# Design and Implementation of 32 Bit 5-stage Pipeline RISC Processor using Verilog HDL

**MR. PARTHKUMAR D. CHHODAVADIYA, PROF. MANISHA PATEL**

**M.E. STUDENT , ASSISTANT PROFESSOR**

**Department of Applied Instrumentation,**
**L. D. College of Engineering-Ahmedabad,India**

**Abstract** *:* **Aim of the work is to design and reduce the dynamic power consumption of low power 32 bits RISC core processor. The design is based on 5-stage pipelined MIPS architecture. This paper proposes the design for the low power RISC processor.**

**It is having five stages pipelining which is designed using Verilog HDL. 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. Proposed instructions are simulated using Xilinx ISE 14.7. The processor is synthesized using Spartan3e XILINX Tool.**

***IndexTerms - FPGA, RISC, 5-Stage Pipeline, MIPS Instruction set Architecture.***

## I. 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.
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. Verilog HDL is used for designing processor which is a parallel programming language. The proposed processor has five stage pipelining and it is 32 bit i.e. it operates on 32 bit data. It has eight registers. As it is a RISC processor an instruction takes one clock to complete.

## II. LITERATURE SURVEY

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].

## III. METHODOLOGY

### 1. INSTRUCTION SET ARCHITECTURE

The instructions of RISC Processor based on MIPS are categories into four different instruction formats; R-Type, I-Type, J-Type and I/O Type as described in figures below;

*a) R-Type Instruction Format*

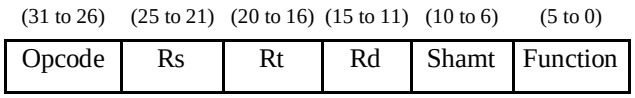


Figure 1. R-Type Instruction Format

Figure 1 shows Register type instruction format. This type of format has total six fields; Opcode field of 6 -bit, used to select the type of instruction format, Rs (Source register), Rt (Target register) and Rd (Destination register) are of 5-bit each which are used for storage of data. Shamt (Shift amount) field of 5-bit, used for data shifting and last is the Function field of 6-bit used for selection of different functions to be performed.

*b) I-Type Instruction Format*

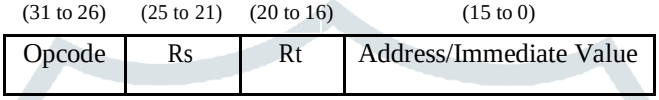


Figure 2. I-Type Instruction Format

Figure 2 shows Immediate type instruction format. This type of format has total four fields, Opcode field of 6 -bit, used to select the type of Instruction format, Rs (Source register) and Rt (Target register) are of 5-bit each, used for storage of data and the last is Address/Immediate Value field of 16-bit, used for immediate data operations.

*c) J-Type Instruction Format*


Figure 3. J-Type Instruction Format

Figure 3 shows J -type instruction format. This type of format has only two fields; Opcode field of 6-bit, used to select the type of Instruction format. Target address of 26-bit, used to specify on which address to jump.

*d) I/O-Type Instruction Format*

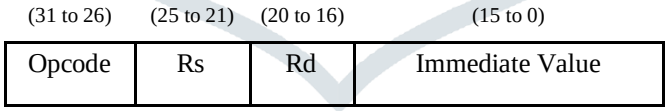


Figure 4. I/O-Type Instruction Format

Figure 4 shows I/O-type instruction format. This type of format has total four fields; all the fields are same as in R-Type and I -Type instruction formats. Instructions which require read and write of data from port uses this type of instruction format. Execution of any instruction becomes faster by using such types of instruction formats.

### 2. DATA FLOW

Data flow is determined by hardware data path, which express data flow process. There is no clear difference between data and control. Operation code, operand, memory address and value, register address and value, jump destination address and content are usually included in data, but control composes of control signal of unit ,time sequence control signal and interrupt control signal, and these signals are not always defined clearly and strictly.

#### *A. R-Format Data Path*

In R-Format data path, fetch instruction from memory and analyze instruction into different parts. Two register specified by instruction fetch data from register file and ALU execute instruction command. Finally, after ALU outputs answer write the answer to register file. Figure 5 shows R Format data path.

For example, ADD R1, R2, R3 instruction, which is add signed word instruction(R1= R2+ R3). Data flow of this instruction shows as following: PC fetches ADD R1, R2, R3 instruction from memory. At first, the instruction access two registers R2 and

R3 and value of the two register is put to ALU. After arithmetic is over, ALU write back result to R1 register. And then, data flow is in the end.

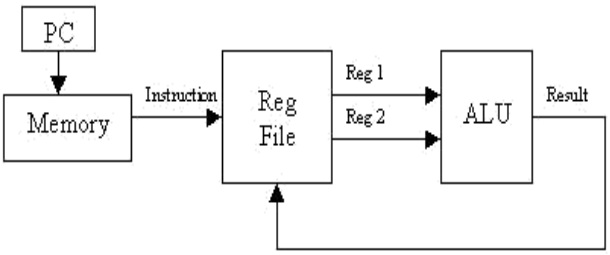


Figure 5- R Format data path.

For another example, SRL R1, R2, R3 instruction, which is shift word right logical instruction. Data flow of this instruction shows as below: PC fetches SRL R1,R2,R3 instruction from memory. At first, the instruction access two register R2 and R3 and value of the two register is put to ALU. After arithmetic is over, ALU write back answer to R1 register, And then, data flow is in the end.

### *B. RI-Format Data Path*

RI-Format instruction is similar to R-Format instruction.

The difference between them is that the second read register of R-format instruction is replaced by immediate of RI-Format instruction. The immediate is 32-bit signed number which is extend by 20-bit number, and put to ALU as the second operand. Finally, write-back result to register file. RI-Format data path shows as Figure. 6.

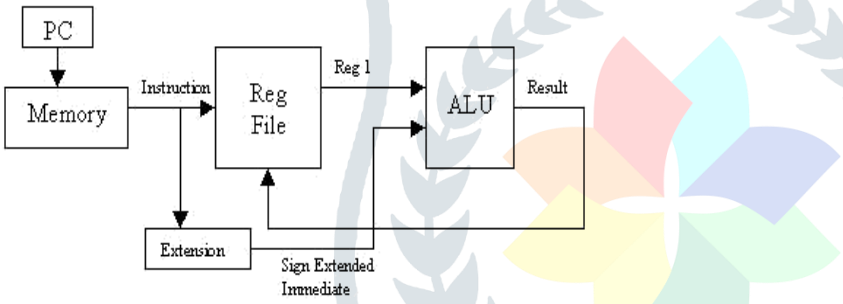


Figure 6- RI-Format Instruction Data Path

Format includes ADDI R1, R2, data6 instruction, SUBI R1, R2, data6 instruction etc .When ADDI R1, R2, data6 instruction executes, PC fetches ADDI R1, R2 data6 instruction from memory and register R2 value is put to ALU. At the same time, immediate data6 is extended to 32-bit signed number and put to ALU Finally, after ALU completes add of the two operands, ALU writes back answer to R1 register. The difference data flow between SUB R1,R2 data6 instruction and ADD R1,R2,data6 instruction is that the former instruction do subtraction

### *C. Load Word Data Path*

Load word data path is similar to I-Format data path. The difference between the two data path is that result is written to memory in load word data but result is written to register in I-Format. In load word data path, fetch data from memory and load it to register file. Load word data path shows as Figure.3. LW R1, R2, data6 instruction is the only one instruction in load word data path. It works shows as below: PC fetch LW R1, R2, data6 instruction from memory. R1 register is to load data. Firstly send R2 register value to ALU, at the same time, extend data6 immediate to 32-bit and send it to ALU.

The answer of adding the two numbers is memory address. And then, copy content of the memory address to R1 register.

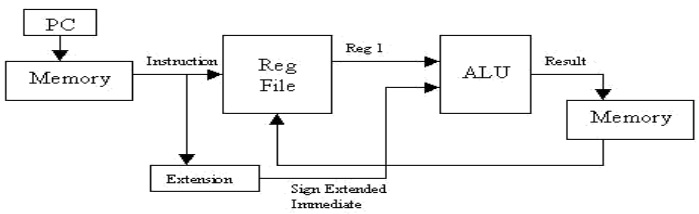


Figure. 7 Load Word Data Path

### *D .Memory Word Data Path*

Memory word data path is similar to load word data path, but target which register is to write is memory but not register file. There is only SW R1, R2, data6 instruction in load word instruction. PC fetches SW R1, R2, and data6 instruction from memory. R1 register stores data which is to be stored. Firstly, send R2 register value to ALU, at the same time, extend data6 immediate to

32-bit and send it to ALU. The result of adding the two numbers is memory address. Memory instruction data path shows as Figure 8

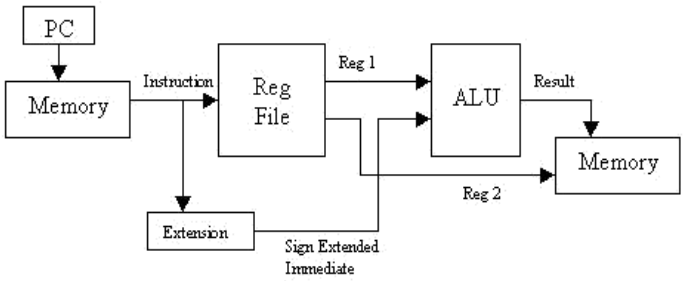


Figure. 8 Store Word Data Path

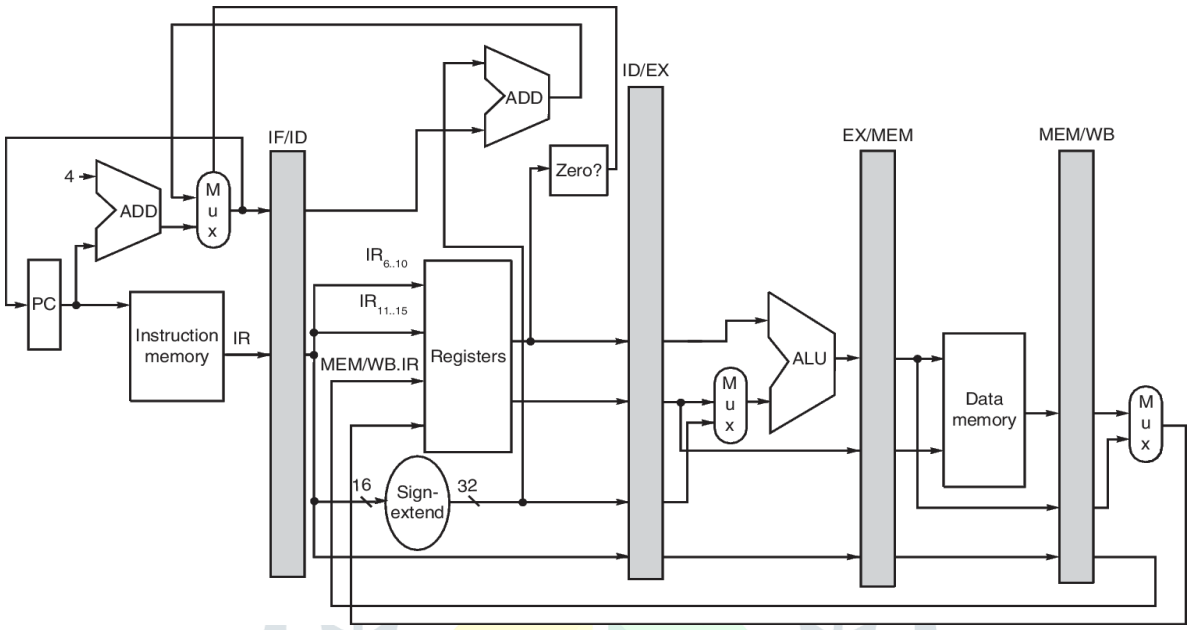


Figure 9. Block Diagram of RISC Processor

## IV. LOW POWER TECHNIQUES

3.1 Power Dissipation Sources: The power loss can be classified as described below:

a) Leakage power consumption: The dissipation of power occurs when system is idle/standby mode .
b) Dynamic power consumption: Due to logic transitions of logic gates or flip-flops.

3.2 Low-Power Design Technique: The various layers at which the power reduction can be achieved are as listed below:

a) Circuit Logic: Logic Styles, Energy Recovery, Transistor Sizing
b) Architecture: Parallelism, Pipelining, Redundancy, Data Encoding
c) Technology: Threshold Reduction, Multi-threshold devices
d) System: Partitioning, Power down
e) Algorithm: Complexity, Concurrency, Regularity

The power reduction techniques are done based upon the changes in either Pipelining, Logic styles, Data Encoding. These various techniques will be explored to achieve power reduction.

3.3 Coding option using Verilog HDL to optimize power
FPGA:

a) No reset is best since the FPGA gets a global reset automatically
b) Use low power architecting for multiple Block RAM arrays (use CORE generator)
c) Only enable BRAM during active read or write cycles
d) Design with Synchronous Resets
e) Minimize local resets, if possible
f) Build small memory blocks with LUTs (<4k bits)
g) Control the use of Clock Enables

## V. FPGA BASED SIMULATION

To design and verify a 32-bit RISC core, following design methodology is used
Test bench for top level design, along with stimulus

a) Simulation results / Timing diagrams
b) Verification Scheme: Functional Simulation using Modelsim Simulator
c) Hardware Description Language to describe design: Verilog HDL
d) Target Technology: Xilinx Spartan-3 Device
e) Design synthesis using Xilinx ISE software
f) FPGA Implementation:

| Device Utilization Summary (estimated values) | |
|---|---|

Generate a design bit stream for FPGA device.
Design to be synthesized using Xilinx ISE Software.

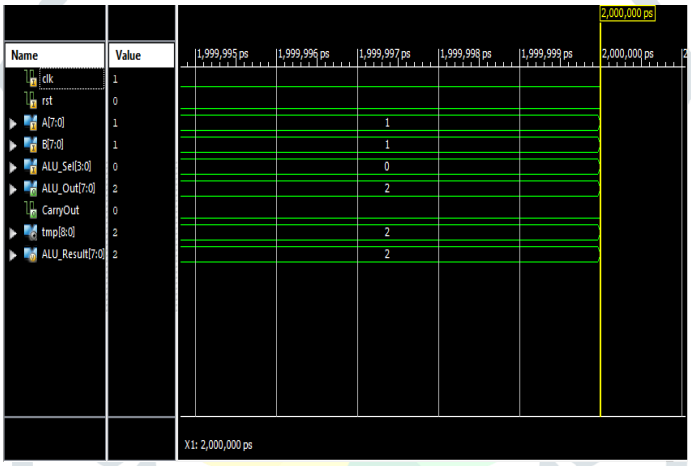


Fig. 10 Sumulation Result Of Addition operation on ALU

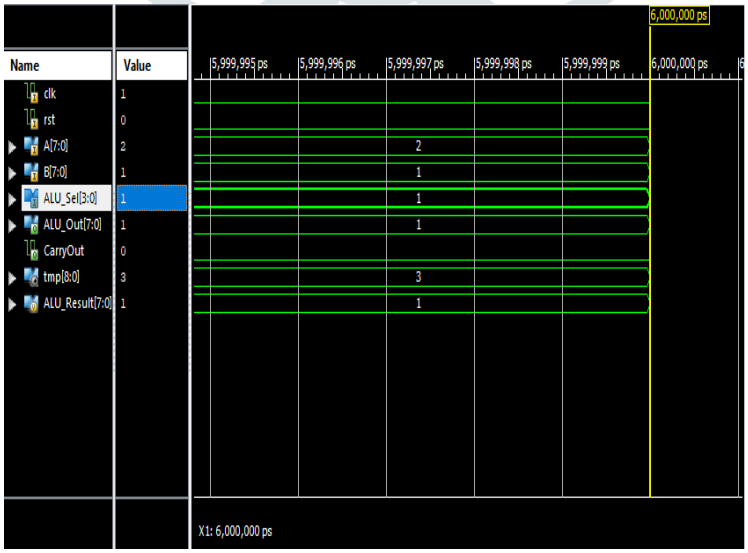


FIG 11: Simulation results of Subtraction operation performed in ALU

The simulation result is as shown in figure 10 and figure11. The execution of arithmetic and logical instruction is depicted in the figure. Various inputs are applied to the operand A and B registers. The result is stored in output register. By taking various snapshots of the simulation it is possible to analysis the result at different clock cycles.

| Logic Utilization | Used | Utilization | Proposed By Base Paper [1], Rohit J., Raghavendra M. ,IEEE 2017 | Utilization |
|---|---|---|---|---|
| Number of Slice Flip-Flop | 1321 | 28% | 15519 | 37.31% |
| Number of LUTs | 604 | 6% | 14370 | 69.09% |
| Number of slices | 2542 | 27% | 4129 | 43.01% |
| Number of BRAM | 11 | 55% | 6.5 | 13% |
| Number of GCLKs | 1 | 4% | 1 | 4% |

## VI. RESULTS

Table : 5 Utilization Report

Table : 6 Power Consuption Report

| On-Chip | Power (W) | Base Paper (w) [1], Rohit J., Raghavendra M. ,IEEE 2017 |
|---|---|---|
| Clocks | 0.000 | 0.000 |
| Logic | 0.000 | 0.000 |
| Signals | 0.000 | 0.000 |
| IOs | 0.000 | 0.000 |
| Total | 0.081 | 0.09 |

## VII. CONCLUSION

The proposed 32-bit RISC processor is designed using a parallel programming language called Verilog HDL. It is simulated and synthesized using Xilinx ISE 14.7. 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.

## VIII. FUTURE SCOPE

The above developed design can be reduced further using system portioning. Consumption of the power can further be reduced at the manufacturing stage using techniques such as lithography.

## IX. ACKNOWLEDGEMENT

I would like to thank my guide Prof. Manisha patel 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.

## X. REFERENCES

[1] Rohit J., Raghavendra M.,'' Implementation of 32-bit RISC processors without interlocked pipelining on Artix-7 FPGA Board", Second International conference on Circuits, Controls and Communications,978-1-5386-0615-5/2017 IEEE
[2] Neha Dwivedi, Pradeep chhawcharia,'' Power Mitigation in High-Performance 32-bit MIPS-based CPU on Xilinx FPGAs'', 2017 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), 978-1-5386-2787-7/2017 IEEE
[3] S.P.Ritpurkar, M.N. Thakare, G.D.Korde," Design and Simulation of 32-Bit RISC Architecture Based on MIPS using VHDL", 2015 International Conference on Advanced Computing and Communication Systems, 978-1-4799-6438-3/2015 IEEE
[4] Soumya Murthy, Usha Verma,''FPGA based Implementation of Power Optimization of 32 Bit RISC Core using DLX architecture", 2015 International Conference on Computing Communication Control and Automation, 978-1-4799-6892-3/ 2015 IEEE
[5] Rupali S. Balpande, Rashmi S. Keote, "Design of FPGA based Instruction Fetch & Decode Module of 32-bit RISC (MIPS) Processor", IEEE International Conference on Communication Systems and Network Technologies, 2011
[6] Neenu Joseph, Sabarinath.S, Sankarapandiammala K, "FPGA based Implementation of High Performance Architectural level Low Power 32bit RISC Core" International Conference on Advances in Recent Technologies in Communication and Computing, Annual IEEE pp.5357, 2009.
[7] 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.
[8] Ball, "Designing Soft-Core Processors for FPGAs Processor Design", in Processor Design: System-on-Chip Computing for ASICs and FPGAs, 1. Nurmi, 1st Ed: Springer Netherlands, 2007, pp. 229-256
[9] Pranjali S. Kelgaonkar, Prof. Shilpa Kodgire, "Designed 32-bit Pipelined RISC on Spartan-6", IEEE, 2016.
[10] 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.
[11] Xilinx Inc, Libraries Guide for ISE available at www.xilinx.com 2013
[12] Xilinx Inc, Xilinx ISE Software Manuals, available at www.xilinx.com 2013
[13] Xilinx Inc, XST User Guide, available at www.xilinx.com 2013
[14] Modelsim User Guide,available at www.mentor.com 2013
[15] DLX Architecture, available at http://en.wikipedia.org/wiki/DLX
[16] Patterson, David; Hennessy, John (1994). "Computer Organization and Design" (1st Ed.). Morgan Kaufmann. ISBN 978-1-55-860281-6.
Sailer, Philip M.; Kaeli, David R. "The DLX Instruction Set Architecture Handbook". Morgan Kaufmann. ISBN 1-55860-371-9.
[17] Stephen B. Furber, "VLSI RISC Architecture and Organization", 2nd Edition CRC Press, pp. 24-28
Alex van Someren, Carol Atack, "The ARM RISC Chip, A Programmers Guide", 6th Edition CRC Press, 1993
[18] Y. Ye and K. Roy, ".QSERL: Quasi-Static energy recovery logic. IEEE J. Solid-States Circuits", vol. 36, no. 2, pp. 239.248, Feb. 2001.
[19] Xilinx Inc, Libraries Guide for ISE available at www.xilinx.com 2013
[20] Xilinx Inc, Xilinx ISE Software Manuals, available at www.xilinx.com 2013
[21] Xilinx Inc, XST User Guide, available at www.xilinx.com 2013
[22] Modelsim User Guide, available at www.mentor.com 2013
[23] DLX Architecture, available at http://en.wikipedia.org/wiki/DLX