iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** V **Month of publication:** May 2023

DOI: https://doi.org/10.22214/ijraset.2023.52205

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue V May 2023- Available at www.ijraset.com

Basic RISC-V Instruction Set Architecture: Design and Validation

Pavan P[1], Kamal P S[2], Govardhan G[3], Suresh Kumar V[4]
[1, 2, 3]*Students,* [4]*Assistant Professor, Electronics and Communication Department, Maturi Venkata Subba Rao Engineering College, Osmania University, Telangana, India.*

Abstract: ***This project's primary goal is to design and implement a simple RISC V instruction set architecture. This paper offers insights into the architecture of the risc v instruction set. This system employs the RISC V R-type (register) type instruction format. Using this format, we designed the fundamental isa and tested its functionality using verilog code. There is no licence fee for using RISC V, an open source isa that is available to everyone. Reduced instruction set (RISC) computers are created to make the individual instructions given to computers to perform various tasks more manageable. Most instruction set architectures, or isas, are proprietary and cannot be used or modified without permission from the companies; as a result, an isa that is free and open source, which is provided by risc v, will help in increasing the speed and lowering system costs by using these instruction set formats, and we are designing 32 bit isa architecture. instruction formats for the risc v instruction set architecture***

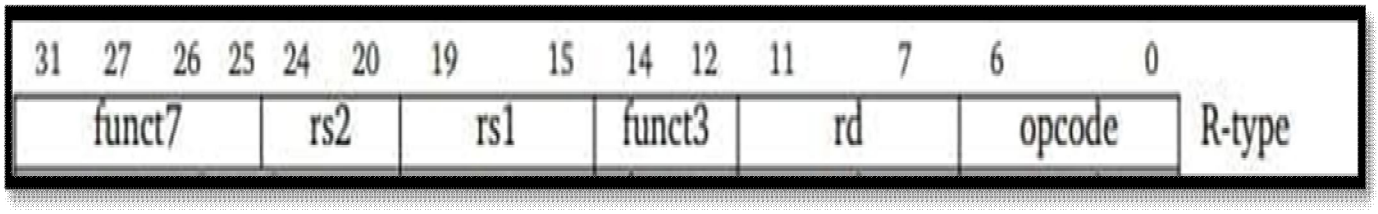
Keywords: RISC V, ISA, Instruction formats(R-type, J-type etc)

I. INTRODUCTION

RISC V is an open source architecture that anyone can use for free. RISC V offers various instruction formats for carrying out various operations. The suggested method here is the design of instruction architecture set that uses these formats and verifies the operations of these formats, such as addition, subtraction, and or xor, etc. These formats include R-(register) -type, J-(jump)-type, S-type, etc. We can use these instruction set formats for implementing different applications, but we must write different verilog code depending on the applications

II. PROPOSED SYSTEM

The r-type instruction format is performed using the suggested technique, where r stands for register, indicating that operations are carried out on the registers rather than memory locations. One of the instruction formats used in the risc-v isa is the r-type instruction format, which is used to execute arithmetic and logical instructions. The first field in the r-type instruction format is called the opcode field, which is also known as the operation code. The r-type instruction format consists of six subfields and has a fixed length of 32 bits.



1) The first field in the R - type instruction format is called as opcode field which can also be called as operation - code. This opcode is of 7-bit length (0 – 6) and this opcode is used to describe which type of instruction format is used such as R-type , I-type ,U-type etc.

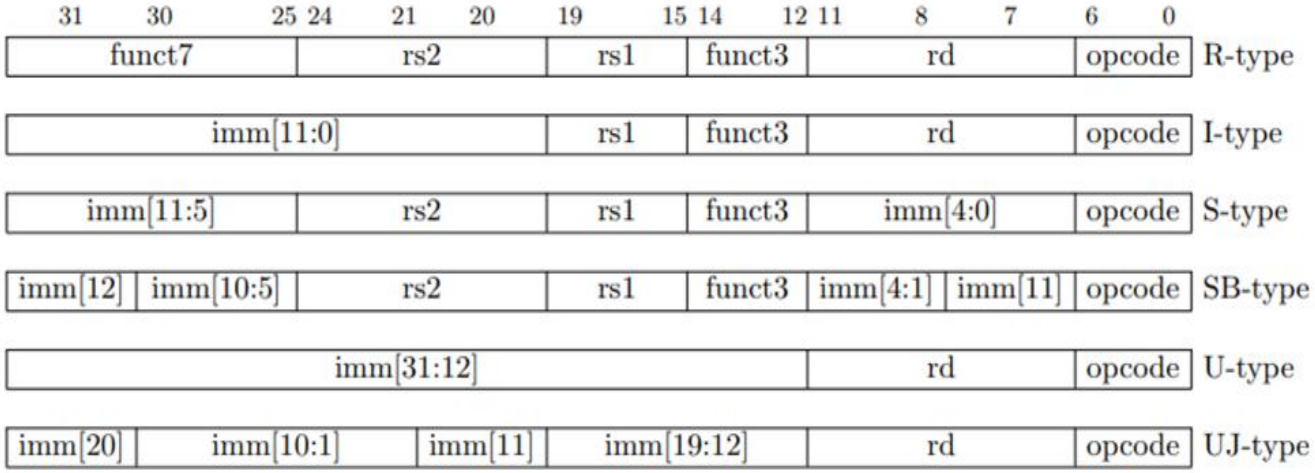
2) The next field in the R - type instruction format is called rd field. Rd here in the “rd field” stands for Destination Register. This rd field indicates the place where the result of the operation is stored .The length of the rd field is 5-bits (7-11).

3) funct3 (function - 3) is the field after the destination register .This field is 3 – bit length i.e. from bit 12 to bit 14 and it further tells about the operation, such as whether it is an addition, subtraction, or a logical operation that is being performed .

4) There are two source register namely rs1 and rs2 with the bit length of 5 bit each .rs1 is 5 bit length from 15 -19 and rs2 has bit length of 5 bit i.e. 20 -24.

5) The funct7 field is the last field of R -type instruction format .It describes the operation, such as whether it is a shift or multiplication operation etc. Both funct3 and funct7 are used to describe the operations depending on the input format given to them.

Below are examples of different types of instruction formats



Instruction formats define the pattern or structure used to represent instructions in a language that a computer can understand. The six basic formats that are supported by the RISC V isa are listed below, each of which has a different functionality.

A. R-type Format (Proposed Method)

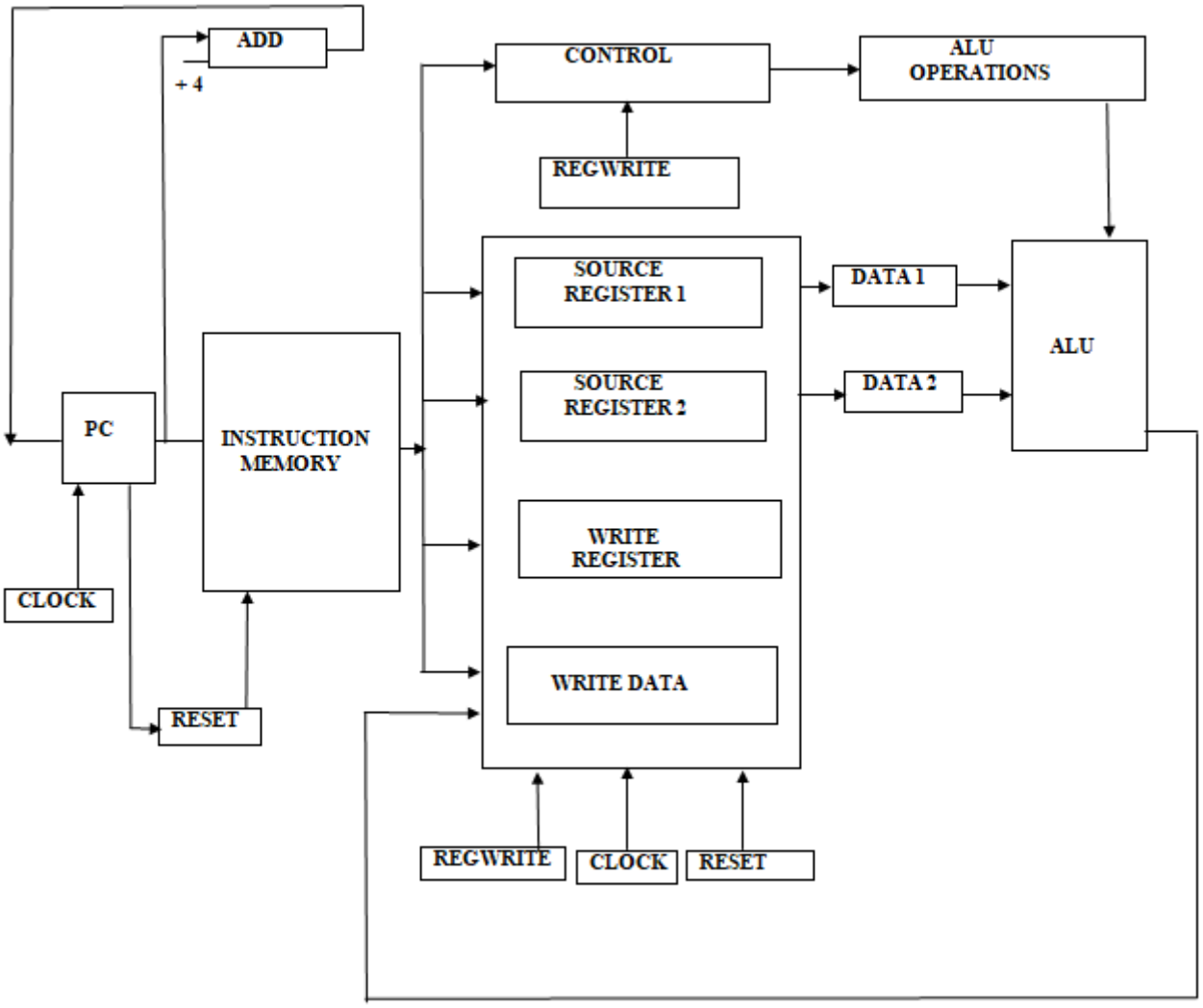
As was already mentioned, the r-type format is used to carry out various operations, including arithmetic and logical instructions. Due to the fact that these instructions operate on two register operands, they facilitate the effective execution of various operations. There are six sub fields in this r-type.

1) Opcode
2) Funct3
3) Funct7
4) rd (Destination Register)
5) rs1 (Source Register 1)
6) rs2 (Source Register 2)

The fundamental isa that we developed here implements and validates this R -type format.

- I-type format: This I-type format is used to carry out instructions that use register and immediate value operations.
- S-type format: Instructions that store a value from a register into memory are implemented using this s-type format.
- B-type format: The branch instructions that conditionally transfer control to a newly specified instruction address are the type of instructions that use the b-type format.
- U-type format: For performing instructions that load a 20- bit immediate value into a register, use the u-type format.
- J-type format: the j-type format is used to implement jump type instructions that unconditionally transfer control to a new instruction address.

III.BASIC INSTRUCTION SET ARCHITECTURE



A. Working

The RISC V typically has five stages, they are following:

- Fetch
- Decode
- Execute
- Memory
- Write back

Below is a brief explanation of each stage using the basic isa that we suggested in our project.

1) Fetch Stage

Program Counter

Program Counter, or PC, is this stage's first step. The address of the instruction to be executed is managed by the Program Counter, a unique register. It also goes by the name "Instruction pointer." The programme counter either increases or decreases in value as each instruction is fetched. The CPU will therefore fetch the instruction from the instruction memory based on the value of the program counter (PC). The output of the programme counter is passed to the adder when the PC is initially set to zero. After each operation, this adder increases the programme counter's value by 4 points. PC is increased here by +4 to indicate the next instruction.

2) Decode Stage

Instruction Memory

The programme counter's instructions are kept in the instruction memory. This stage involves reading the address provided by the programme counter and decoding the instruction. So, it is also known as the decoding stage.

This decoder stage describes the type of operation to be carried out and the various operands needed. It decodes the command and sends the information to the Execute stage, which is the following stage.

3) Execute Stage

Control

The memory used for instructions is used to At this point, the instructions are followed and carried out. This execution stage has several substages, including write reg, write data, rs1 (source register 1), and rs2 (source register 2). After the decoder has completed decoding the instructions, the data is separated and given to the various sub-stages. The write reg, or write register, is used to store the output, and the write data is used to display the output. The source registers store the input values provided by the decoder stage. The control receives the data and oversees subsequent action. Data will be obtained from instruction memory in this way.

4) Memory Stage

Because every instruction or piece of data that is used must be stored in memory, this stage is crucial. Thus, the storage facility is provided by this memory stage. Here, depending on the needs of the process, the data is read or written into memory. Due to the load-store mechanism used by the RISC V, data must be retrieved from memory before an operation can be carried out. Once the operation is complete, the result must be stored back in memory.

5) Write Back

ALU (Arithmetic And Logic Unit)

This ALU unit is in charge of carrying out various mathematical and logical operations. It receives information from three sources: rs1 and rs2, respectively, and the control unit, which specifies the type of operation to be carried out. What type of operation needs to be carried out is specified by the input from these source registers.

The data obtained after the various ALU operations are carried out is stored and displayed using the write back.

.

Here is an example of an R-Type instruction in RISC-V assembly language

INSTRUCTION	OPCODE
ADD	000000
SUB	000001
MUL	000101
AND	000010
OR	000011
SLT	000100

The RISC V instruction set architecture (ISA) offers a variety of instruction formats that we can use to create custom architectures and applications by writing Verilog code. The table that follows lists the various instruction formats and how they work.

B. RISC V Operations Carried out in the R-type Format

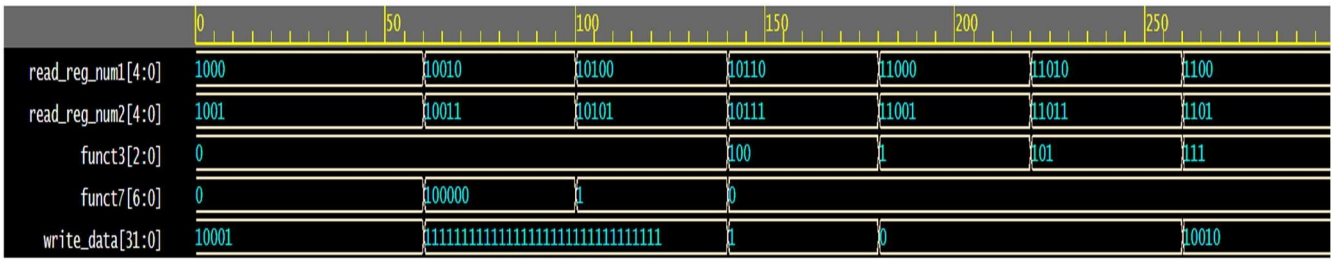
*** R-Type Instruction formats***

INSTRUCTION	NAME	FORMAT	OPCODE	FUNCT3	FUNCT7	DESCRIPTION
and	AND	R	0110011	0*7	0*00	rd= rs1 & rs2
or	OR	R	0110011	0*6	0*00	rd= rs1 | rs2
xor	XOR	R	0110011	0*4	0*00	rd= rs1 ^ rs2
Add	ADD	R	0110011	0*0	0*00	rd= rs1 + rs2
Sub	SUD	R	0110011	0*0	0*20	rd= rs1 - rs2
Sll	SHIFT LEFT LOGICAL	R	0110011	0*1	0*00	rd= rs1<< rs2
Srl	SHIFT RIGHT LOGICAL	R	0110011	0*5	0*00	rd= rs1 >>rs2
sra	SHIFT RIGHT ARITH*	R	0110011	0*5	0*20	rd= rs1>>s2
Slt	SET LESS THAN	R	0110011	0*2	0*00	rd= (rs1< rs2)?1:0
sltu	SET LESS THAN(U)	R	0110011	0*3	0*00	rd= (rs1< rs2)?1:0

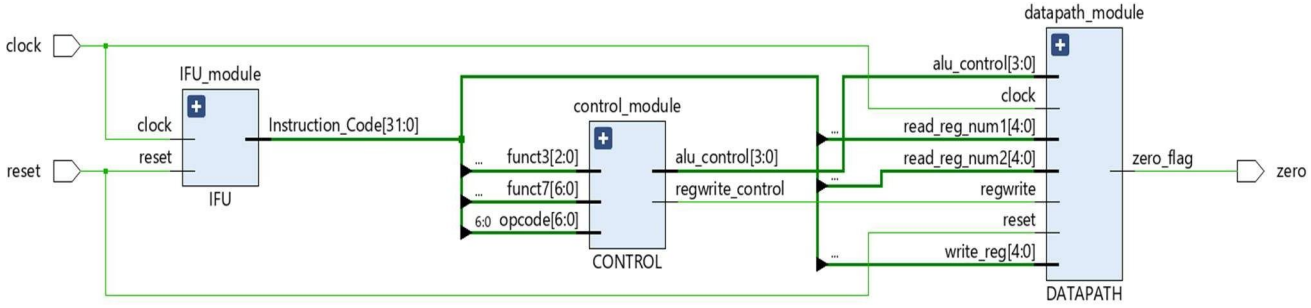
IV. RESULTS

A. R-Type Operation Results

We use the RISC V-provided R-type instructions, which perform operations like ADD, SUB, AND, OR, etc based on the data provided as input and then store and display the results in the write-back register.



Elaborated design output

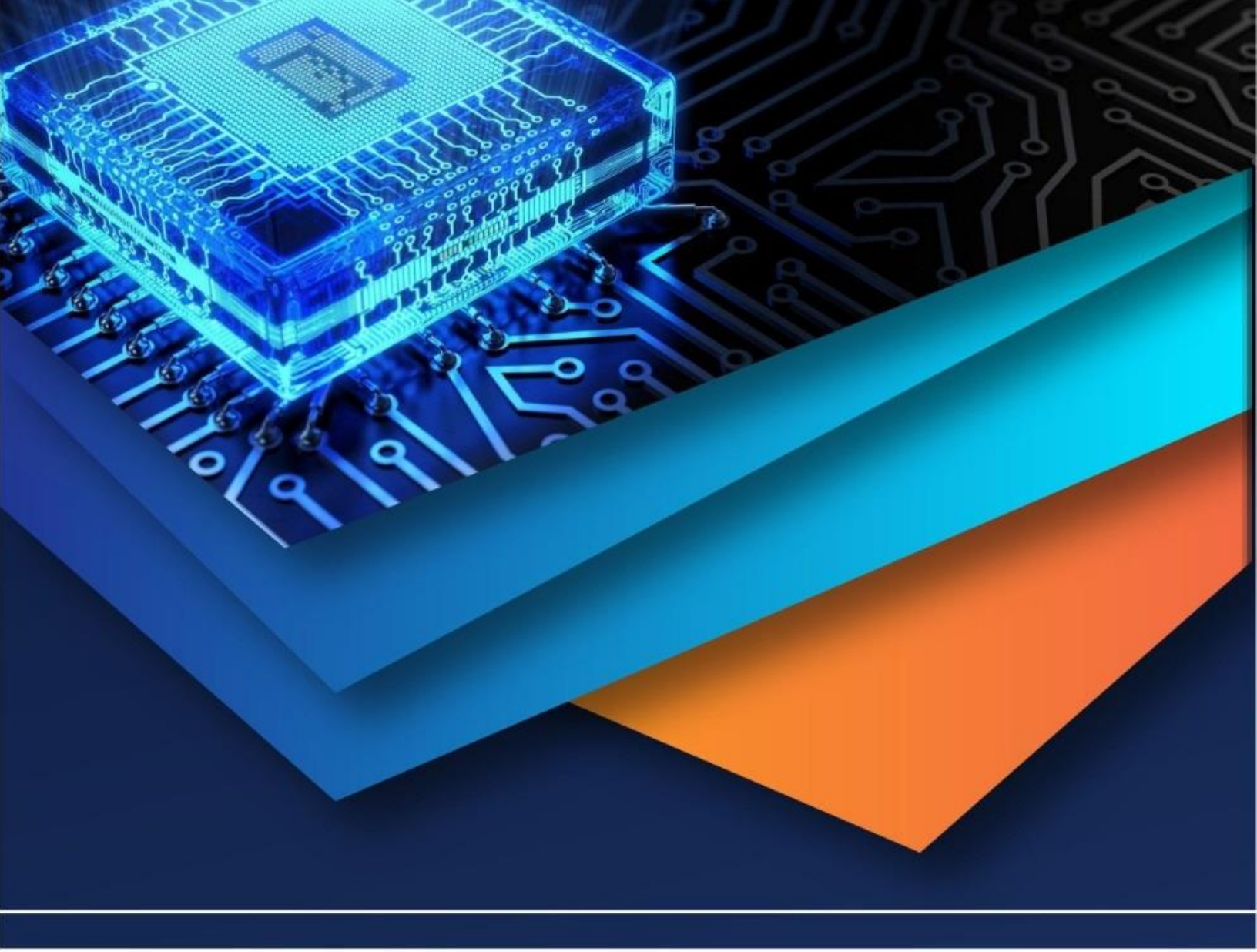


V. CONCLUSION

As mentioned earlier, RISC V is open source and freely available to anyone. We can use the instruction formats to design our own architecture or develop an application, and the source code of the RISC V can be modified depending on the type of application we develop. RISC V has many useful specifications that we have used to design a five stage 32-bit isa using verilog HDL and generate the RTL schematic diagram and perform the simulation. The main aim of this RISC V is to fulfill the software requirements with low cost and flexibility.

REFERENCES

[1] Digital design and computer architecture RISC – V edition by sarah L.Harris and David Harris.
[2] Computer organization and architecture by Baljinder Singh and Ikvinder Singh.
[3] Modern computer architecture and organization by Ledin Jim and Packet publishing limited.
[4] “The RISC-V Instruction Set Manual Volume I: User-Level ISA" by Andrew Waterman, Krste Asanović, and David Patterson.
[5] "The RISC-V Reader: An Open Architecture Atlas" by David Patterson and Andrew Waterman.
[6] "RISC-V Instruction Set Architecture (ISA) Overview" by Krste Asanović, available at https://riscv.org/isa-extensions/overview/.
[7] "RISC-V ISA Specification" by the RISC-V Foundation, available at https://riscv.org/specifications/.
[8] "A Scalable Vector Extension for RISC-V" by Andrew Waterman.
[9] ."The RISC-V Instruction Set Architecture: A Review" by Yunsup Lee.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY