International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 05 | May 2021 www.irjet.net p-ISSN: 2395-0072

# Implementation of 32-bit RISC-V Processor

**Nischitha D D[1], K Sanjana Priya[2], Shivani D[3], Yashaswini K L[4] and Dr. C M Patil[5]**

[1-5]*Department of Electronics and Communication Engineering, Vidyavardhaka College of Engineering, Mysore, Karnataka, India - 570002*

---

**Abstract -** *The main aim is to implement a 5-stage pipelining based 32-bit RISC-V Processor. The processor is designed on Verilog HDL in Cadence tool. It supports basic instruction and vector arithmetic. This processor is handled using R-Type, I-Type and Jump instruction. It designs and synthesizes the MIPS processor using register files and insert the ALU forwarding unit in order to avoid the stalls and hardware interlocks. It supports data forwarding to prevent data hazard and supports hazard detection as well. After that calculate area and power using Cadence RTL compiler using slow and fast libraries of 45 nm technology. The required netlist is obtained using genus tool.*

**Key Words: RISC-V (Reduced Instruction Set), Harvard Architecture, ALU (Arithmetic and Logical Unit), Verilog, Cadence, Genus.**

## 1. INTRODUCTION

These days there has been developing interest in RISC-V processor because of low power utilization and faster execution. The feature of the RISC-V processor is to speed up the execution. The 32-bit RISC-V processor includes load and store architecture and also uses the Harvard architecture to expand the performance of the processor.

Popularly used RISC processors are MIPS, SuperH, IBM and SPARC etc. MIPS processors are mainly used in Cisco Routers, digital cameras, Sony play station game consoles and Windows CE devices. The MIPS processors are generally same kind of designs. However, it changes in the execution stages like pipelining, single or multiple. The operations are performed on chip registers rather than memory locations, because the access time differs for register compared to memory location. Micro pipeline technique is used in RISC-V Processor, as each instruction is executed in number of stages simultaneously.

## 2. LITERATURE SURVEY

Low Power Implementation of RISC-V Processor

It has become serious issue in design of System on Chip (SoC) with increase in the rate of power consumption. So, it is very important that the chip design engineer to improve the power management at the architecture level itself. Because of the reduction in the size of the chip, leakage of current is increasing rapidly. It is required to manage the power of all design of 90 nm and below. Since the management of leakage current has high priority on the design and the implementation, as for few libraries and design, hence the source dissipates power in CMOS. RISC-V is newly introduced ISA, which is free open source for industry implementation being under the control of RISC-V establishment.

The main theme is to get the efficient low power RISC-V processor with DFT. The power management techniques for switching and leakage power reduction are applied on the design are as follows

i. Multi-$V_{th}$

ii. Clock Gating and Clock Tree Optimization

iii. Multi-supply voltage

iv. Power Shut Off (PSO)

Improving Energy Efficiency and Reducing Code Size with RISC-V Compressed

In both simple and complex type of processor the Power dissipation and the energy efficiency is concentrated more. Instruction set architects has broadly used two techniques to reduce the relative energy cost of instruction stream delivery.

1. To increase the amount of work performed by a single instruction.
2. To reduce the size of the instruction.

Finally, discussed the implications of RISC-V Compressed (RVC) for energy efficiency, performance, and processor design. And compare the code size of RVC with other commercial ISAs.

Variable-length RISC ISAs can reduce static and dynamic code size as compared to their fixed-length counterparts; also, they avoid performance loss of a less-capable ISA that comprises only short instructions.

To analyze the set of RISC-V Compressed (RVC) instruction and to evaluate RISC-V Compressed (RVC) effectiveness, have to collect static and dynamic measurements from a subset of the SPEC CPU2006 benchmark. Static measurements were obtained directly from the resulting executables and object code. Dynamic measurements were obtained from a RISC-V instruction set simulator, running the benchmarks to completion using their small input sets.

Arnold: an eFPGA-Augmented RISC-V SoC for Flexible and Low-Power IoT End-Nodes

RISC-V Micro-Controller Unit (MCU) demonstrate the flexibility of the System on Chip (SoC) to tackle the challenges of many emerging IoT applications, such as

i. Interfacing sensors and accelerators with non-standard interfaces.

ii. Performing on-the-fly pre-processing tasks on data streamed from peripherals

iii. Accelerating near-sensor analytics, encryption, and machine learning tasks.

The contribution of the presented heterogeneous System on Chip (SoC) design and silicon demonstrator is summarized as follows.

1. Architectural Flexibility
2. Power management
3. Leading edge performance and energy efficiency

HW/SW approaches for RISC-V code size reduction

Code density is a significant concern for low-cost IoT MCUs, as it directly impacts the on-chip memory area (and cost) and indirectly influences power and performance.

The code size in RISC-V can be reduced by: -

- Firstly by, tuning the toolchain at compile and link-time with optimal settings, including those needed for including libraries and using the linker script.
- Secondly demonstrated that RISC-V non-standard extensions, such as the Xpulp extension, can boost performance without any code size penalty.

The memory becomes a fundamental unit to use with care as it is usually one of the most expensive parts of chips and it biases the Performance, Power, and Area (PPA) results.

The proposed steps are:

a) An analysis of the main compiler and linker options to reduce code size.

b) A code size comparison of the two ARM and RISC-V ISAs.

c) Evaluates the impact on code size of the custom RISC-V Xpulp extension and originally designed for pushing energy efficiency.

d) A new RISC-V extension that targets an increased code size density as a possible solution to decrease the density gap between the two ISAs.

e) Finally, evaluated the impact on core logic area of the HW implementation of compressed push/pop/popret on an open-source core.

A Compression Instruction Set Design based on RISC-V for Network Packet Forwarding

The multi-core processor of RISC for network packet forwarding has been restricted by the on-chip storage space. As more and more cores are implemented in one chip, the storage resources allocated by each core on the chip become less and less, by this the conflict of visiting RAM has become more prominent. Therefore, the utilization of more compact instruction size will decrease the number of visits, and get a higher instruction cache hit rate, accordingly improving the performance of the application and energy productivity.

A compressed ISA for the network packet forwarding is demonstrated by trying the new instruction set which altered by this method has higher compression effectiveness and better system execution for the network packet forwarding applications. By implementing the network processor of the multi-core architecture, the processing capacity for packet forwarding has been improved. It also brought some problems, like high cost, less storage space for each core, higher energy dissipation, etc.

In order to obtain the real compression effect, designed a complete hardware and software experimental environment.

- Hardware Experimental Environment

It adopts the high-performance hardware FPGA board and the single core supporting RISC-V Compressed (RVC), and RISC-V(RV).

- Software Experimental Environment

The software experiment environment includes test assembly, PC test program, compiler, etc. And have an approach to custom RISC-V compression instruction set, and presents a custom instruction set for network packet forwarding.

A RISC-V Processor SoC With Integrated Power Management at Submicrosecond Timescales in 28 nm FD-SOI

A RISC-V Processor System on Chip (SoC) is implemented with integrated voltage regulation, adaptive clocking, and power management in a 28 nm completely depleted silicon on-insulator process. There is a second core which serves as an integrated power-management unit that can measure system state and change core voltage and frequency, permitting the execution of a wide variety of power management algorithms that can react at sub microsecond timescales while adding simply 2.0% area overhead.

Energy efficiency is the key constraint in modern Systems on Chip (SoCs). The Server-class chips are thermally limited and require better energy efficiency to improve performance, while the utility of mobile and IoT devices depends substantially on low energy consumption to prolong battery life.

The system is partitioned into two voltage domains:

1) Core domain where a variable-voltage contains the application processor.

2) Encore domain where a fixed 1V contains the power measurement and control blocks. Because the core can operate at varying frequencies and voltages, all digital communication between the core and the encore uses level shifters and asynchronous queues.

This processor SoC couples an energy-efficient RISC-V core and vector accelerator with a power-management processor, integrated voltage regulators, and an adaptive clock generator,

allowing for improvements in system energy efficiency through the use of power-management algorithms running in microsecond-scale feedback loops entirely on-die.

Design of 32-bit Asynchronous RISC-V Processor using Verilog

The 32-Bit RISC-V processor Comprises of Arithmetic Logic Unit (ALU), Booth's Multiplier, Control Unit, Register Bank, Memory and Data path. The Architecture is partitioned into five phases. The pipeline stages are Instruction Fetch (IF), Instruction Decode (ID), Instruction Execution (IE), Memory Access (MA), and Write Back (WB). Design has 3 types of memories; data, instruction and register memory.

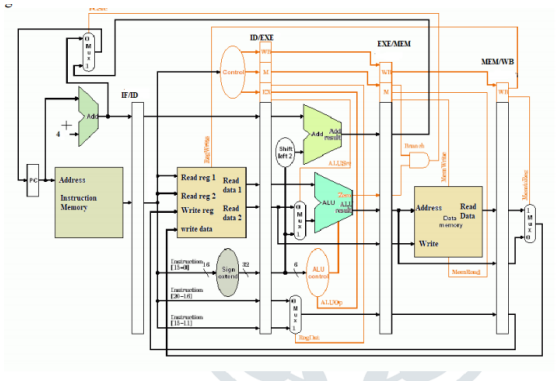



Fig1: 32-bit Asynchronous RISC-V Processor

Every instruction starts with a 6-bit opcode. There are 3 types of instruction format: - R-Type, I-Type and J-Type.

- In Register(R) Type: - In addition to the opcode, it has 3 registers (rs, rt, rd), a shift amount field and a function field. This type performs Arithmetic and logical (ALU) operations. Shift Amount (SA) is used to shift and rotate instructions. The amount of shift is decided by the source operand rs. Function is used because it contains the control codes to differentiate the multiple instructions.
- In Immediate (I) type: - In addition to the opcode, it has 2 registers (rs, rt) and 16-bit immediate value. Equal or not equal operation is performed.
- In Jump (J) type: - In addition to the opcode, it has 26-bit target address and perform jump operation.

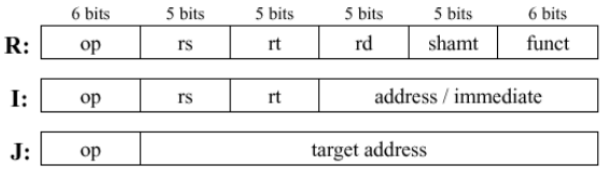



Fig2: Instruction Format of Asynchronous RISC-V

FPGA Implementation of 32-bit RISC-V Processor with Web-Based Assembler- Disassembler

The proposed RISC-V processor is planned utilizing Verilog and it is executed on Cyclone IV 4CE115 FPGA gadget accessible on Altera DE2-115 Board. Additionally, electronic constructing agent and disassembler instruments are created and distributed as a piece of this task.

Before utilizing the objective RISC-V processor, the client can create machine code utilizing the electronic constructing agent instrument. At that point, the created machine code can be downloaded onto the RISC-V processor utilizing UART. The online constructing agent and disassembler apparatuses are created with advancements, for example, HTML5, CSS and JavaScript. The proposed processor is a completely useful processor that utilizes RV32I base number instructional set with 37 guidelines.

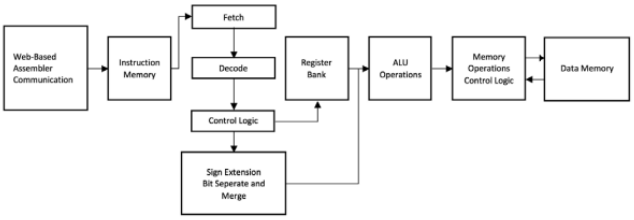



Fig3: Processor Design

Due to the absence of an electronic RISC-V Assembler and Disassembler, individuals invest pointless energy making an interpretation of the get together code to machine code. The application that takes the RISC-V Assembly Code as a contribution from a 'text region' in a Web-website at that point changes it over to the Machine Code as a yield was created. Using the online constructing agent and disassembler communication between this framework and FPGA gave by the UART. The given info goes to FPGA and reproduction works dependent on the information. All the reproduction systems have been done in Logisim. Logisim is an instructive device for planning and recreating rationale circuits. 32-cycle RISC-V processor was executed in a primary base and completely bolsters RV32I base whole number guidance set. To show the yield, the LCD on FPGA was utilized.

❖ Design of a 32-bit, dual pipeline superscalar RISC-V processor on FPGA.

A 40 MHz, 32-cycle, 5-stage double pipeline superscalar processor dependent on RISC-V Instruction Set Architecture is introduced. It upholds number, increase partition and nuclear read change compose tasks. The proposed framework executes all together giving of directions. The configuration consolidates a unique branch expectation unit, memory subsystem with virtual memory, separate guidance reserve and information store, number and floating point execution units, intrude on regulator, blunder control module, and a UART fringe. The interfere with regulator upholds four degrees of preemptive need, which is programmable for singular intrudes.

Error control module gives single blunder remedy and twofold mistake identification for the primary memory. Wishbone B.3 transport standard is embraced for on-chip correspondence. The processor is actualized on Virtex-7 XC7VX485TFFG1761-2 FPGA based board. Module gives single blunder remedy and twofold mistake identification for the primary memory. Wishbone B.3 transport standard is embraced for on-chip correspondence.

Design of RISC Processor Using VHDL and Cadence

RISC Processor is done using VHDL synthesis model. Initially, a data path module of the processor will be demonstrated how the individual modules are going to send the data from one module to the next and then a screen capture will be appeared on the top-module view to show how each bit of the processor will be from an external viewpoint. The complete processor contains nine blocks. It includes Program Counter (PC), Clock Generator, Instruction Register, ALU, Accumulator, Decoder, IO Buffer, Multiplexer, and Memory. Along with these modules an appropriate busses form together to create a processor capable of storing, loading, and performing arithmetic and logical operations. When the fetch signal is high and the address multiplexers chooses the contents of the Program counter to be stacked on to the address bus and read cycle is started and the micro instruction code is placed onto the data bus. The program counter is incremented to point to the next instruction in the memory location of the control memory. The data bus transfers the micro instruction to the Instruction Register.

The instruction registers are of two formats: -

a) Opcode, data operand: The opcode is sent to the ALU and decoder for decoding and a series of micro-operation are created. The data operand is loaded on to the data bus and moved to the ALU for its particular micro-operations as determined by its opcode.

b) Opcode, Address of data operand: The address of the data operand is loaded onto the address bus and a memory read cycle is started. Here the memory location in the primary memory indicated by the address lines is read and the information is moved onto the data bus and the ALU undergo the operations specified by its opcode.

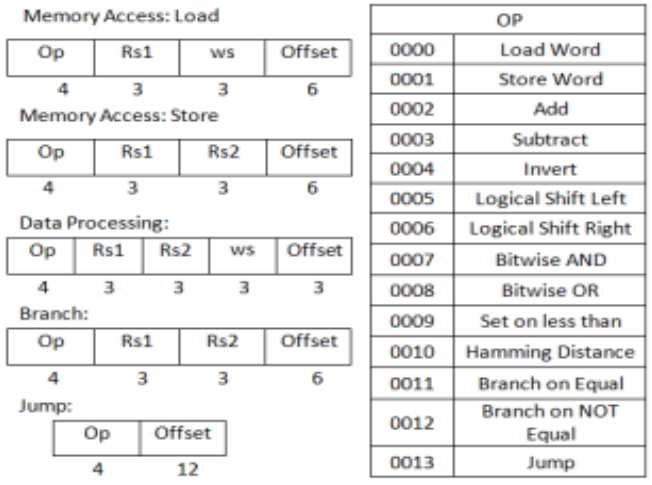
A clock generator produces a clock signal. This signal is used as inputs to the decoder which controls the operation of the processor. It produces reset pulse which should be dynamically low. In instruction register instructions are fetched and stored.

In instruction register instructions are fetched and stored. It is performed only during positive edge of the clock. An accumulator is a type of register which acts as a temporary storage location of ALU output and it is activated only at the positive edge of clock. Memory has mem_rd, mem_wr and address as inputs and data as output. If mem_rd is high, it peruses the data of memory to the data register, if mem_wr is high, data is written to the memory. Arithmetic and Logic Unit (ALU) is a multiplexer, performs standard mathematical operations. ALU operations should be synchronized to the negative edge of the clock. The address multiplexer chooses one output of the two given inputs. At the point when the fetch signal is high, the address of the program counter is moved on to the address buses and thus instruction is fetched. In any case, assuming low, the operand address determined in the address field of the instruction register is moved onto the address bus and consequently fetched. A decoder gives the correct sequencing of the system. Depending upon the opcode it translates after it gets from the instruction register. The decoder is a basically limited state machine which comprises of states. After converting VHDL to PSPICE library and object file. The next step is to synthesize.

❖ Design of a 16-Bit Harvard Structure RISC Processor in Cadence 45nm Technology

The MIPS processors are generally same kind of designs. However, it changes in the execution stages like pipelining, single or multiple. The operations are performed on chip registers rather than memory locations, because the access time differs for register compared to memory location. Due to the operation speed mobile phones, tablets and portable devices are using ARM RISC processor. The disadvantage in portable devices was that takes high power which leads to less battery life and causes failure in silicon parts of the devices. This disadvantage has been decreased in this project during by-pass the pipelining stages, but it causes Dynamic power dissipation. The power dissipation is mainly due to unwanted switching stages or a greater number of transitions present in the device.

The pipelining stage includes fetch, decode, execute and memory read/write operations. 4-stage pipelining and Clock gating is implemented to reduce the performance and power. Mainly this design reduces the Dynamic power dissipation as clock turn off clock signal when not need and it load up to four clock cycles so that simultaneously task can be done. Every output of the pipelining stage is the next state input. The microinstructions performed in this design were separated.




| OP | |
|---|---|
| 0000 | Load Word |
| 0001 | Store Word |
| 0002 | Add |
| 0003 | Subtract |
| 0004 | Invert |
| 0005 | Logical Shift Left |
| 0006 | Logical Shift Right |
| 0007 | Bitwise AND |
| 0008 | Bitwise OR |
| 0009 | Set on less than |
| 0010 | Hamming Distance |
| 0011 | Branch on Equal |
| 0012 | Branch on NOT Equal |
| 0013 | Jump |

Fig4: Instruction Format of RISC

ASIC Design of MIPS Based RISC Processor for High Performance

RISC uses pipelining concept and number of register to store the intermediate data values.The execution of an instruction is divided into number of stages.

• The IF stage gets the next instruction from memory with the address present in the Program Counter (PC) and afterward it will be stored in the Instruction Register (IR).

• In ID stage instruction are decoded and evaluates the program counter instruction, and reads if any operand is required from register.

• In EXE stage the execution of ALU operation on instructions takes place.

• Memory Access stage takes place only if any current instruction requires the memory access.

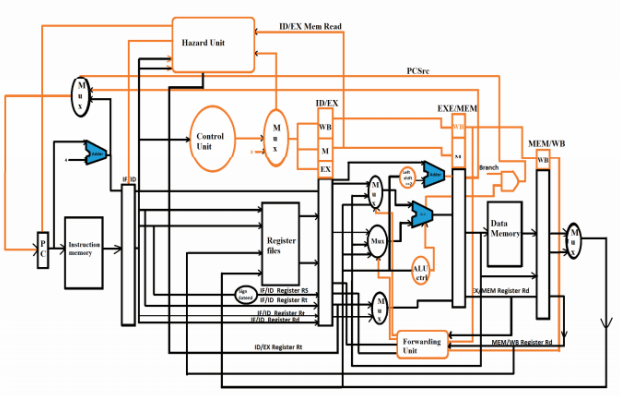



Fig5: MIPS Processor with forwarding unit

**Hazard Unit**

- Hazards happen because of instruction pipelining in CPU. An incorrect compilation result occurs when the next instruction cannot execute in the particular clock cycle.
- The control unit decides if Hazards will happen or not when an instruction is fetched. On the off chance that the Hazard will occur, at that point the control unit will embed no operation.

**Forwarding Unit**

The processor will execute millions of instructions per second, this leads to a problem called Interlocking in the pipeline stages. This can be solved by stalling the pipeline stages, i.e., Divide the pipeline into two parts, instruction fetch and instruction execute. To stall the pipeline ALU forwarding unit is used in order to make use of the ALU result directly. If the hazard occurs, the operands will come from either MEM/WB or EX/MEM pipeline registers. If there are no hazards, then register file will provide the operands for an ALU.

**Execution unit**

MIPS execution unit contains ALU, where operations are done based on opcode. Addition of program counter value to the sign extension unit, which is left shifted by two units the help us to get branch address. The sign extended unit increases the number by attaching the most significant bit in order to preserve the sign of a binary number. The control signals for ALU are produced by the ALU controller. ALU controller is a circuit has two inputs followed by an output, which is a two-bit data that tells ALU, which type of arithmetic and logical operation that ALU performs on the two-input data. The simulation results are obtained from Xilinx tool.

## 3. CONCLUSIONS

MIPS process is a best way to eliminate the hazards in original data path with the help of forwarding unit, where by fetching the results from the pipeline registers before they return back to the register record. Because of this processor will not attend high impedance or unknown state, which results in the performance enhancement. A 32-bit RISC-V Processor has been implemented with Harvard architecture and 5-stage pipelining structure. The RISC-V architecture is simulated in Cadence RTL Compiler and synthesized using Genus tool.

## REFERENCES

[1] G. Rajesh Babu, M. Bhanu Prakash, M. Vijaya Kumari, Ch. V. D. Ashok Kumar, G. Sai- "Design of 32-BIT ASYNCHRONOUS RISC-V PROCESSOR using Verilog"

[2] Saeid Moslehpour, Chandrasekhar Puliroju, Akram Abu-Aisheh – "Design of RISC Processor Using VHDL and Cadence"

[3] Chandran Venkatesan, Thabsera Sulthana M, Sumithra M.G, Suriya M – "Design of a 16-Bit Harvard Structure RISC Processor in Cadence 45nm Technology"

[4] Agineti Ashok, V. Ravi- "ASIC Design of MIPS Based RISC Processor for High Performance"

[5] Gokulan T, Akshay Muraleedharan, Kuruvilla Varghese- "Design of a 32-bit, dual pipeline superscalar RISC-V processor on FPGA"

[6] Etki Gür, Zekiye Eda Sataner, Yusuf H. Durkaya, Salih Bayar- "FPGA Implementation of 32-bit RISC-V Processor with Web-Based Assembler-Disassembler"

[7] Andrew S. Waterman- "Improving Energy Efficiency and Reducing Code Size with RISC-V Compressed"

[8] Shashi Kumar V1, Gurusiddayya Hiremath- "Low Power Implementation of RISC-V Processor"

[9] Pasquale Davide Schiavone, Davide Rossi Member, Alfio Di Mauro, Frank Gürkaynak, Timothy Saxe, Mao Wang, Ket Chong Yap, Luca Benini Fellow- "Arnold: an eFPGA-Augmented RISC-V SoC for Flexible and Low-Power IoT End-Nodes"

[10] Z Cao, Q Lv, Y Wang, M Wen, N Wu, C Zhang - "A Compression Instruction Set Design based on RISC-V for Network Packet Forwarding"

[11] Ben Keller, Martin Cochet, Brian Zimmer, Jaehwa Kwak, Alberto Puggelli, Yunsup Lee, Milovan Blagojevi´c, Stevo Bailey, Pi-Feng Chiu, Palmer Dabbelt, Colin Schmidt, Elad Alon, Krste Asanovi´c, and Borivoje Nikoli´c- "A RISC-V Processor SoC With Integrated Power Management at Sub microsecond Timescales in 28 nm FD-SOI"

[12] Matteo Perotti, Pasquale Davide Schiavole, Giuseppe Tagliavini, Davide Rossi, Tariq Kurd, Mark Hill, Liu Yingying, Luca Benini- "HW/SW approaches for RISC-V code size reduction"