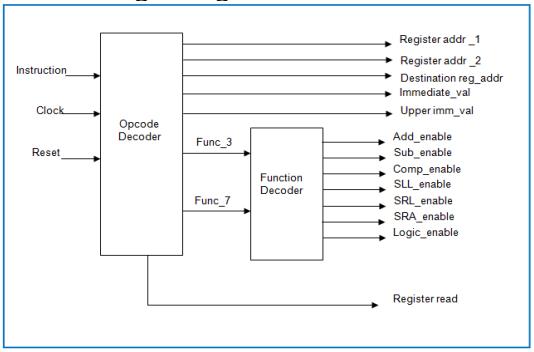

Figure3: Decode stage of the RISC V Processor design

Opcode decoding: The 7 bits of the LSB in an instruction represent the Opcode field, the decoder firsts decodes the Opcode to find out the type of instruction. The 2 LSB bits indicates the base of the instruction i.e., the base is 32 bit instruction format

hence it is coded as "11" for all the instructions indicating that it is a 32 bit format.

Function decoding: the function field is decoded next in order to enable the corresponding module in the execute stage to perform the operation. The function field is 3 bits while in some instructions it is necessary to check the 7 bit function field once the 3 bit function field is decoded. Also the register read signal is made high in order to read the operands from the data memory block.

4.3 Register Select Stage Design

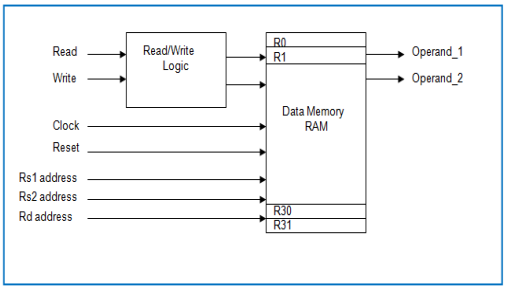

Figure 4: Register select stage of the RISC V Processor design

4.4 Sign extension and Zero padding module

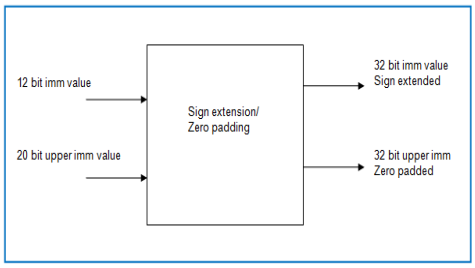

Figure 5: Sign extension/zero padding module of the RISC V Processor design

The 12 bit immediate offset is sign extended to 32 bit immediate value using the sign extension logic. Similarly the 20 bit upper immediate value is converted to a 32 bit value by padding it with the zeros to the LSB side of the 20 bit value. The upper immediate values are usually used in the control flow type of instructions.

The sign extension/zero padding module is a combinational module which works in parallel with the register select module. This module takes the immediate values once the instruction is being decoded and converts them to the compatible 32 bit value which can later be used by the execution unit for further processing.

4.5 Execute Stage Design

The execute module designed is a distributed type of design where separate modules are present for each instruction (similar type uses the same module). An adder is present for normal 32 bit addition, similarly a subtractor for subtraction. The logic instructions are computed using the AND, OR, XOR gates in the logic block while the shifter is present for the shifting operations.

A comparator is available in order to compare the two operands and to say whether the branch has to be taken or not taken based on the condition. The set less than instructions are used to set the value of the destination register as '0' or '1' based on the comparison of operand values whose logic is determined by a separate module.

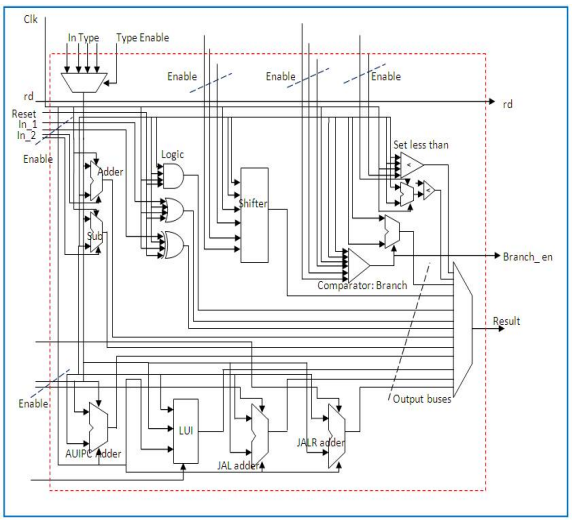

Figure 6: Execute stage of the RISC V Processor design

An adder is constructed in order to compute the AUIPC instruction where one input to the adder is the PC value while another input is the 20 bit upper immediate value which is zero padded to a 32 bit value. The added result is then taken and the lower 12 bits are cleared. Similarly a separate adder is used for the JAL and the JALR instructions, JAL instruction adds the sign extended immediate offset to the PC value to form the target address while the destination register is stored with the next address of the program. In the case of JALR instruction the adder adds the sign extended immediate offset to the PC value and then sets the LSB of the result to '0' while the address of the instruction following the pcplus4 is written into the destination register. The LUI instruction is used to build a 32 bit constant and store it into the destination register.

A multiplexer is added at the beginning in order to select the type of input to the execute module i.e., the input would be an immediate value or a value fetched from the registers. Also a multiplexer is needed to route the result to the output bus of the execute module.

4.6 Write Back Stage Design

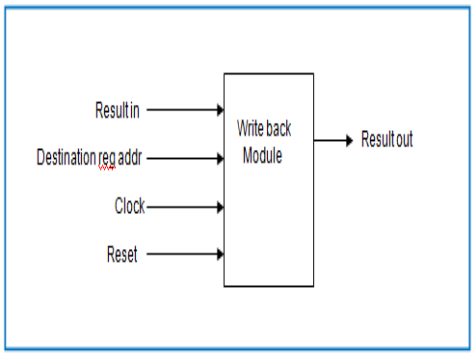

Figure 7: Write back stage of the RISC V Processor design

4.7 Micro-architecture design of the processor

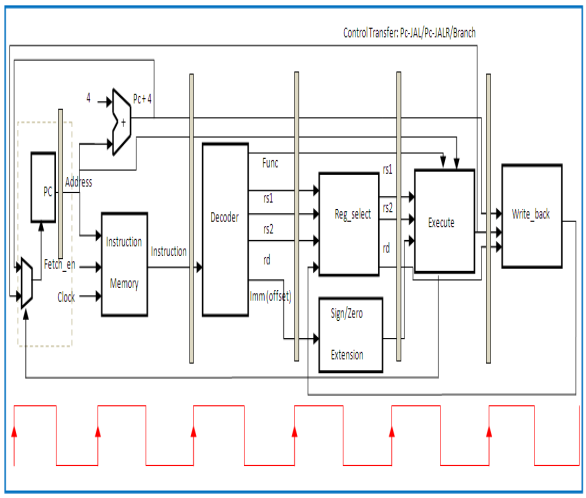

Figure 8: Micro-architecture design of the RISC V Processor design

All the stages discussed so far are integrated together to form the overall design of the processor. All the five stages fetch stage, decode stage, register select stage, execute stage and the write back stage are integrated following which pipelining is performed in order to reduce the CPI of the processor.

Registers are introduced in between each stage in order to achieve pipelining in the design. There is a latency of one clock cycle in the fetch stage for fetching the instruction while there is a latency of one clock cycle in the decode stage too. Similarly there is a latency of one clock cycle in each of the stages of the design finally contributing to a latency of number of stages in the design. However there is latency, initially the processor speeds up as the instructions are fetched and processed simultaneously in the following instructions.

Figure 8 shows the top module of the design i.e., processor design with the pipeline registers. The figure depicts the entire architecture without the addition of the hazard detection cases.

V. RISC V PROCESSOR WITH THE ELIMINATION OF HAZARDS

The processor designed is a five stage pipelined system. Pipelining improves the CPI of the machine but results in the hazards which in our design are considered and eliminated. The structural hazard is eliminated by having separate memories for the data and the instructions. Considering the data hazards, only the RAW hazards occur in the design since it is a simple pipeline design and this hazard is eliminated by incorporating the data forwarding unit in the execute stage.

Pipeline hazards and the solutions to overcome the structural and data pipeline hazards are also covered in this chapter. The next chapter gives the results and discussion to support the micro-architecture design of the RISC V processor.

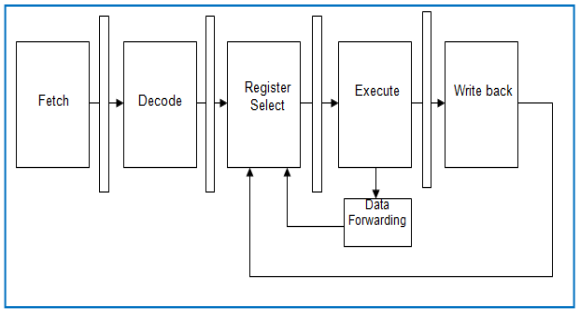

Figure 9: RISC V design with forwarding unit.

VI. RESULTS AND DISCUSSION

The specification for the design is gathered from the RISC V ISA manual (Version 2.0000) and coded using the VHDL language. The design need to be simulated and synthesized once the VHDL coding is written, the tools used for this purpose are –

- Modelsim *(Version – 6.3g)* – Simulation
- Xilinx ISE Design Suit *(Version 14.2)*, Project Navigator – Synthesize

Timing simulation is performed by loading the following libraries –

- Unisim library
- Simprim library

Once the synthesis is done then post PAR and post synthesis simulation is performed, the netlist is collected from the netgen folder and added to the design in the Modelsim tool. The timing simulation is performed along with the testbench code, top module design, libraries and the netlist. Then the golden waveforms are compared with the resulted waveforms.

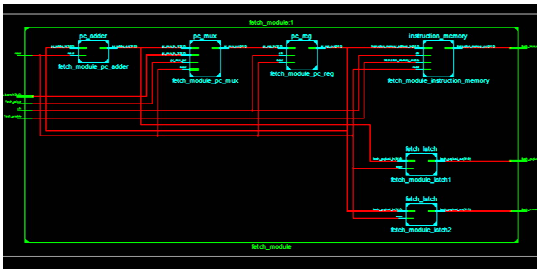

Figure 10: RTL of the fetch module

The fetch stage of the RISC V processor comprises of four modules - Program counter, PC adder, Instruction memory and Address selector. Two latches are introduced in the design to latch the output of the PC adder and the Program counter respectively, figure 10 shows the RTL of the fetch module.

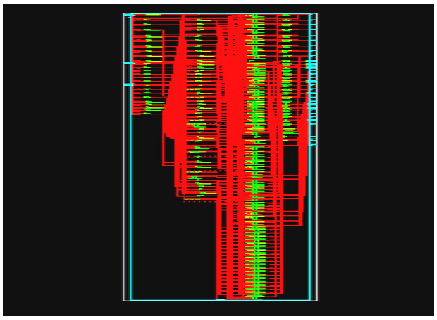

Figure 11: RTL of the decode module

The decoder is responsible to split the instruction i.e., decode to interpret the type of instruction and then the respective enable signals that has to be enabled for the processing of the instruction. Figure 11 shows the RTL of the fetch module, the decoder performs the decoding as –

- Opcode decoding
- Function decoding

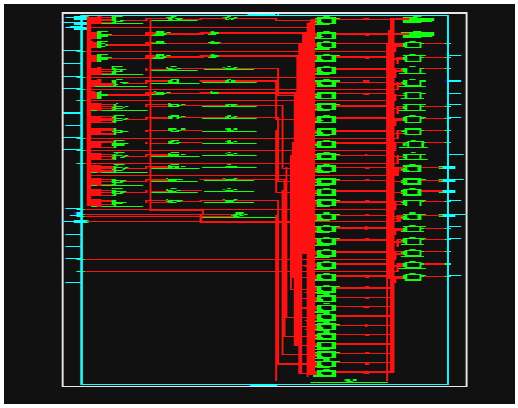

Figure 12: RTL of the register select module

The register select stage is also known as the data memory, usually comprises of 32 registers each 32 bit in length. Figure 12 shows the RTL of the register select module.

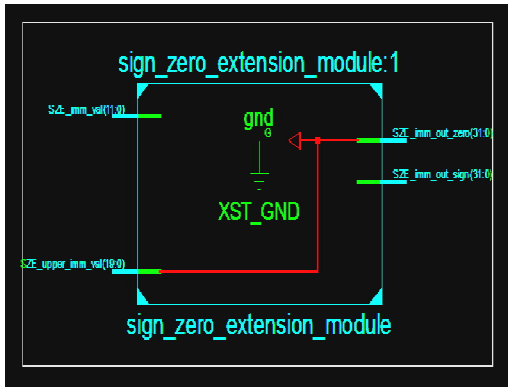

Figure 13: RTL of the sign extension/zero padding module

The sign extension/zero padding module is a combinational module which works in parallel with the register select module. This module takes the immediate values once the instruction is being decoded and converts them to the compatible 32 bit value which can later be used by the execution unit for further processing. Figure 13 shows the RTL of the sign extension/zero padding module.

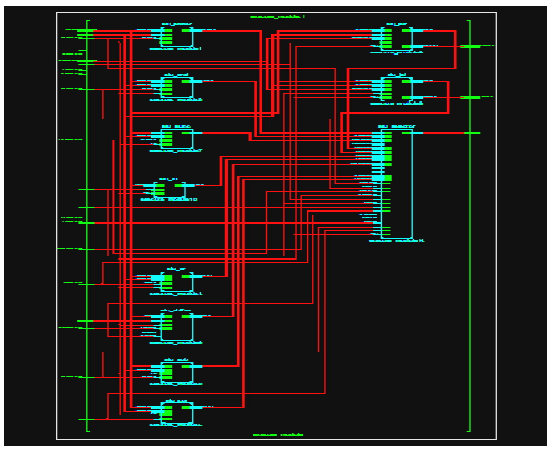

Figure 14 : RTL of the execute module

The execute module designed is a distributed type of design where separate modules are present for each instruction (similar type uses the same module). The execute stage of the processor is responsible for all the computations. The ALU designed for the RISC V processor is a single cycle processor, which later is made as a combinational one in order to overcome the data hazards problem encountered due to instruction dependencies. Figure 14 shows the RTL of the execute module.

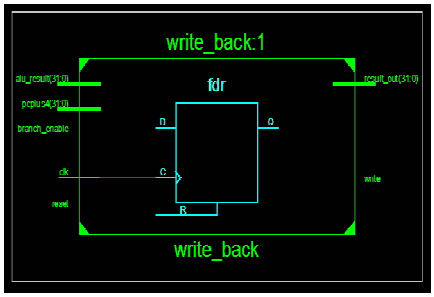

Figure 15: RTL of the write back module

The write back stage design is a register which collects the result from the execute stage and then writes it into the destination register. This stage functions to store the result into the register. Figure 15 shows the RTL of the write back module.

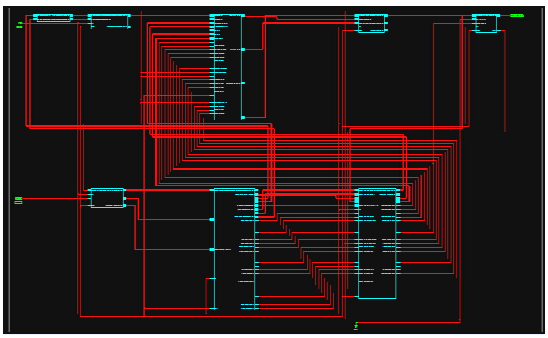

Figure 16: RTL of the RISC V Processor

All the five stages fetch stage, decode stage, register select stage, execute stage and the write back stage are integrated following which pipelining is performed in order to reduce the CPI of the processor. Also data forwarding unit is incorporated to overcome the data hazards. Figure 16 shows the RTL of the RISC processor top module.

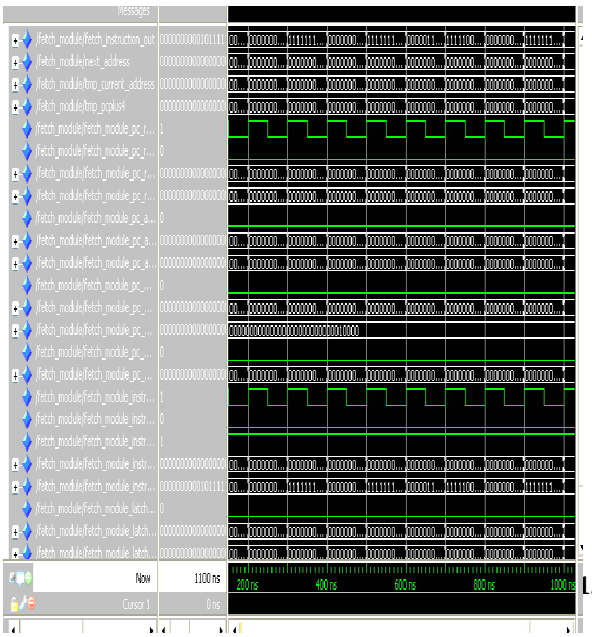

Figure 17: Simulation waveform of RISC processor

To evaluate the performance of the RISC V processor, the proposed design is compared with the MIPS processor [3]. The timing summary table clearly proves that the RISC V processor performance is better than the MIPS processor architecture.

Table 1: Device Utilization Summary of MIPS Vs RISC V processor

	MIPS PROCESSOR Existing	RISC V PROCESSOR Proposed
Speed Grade	-5	-5
Maximum Period	12.094 ns	7.266 ns
Maximum Frequency	82.688 MHz	137.636 MHz
Maximum input arrival time before clock	6.115 ns	4.583 ns
Maximum output required time after clock	6.678 ns	1.371 ns
Maximum combinational path delay	No path found	2.899 ns

Table 2: Performance Summary of MIPS Vs RISC V processor

	MIPS PROCESSOR Existing	RISC V PROCESSOR Proposed
Number of Slices	456 out of 7680 (5%)	249 out of 7680 (32%)
Number of Slice Flip Flops	290 out of 15360 (1%)	127 out of 15360 (8%)
Number of 4 input LUTs	883 out of 15360 (5%)	465 out of 15360 (30%)
Number of bonded IOBs	10 out of 221 (4%)	36 out of 221 (29%)
Number of GCLKs	1 out of 8 (12%)	1 out of 8 (12%)

It clearly tabulates the synthesis report details of the design. Also it compares the existing MIPS architecture with the RISC V processor architecture and proves the RISC V processor architecture to be efficient compared to the MIPS processor

CONCLUSION

The project proposes a micro-architecture design of a 32 bit RISC V microprocessor; the proposed micro-architecture is meritorious compared to the earlier versions. The prominent features which make RISC V better than other ISAs are RISC-V ISA is very easy to decode and all instructions in it are easy to schedule and do hazard checking. The entire project phase comprises micro-architecture design of RISC – V ISA (Integer computational and control transfer instructions) processor and the elimination of the structural hazard and the data hazards. The design

when targeted on FPGA results in a maximum speed of 137.636 MHz which proves the architecture to be a better one compared to the MIPS architecture in terms of performance.

FUTURE WORK

Pipelining, basic way of obtaining faster processor was inspected throughout this project. Different solution proposals have been stated for problems faced while implementing pipelining. It is clear that the main point causing problems was the dependencies between instructions. These dependencies dependences degrades the instruction throughput and CPI can be greater than one which is optimal solution and this problem was resolved by constituting forwarding (bypass) lines between stages. Structural deficiencies are eliminated by using separate Instruction and Data Memory.

There are many directions in which the work can be extended. There can be a research in the future which can propose a method to measure the orthogonality of ISA (Instruction Set Architecture), which is the primary metric for the effectiveness of pipelining. Another direction is to inspect the effects of using longer pipelines, fetching longer instructions from memory and implementing sequencing and some handling mechanisms for all those circumstances. The micro-architecture design can be extended to build a System on Chip suitable for an embedded microprocessor system.

REFERENCE

[1] Andrew Waterman, Yunsup Lee, David Patterson, Krste Asanovic, The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2:0 - 10^{-4}, CS Division, EECS Department, University of California, Berkeley, January 18, 2014.

[2] Kirat Pal Singh, Shivani Parmar,"Vhdl Implementation of A Mips-32 Pipeline Processor" International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 7 No.11, 2012.

[3] Iro Pantazi – Mytarelli," The history and use of pipelining computer architecture: MIPS pipelining implementation," New York Institute of Technology Old Westbury, New York, 11568.

[4] Computer Architecture, A Quantitative Approach, Hennessy John and Patterson David, 1990, Fourth edition

[5] Mamun Bin Ibne Reaz, Md. Shabiul Islam, Mohd. S. Sulaiman, "A Single Clock Cycle MIPS RISC Processor Design using VHDL",ICSE2002 Proc. 2002, Penang, Malaysia, pp.199- 203, 2002.

[6] Kui YI, Yue-hua DING, "32-bit RISC CPU Based on MIPS Instruction Fetch Module Design", International Joint Conference on Artificial Intelligence, 2009, pp. 754-760.

[7] Dal Poz, Marco Antonio Simon, Cobo, Jose Edinson Aedo, Van Noije, Wilhelmus Adrianus Maria, Zuffo,Marcelo Knorich, "Simple Risc microprocessor core designed for digital settopbox applications", Proceedings of the International Conference on Application Specific Systems, Architectures and Processors, 2000, p 3544.

[8] Y. Takahashi, K. Konta, K. Takahashi, K. Shouno, M. Yokoyama, and M. Mizunuma, Carry propagation free adder/subtracter using adiabatic dynamic CMOS logic circuit technology., IEICE Trans. Fundamentals. vol. E86-A, no. 6, pp. 1437.1444, June 2003.

[9] XiangYunZhu, Ding YueHua, "Instruction Decoder Module Design of 32-bit RISC CPU Based on MIPS"Second International Conference on Genetic and Evoltionary Computing,WGEC pp.347-351 Sept.2008

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

[11] Mamun Bin Ibne Reaz, Md. Shabiul Islam, Mohd. S. Sulaiman, "A Single Clock Cycle MIPS RISC Processor Design using VHDL", IEEE International Conference on Semiconductor Electronics, pp.199-203, Dec. 2003

[12] MIPS Technologies, MIPS32™ Architecture for Programmers Volume I: Introduction to the MIPS32™ Architecture, rev. 2.0, 2003.

[13] 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 , Vol. 2, Issue No.4, pp. 340-346, April 2012.

[14] Allam Yamin, "Implementation and comparison of FALCON-A ISA on FPGA platforms", Master's thesis, Electrical Engineering Department, National University of Computer and Emerging Sciences, Lahore, Pakistan, 2013.

★★★