ISSN- 2350-0530(O) ISSN- 2394-3629(P)
IF: 4.321 (CosmosImpactFactor), 2.532 (I2OR)

INTERNATIONAL JOURNAL OF RESEARCH – GRANTHAALAYAH
A knowledge Repository

Science

# DESIGN OF RISC PROCESSOR USING VHDL

**Sarika U. Kadam [*1], S. D. Mali [2]**
[*1] Student, Electronics Department, Sinhgad College of Engineering Pune, INDIA
[2] Associate Professor, Electronics and Telecommunication Department, Sinhgad College of Engineering, Pune, INDIA

**DOI:** https://doi.org/10.29121/granthaalayah.v4.i6.2016.2646

## ABSTRACT

***The aim of the paper is to design a 16-bit RISC processor. It is having five stage pipelining which is designed using VHDL. RISC processors have a unique feature called pipelining. Pipelining is used to make processor faster. In Pipelining instruction cycle is divided into parts so that more than one instruction can be operated in parallel. Number of instructions are designed for this processors. Multiplier is also designed using ADD instruction. Proposed instructions are simulated using Xilinx ISE 13.1i. The processor is synthesized using Spartan-3A XC3S50A XILINX Tool.***

**Keywords:**
*Instruction Set Architecture; Pipeline; RISC; VHDL.*

**Cite This Article:** Sarika U. Kadam, and S. D. Mali, "DESIGN OF RISC PROCESSOR USING VHDL" International Journal of Research – Granthaalayah, Vol. 4, No. 6 (2016): 131-138.

## 1. INTRODUCTION

The processors are characterized by nature of their instruction set architecture. There are basically two ways of designing instruction sets CISC and RISC. RISC stands for reduced instruction set computer. It has faster and simpler set of instructions. The main feature of RISC is a pipeline. It has simple addressing modes and fixed length of instructions. RISC processor reduces cycles per instruction at the cost of a number of instructions per program. Reduced instructions in RISC require less number of transistors. RISC is used in portable devices such as Apple iPod due to its power efficiency.

CISC is a complex instruction set computer. It has a complex instruction set so decoding becomes complex. One instruction supports many addressing modes. CISC minimizes the number of instructions per program at a cost of a number of cycles per instruction. Examples of CISC are IBM 370/168 it's a 32-bit processor introduced in 1970, VAX 11/780, Intel 80486 it's a processor with 233 instructions and of varying length launched in 1989.

ISSN- 2350-0530(O) ISSN- 2394-3629(P)
IF: 4.321 (CosmosImpactFactor), 2.532 (I2OR)

Pipelining is a key feature of RISC in which processor works on different stages of instruction at the same time to execute more instructions in shorter time. The different stages of pipelines are instruction fetch, decode, execute, memory and write back. Pipelining improves throughput by working on many operations at the same time. Different processors have a different number of stages of the pipeline. The length of pipeline depends on the longest stage. Five stage pipelined processor is nearly five times faster than the non-pipelined processor. Pipelining gives high-performance processors. VHDL is used for designing processor which is a parallel programming language. The proposed processor has five stage pipelining and it is 16 bit i.e. it operates on 16-bit data. It has eight registers. As it is a RISC processor an instruction takes one clock to complete. It is simulated in Xilinx 13.1i ISE using VHDL and synthesized using Spartan-3A XC3S50A XILINX tool.

## 2. RELATED WORK

In this paper, a 16-bit RISC processor designed using VHDL where behavioral programming is used to model basic units. It is a four stage pipelined processor. Pipelining improves clock cycles per instruction. All the hazards were removed and design is implemented on FPGA [1].

In 2009 Kui YI, Yue-Hua DING designed a 32-bit RISC processor based MIPS. It is a five-stage pipelined processor. A top-down approach is used to design and language used is VHDL. It is easy to debug and simulated successfully on QuartusII [2].

In this paper, instruction fetch unit & decode unit are designed for reduced instruction set computer Processor. It is simulated on QuartusII successfully [3].

In this paper, a 32-bit RISC processor has been designed using VHDL. The reduced instruction set computer has simple decoding as it has all instructions of same length. It also has faster instructions. In the paper, the results are compared with previous processors [4].

## 3. METHODOLOGY

### *3.1.INSTRUCTION SET ARCHITECTURE*

The words of computer's language are instructions and its vocabulary is instruction set.

A. Instruction Formats

Instruction format is a collection of fields of binary numbers. The instructions of proposed processor have 3 formats Register-Type, Immediate-type, Jump-type given in figures below.

1) R-Type Format

Fig. 2 shows format of R-type instruction. It has four fields the first Opcode which is 6-bit and

ISSN- 2350-0530(O) ISSN- 2394-3629(P)
IF: 4.321 (CosmosImpactFactor), 2.532 (I2OR)

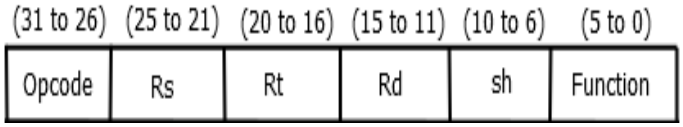


***Figure 2:*** R-type Format

used to select the type of instruction. Other fields Rs, Rt, Rd are source, target, destination registers each 5 bit gives the address of registers where data is present or to be stored. The remaining fields Sh, Function are zero for proposed processor.

OR Rs, Rt, Rd

In the above instruction values of registers, Rt and Rs are added and the result is stored in destination register Rd.

2) I-Type Format

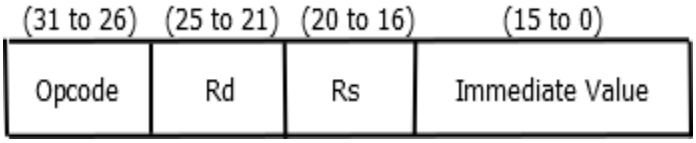


***Figure 3:*** I-type Format

Fig. 3 shows format of I-type of instruction. It has four fields of which first is Opcode which used to select instruction type. Next is Rd which is destination register's address. The immediate value is stored in it. For this proposed processor Rs is set to zero. Lower 16-bits indicate immediate value or data.

MOV Rd, #data

In the above instruction, #data is moved in Rd. Rd gives the address of destination register. #data is 16-bit data.

3) J-Type format

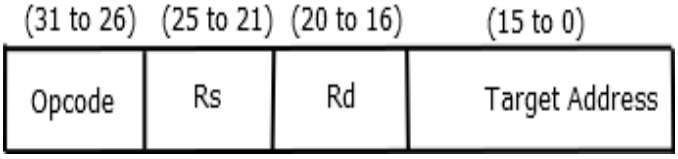


***Figure 4:*** J-type Format

Fig. 4 shows format of J-type instruction. The first 6-bits are Opcode which is used to select instruction type. Rs, Rd are zero for proposed processor. The last 16-bits are target address where a branch takes place.

ISSN- 2350-0530(O) ISSN- 2394-3629(P)
IF: 4.321 (CosmosImpactFactor), 2.532 (I2OR)

JMP Addr

The above instruction is unconditional jump instruction. Addr is an address where a branch takes place. Some conditional jumps are also present which takes place when a particular condition is true.

B. Instruction Set

All the instructions of proposed processor are 32-bit in length. Instructions have different addressing modes. Instruction formats are different for different instructions. The proposed processor has eight registers REG1, REG2, REG3, REG4, REG5, REG6, REG7 and REG8 each 16-bit wide. The processor supports register addressing, immediate addressing, register immediate addressing modes.

***Table 1:*** INSTRUCTION SET

| **Instruction** | **Opcode** |
|---|---|
| ADD Rs, Rt, Rd | 000001 |
| SUB Rs, Rt, Rd | 000010 |
| AND Rs, Rt, Rd | 000011 |
| OR Rs, Rt, Rd | 000100 |
| XOR Rs, Rt, Rd | 000101 |
| NOT Rs, Rd | 000110 |
| MOV Rd, Data | 000111 |
| SLL Rs, Rd | 001000 |
| SRL Rs, Rd | 001001 |
| ROL Rs, Rd | 001010 |
| ROR Rs, Rd | 001011 |
| JMP Addr | 001100 |

### *3.2.DATAPATH*

The data path is a collection of different functional units. It gives required elements for execution of the particular instruction. It differs according to the type of instruction. The proposed processor has three types of the data path. The data path and control unit together forms CPU. The data path elements are nothing but functional units. Such as instruction memory, ALU, PC.

A. R-Format Data Path

Fig. 5 shows register format data path. Here the first instruction code is fetched from program memory using PC. Fetched instruction is in binary form having register addresses, Opcode decides the type of instruction or function it is supposed to perform. Using that register addresses operands are read from the register file and supplied to ALU. ALU performs necessary

ISSN- 2350-0530(O) ISSN- 2394-3629(P)
IF: 4.321 (CosmosImpactFactor), 2.532 (I2OR)

operations on operands. It gives a result which is then stored in register file using destination address present in instruction.

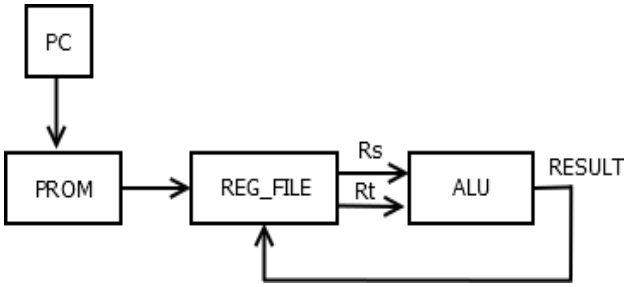


***Figure 5:*** Register-Format Data Path

**SUB Rs, Rt, Rd** is fetched from programmable memory, it is having Opcode "000010". The values of Rs and Rt are read from the register file and given to ALU to perform subtraction. The result is stored in the destination register.

B. RI-Format Data Path

Fig. 6 shows register immediate format data path. The first instruction is fetched from program memory. One operand is read from the register file and other is directly given in the instruction so this mode is register immediate data path. In earlier data path both the operands are read from the register file only with given addresses. Here one operand is directly provided in the instruction itself.

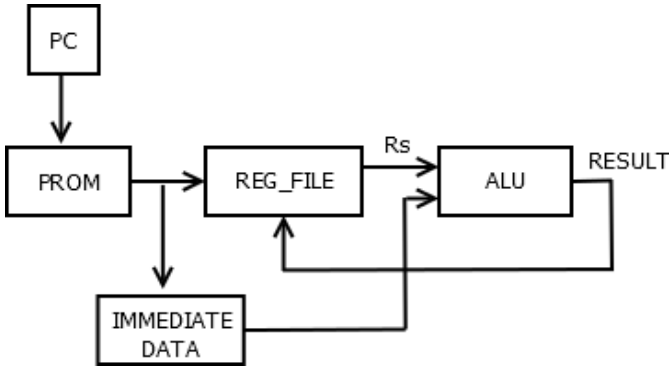


***Figure 6:*** Register Immediate Format Data Path

For example **MOV Rd, Data** instruction moves immediate value **Data** in the destination register Rd. ADD Rd, Rs, Immediate first instruction is read from program memory. The first operand is read from the register file. Second is found in the instruction itself. It is directly provided to ALU. The result of ALU is stored in the register file.

C. J-Format Data Path

Fig. 7 shows jump format data path. The jump can be conditional or unconditional. In the unconditional jump, Opcode is fetched and program counter jumps to next location specified in instruction without any condition check. This is not the case in conditional jump instruction,

ISSN- 2350-0530(O) ISSN- 2394-3629(P)
IF: 4.321 (CosmosImpactFactor), 2.532 (I2OR)

program counter value jumps to new value only when a certain condition is satisfied, such as jump when zero, jump on equal etc.

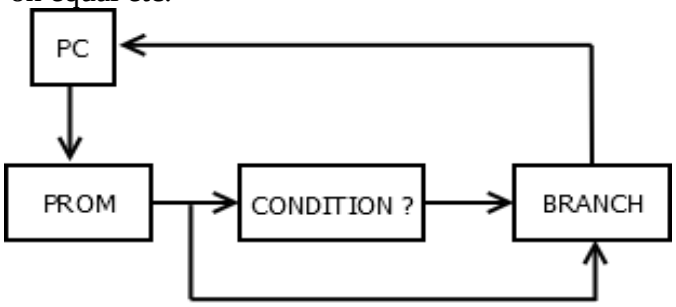


***Figure 7:*** Jump-Format Data Path

For example, **JMP Address** instruction is unconditional jump instruction where (PC) program counter jumps to the value given by Address.

## 4. RESULTS

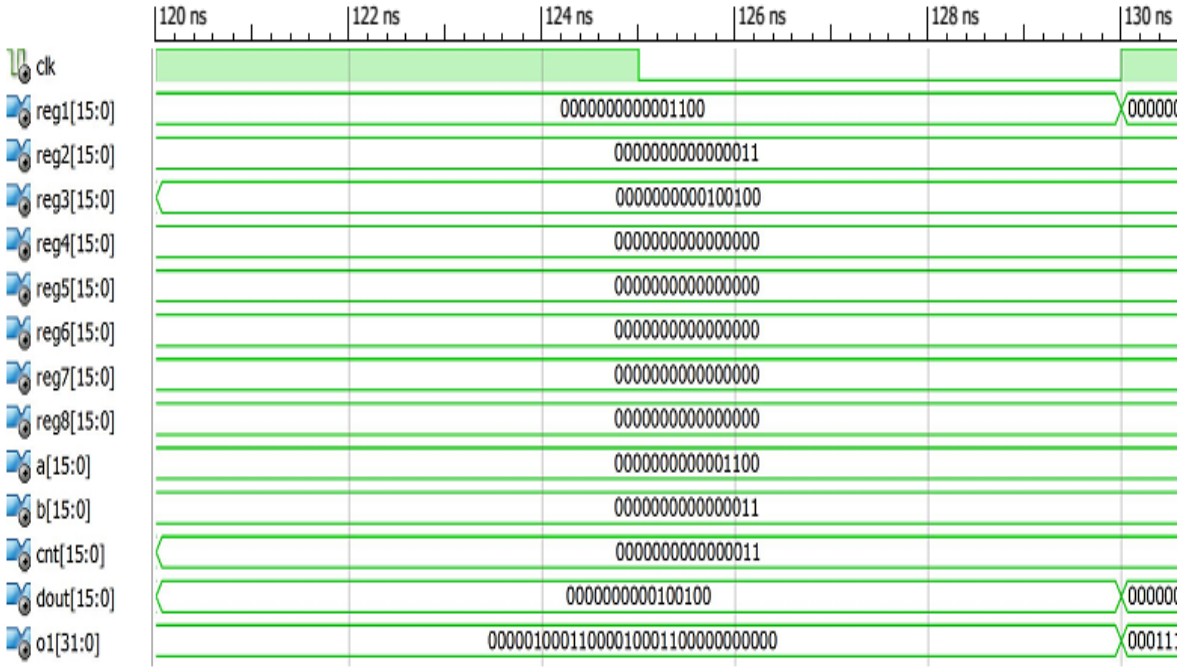


***Figure 8:*** Simulation Waveform for Multiplier

ISSN- 2350-0530(O) ISSN- 2394-3629(P)
IF: 4.321 (CosmosImpactFactor), 2.532 (I2OR)

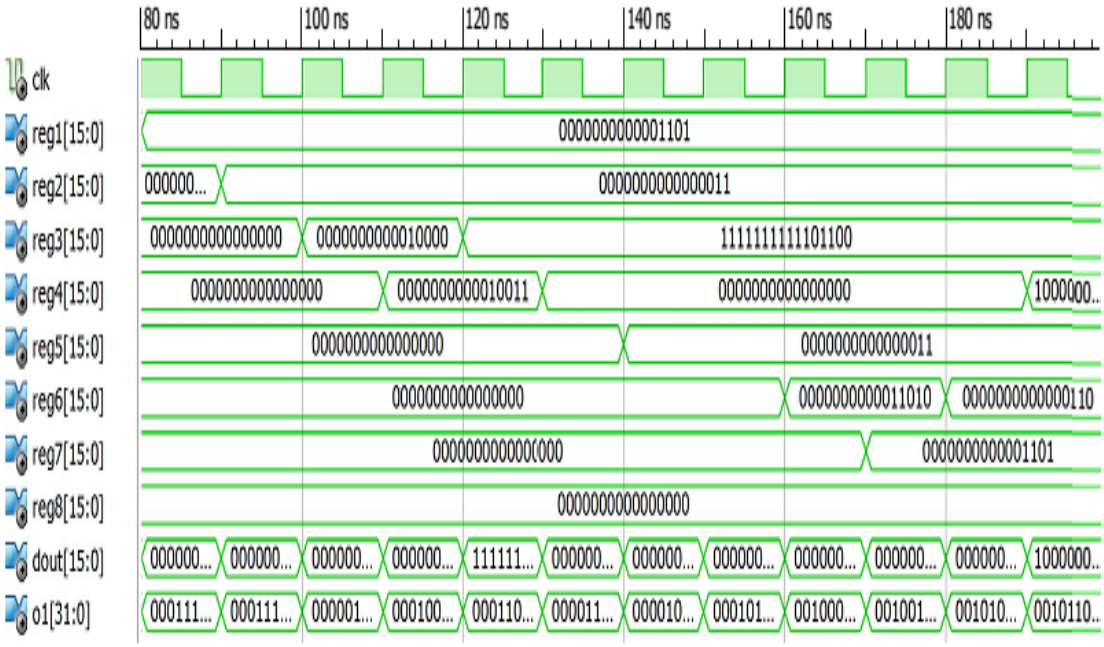


***Figure 9:*** Simulation Waveform for Multiple Instructions

***Table 2:*** PROCESSOR PARAMETERS

| **Parameter** | **Value** |
|---|---|
| Minimum period | 4.744 ns |
| Maximum Frequency | 210.775 MHz |

***Table 3:*** Comparison of previous processors with proposed work

| **Parameter** | **STM32F205xx[8]** | **8-bit RISC[7]** | **Proposed Work** |
|---|---|---|---|
| **Delay (ns)** | 8.33 | 77.43 | 4.744 |
| **Frequency (MHz)** | 120 | 12.91 | 210.775 |

| Device Utilization Summary (estimated values) | | | [-] |
|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** |
| Number of Slices | 618 | 704 | 87% |
| Number of Slice Flip Flops | 580 | 1408 | 41% |
| Number of 4 input LUTs | 838 | 1408 | 59% |
| Number of bonded IOBs | 177 | 108 | 163% |
| Number of GCLKs | 1 | 24 | 4% |

***Figure 10:*** Device Utilization Summary

ISSN- 2350-0530(O) ISSN- 2394-3629(P)
IF: 4.321 (CosmosImpactFactor), 2.532 (I2OR)

## 5. CONCLUSION

The proposed 16-bit RISC processor is designed using a parallel programming language called VHDL. It is simulated and synthesized using Xilinx ISE 13.1i. All instructions are simulated successfully. Simulation results show that the proposed processor is working correctly. The proposed processor has a delay of 4.744 ns and operating frequency of 210.775 MHz. when the proposed work compared with previous processors, it can be seen that proposed processor has less delay.

## 6. ACKNOWLEDGEMENT

I would like to thank my guide Prof. S. D. Mali for his support and guidance in fulfillment of this work. I am grateful to him for his constant encouragement and valuable guidance. This work is a combined effect of efforts put in by me and my guide. I would also like thank all those who are part of this work.

## 7. *REFERENCES*

*[1] Pravin S. Mane, Indra Gupta, M. K. Vasantha, "Implementation of RISC Processor on FPGA", Electrical Engineering Department, Indian Institute of Technology Roorkee, 2006 IEEE.*

*[2] Kui YI, Yue-Hua DING, "32-bit RISC CPU Based on MIPS", International Joint Conference on Artificial Intelligence, 2009 IEEE.*

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

*[4] Mr. S. P. Ritpurkar, Prof. M. N. Thakare, Prof. G. D. Korde, "Synthesis and Simulation of a 32Bit MIPS RISC Processor using VHDL", International Conference on Advances in Engineering & Technology Research, 2014 IEEE.*

*[5] David A. Patterson, John L. Hennessy, "Computer Organization and Design", the Hardware and Software Interface, Third Edition.*

*[6] Douglas L. Perry, "VHDL Programming by Example", Fourth Edition.*

*[7] R. Uma, "Design and Performance Analysis of 8-bit RISC Processor using Xilinx Tool", International Journal of Engineering Research and Applications, 2248-9622, April 2012.*

*[8] STM32F205xx, STM32F207xx 2013 datasheet, STMicroelectronics.*