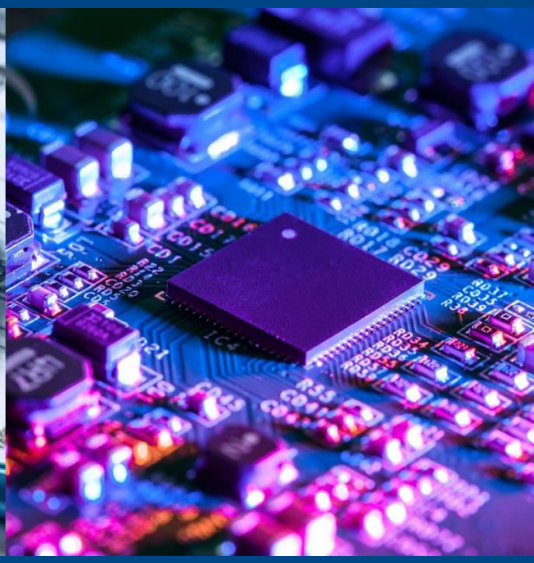
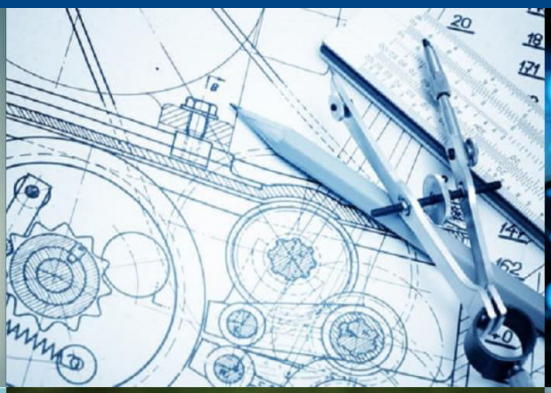
ISSN: 2582-7219

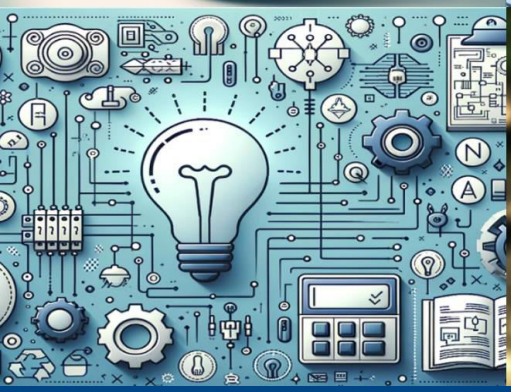




International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206 **Volume 9, Issue 4, April 2026**

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904015

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Design and Simulation of 5-Stage Pipelined RISC-V (RV32I) Processor

Ambati. Sriharshitha, Anabathina. Ramya, Kallagunta. Hema Sree, Kamineni. Hema Bindu, Shaik. Riyazuddien

UG Student, Department of ECE, Vasireddy Venkatadri International Technological University, Andhra Pradesh, India

UG Student, Department of ECE, Vasireddy Venkatadri International Technological University, Andhra Pradesh, India

UG Student, Department of ECE, Vasireddy Venkatadri International Technological University, Andhra Pradesh, India

UG Student, Department of ECE, Vasireddy Venkatadri International Technological University, Andhra Pradesh, India

Professor, Department of ECE, Vasireddy Venkatadri International Technological University, Andhra Pradesh, India

ABSTRACT: This project presents the design and simulation of a 5-stage pipelined RISC-V (RV32I) processor implemented using Verilog HDL. The motivation behind this work is the increasing demand for efficient, open-source processor architectures in academic and embedded system applications. The primary objective is to develop a functional processor core that demonstrates key pipelining concepts such as instruction-level parallelism and performance improvement through overlapping execution stages. The proposed design includes the five standard pipeline stages: Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB). To enhance performance and correctness, important mechanisms such as data forwarding, load-use hazard detection with stalling, and branch handling with pipeline flushing are implemented. A small instruction memory and data memory are integrated to execute a sequence of instructions including arithmetic, load/store, and branch operations. The processor is simulated using a testbench in Verilog to validate functionality. The results demonstrate correct execution of instructions with proper handling of hazards and pipeline control. Performance metrics such as cycle count, instruction count, and Cycles Per Instruction (CPI) are calculated to evaluate efficiency. The design shows improved throughput compared to a non-pipelined approach and serves as a foundational model for understanding modern processor architectures. This work can be further extended for advanced features such as branch prediction and superscalar execution.

KEYWORDS: RISC-V, RV32I Architecturre, 5-Stage Pipelined Processor, Verilog HDL, Pipeline Architecture, Hazard Detection and Resolution, Pipeline Control (Stall and Flush), Cycles Per Instruction (CPI)

I. INTRODUCTION

The field of computer architecture has evolved significantly from early stack-based and complex instruction set designs to more efficient and optimized architectures such as Reduced Instruction Set Computing (RISC). RISC architecture emphasizes simplicity by using a limited set of instructions, fixed instruction formats, and faster execution cycles. RISC-V is a modern, opensource Instruction Set Architecture (ISA) developed to overcome the limitations of proprietary architectures. Unlike commercial ISAs such as ARM and Intel, which require costly licensing and restrict customization, RISC- V is freely available, making it highly suitable for academic research, industrial applications, and innovation in embedded and high-performance systems.

In earlier processor designs, single-cycle architectures were widely used, where each instruction is executed in a single clock cycle. Although simple and easy to design, this approach suffers from performance inefficiencies because the clock cycle duration is determined by the slowest instruction. As a result, faster instructions also take the same amount of time, leading to underutilization of system resources. To address this limitation, pipelining was introduced as a key technique to improve processor performance by allowing multiple instructions to overlap during execution.A pipelined processor divides the instruction execution process into multiple stages, typically

Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB). Each

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

stage performs a specific function, and pipeline registers are used to store intermediate results between stages. This enables concurrent execution of multiple instructions, significantly increasing throughput. However, pipelining introduces challenges such as data hazards and control hazards, which must be properly managed to ensure correct execution. Techniques such as data forwarding, stalling, and branch handling are used to overcome these issues and maintain pipeline efficiency.

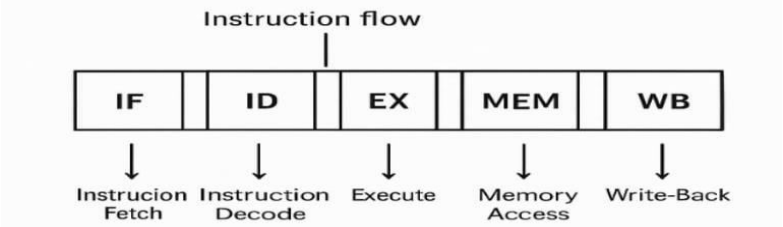


Figure 1.1: Five-Stage Pipeline Instruction Flow in a RISC-V Processor.

In this project, a 5-stage pipelined RISC-V processor based on the RV32I instruction set is designed and implemented using Verilog HDL. The design includes key components such as the Arithmetic Logic Unit (ALU), register file, instruction memory, data memory, and pipeline registers.Hazard handling mechanisms, including forwarding logic, load- use hazard detection, and branch control with pipeline flushing, are incorporated to ensure accurate and efficient operation. The processor is verified through simulation using a testbench, and performance metrics such as cycle count, instruction count, and Cycles Per Instruction (CPI) are evaluated. This project highlights the advantages of pipelining in modern processor design and provides a strong foundation for future enhancements such as cache integration and advanced control mechanisms.

II. LITERATURE REVIEW

In existing systems, RISC-V processors are commonly implemented using a single-cycle architecture, where each instruction is executed completely within one clock cycle. In this design, all stages of instruction execution—Instruction Fetch, Decode, Execute, Memory Access, and Write Back— are performed sequentially in a single cycle. While this approach simplifies the design and control logic, it limits the overall performance of the processor.

The architecture consists of two main components: the datapath and the control unit. The datapath includes components such as the Program Counter (PC), instruction memory, register file, ALU, and data memory, which are responsible for executing instructions. The control unit generates control signals based on the instruction opcode to direct the operation of the datapath.In this system, the instruction is first fetched from instruction memory using the Program Counter, then decoded to determine the required operation and operands. The ALU performs the necessary arithmetic or logical computation. For load and store instructions, data memory is accessed to read or write data, and finally, the result is written back to the register file. Since all these operations are completed within a single clock cycle, the clock period must be sufficiently long to accommodate the slowest instruction, particularly memory access operations.

As a result, single-cycle processors suffer from limitations such as long critical path delay, inefficient hardware utilization, and increased design complexity. Only one instruction is executed at a time without overlap, which reduces throughput and overall performance. These drawbacks highlight the need for pipelined architectures, which improve performance by enabling parallel execution and better resource utilization.

III. CIRCUIT DESIGN

a) Overall Architecture

The proposed processor is a 5-stage pipelined design based on the RISC-V (RV32I) architecture, implemented using Verilog HDL. The design follows the principle of instruction pipelining, where multiple instructions are executed in an overlapped manner different stages. The five stages—Instruction Fetch (IF), Instruction Decode (ID), Execute (EX),

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904015

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Memory Access (MEM), and Write Back (WB)—are connected through pipeline registers (IF/ID, ID/EX, EX/MEM, MEM/WB). These registers store intermediate results and ensure proper synchronization between stages, thereby enabling continuous instruction flow and improved throughput.

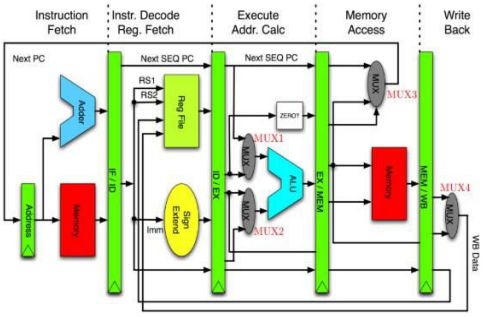


Figure 2.1: Architecture of a 5-Stage Pipelined RISC-V Processor Datapath

b) Instruction Fetch (IF)

The IF stage is responsible for fetching instructions from instruction memory using the Program Counter (PC). The PC holds the address of the current instruction and is incremented by 4 after each fetch to point to the next instruction. In case of branch instructions, the PC is updated with the computed branch target address. The fetched instruction and the current PC value are stored in the IF/ID pipeline register. This stage ensures a steady supply of instructions to the pipeline and plays a crucial role in maintaining pipeline efficiency.

c) Instruction Decode (ID)

In the decode stage, the fetched instruction is analyzed to
identify opcode, source registers (rs1, rs2), destination register (rd), and immediate values. The register file provides operand values corresponding to the source registers. Immediate values are generated and sign-extended for use in later stages. This stage also includes a hazard detection unit, which checks for dependencies between instructions, particularly load-use hazards. If such a hazard is detected, a stall signal is generated to pause the pipeline and maintain correct execution. The decoded information is stored in the ID/EX pipeline register..

d) Execute (EX):

The execute stage performs arithmetic and logical operations using the Arithmetic Logic Unit (ALU). It also calculates effective addresses for memory operations and evaluates branch conditions. To minimize delays caused by data dependencies, a forwarding unit is implemented, which selects operands from later pipeline stages (EX/MEM or MEM/WB) instead of waiting for write-back. This reduces pipeline stalls and improves performance. The results of Aoperations, along with control signals, are stored in the EX/MEM pipeline register for further processing.

e) Memory Access (MEM):

The MEM stage handles interactions with data memory. For load (lw) instructions, data is read from memory, and for store (sw) instructions, data is written into memory. The memory address is obtained from the ALU output generated in the EX stage. This stage ensures proper data transfer between processor and memory. The output data or ALU result is then passed to the MEM/WB pipeline register.

f) Write Back (WB):

The Write Back stage is the final stage of the pipeline where the result of an instruction is stored in the register file. The value produced either by the ALU (from the Execute stage) or fetched from memory (from the Memory Access stage) is written into the destination register. This makes the result available for future instructions and marks the completion of the instruction execution

.

IV. EXPERIMENTAL FINDINGS AND ANALYSIS

The simulation results of the 5-stage pipelined RISC-V (RV32I) processor validate both functional correctness and theoretical concepts of pipelined execution. In theory, pipelining improves processor performance by overlapping

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904015

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

instruction execution, thereby increasing instruction-level parallelism (ILP). This behavior is clearly observed in the waveform, where multiple instructions are simultaneously present in different stages (IF, ID, EX, MEM, WB). The absence of undefined (X) states and the smooth propagation of signals across pipeline registers confirm proper synchronization and stable datapath design.

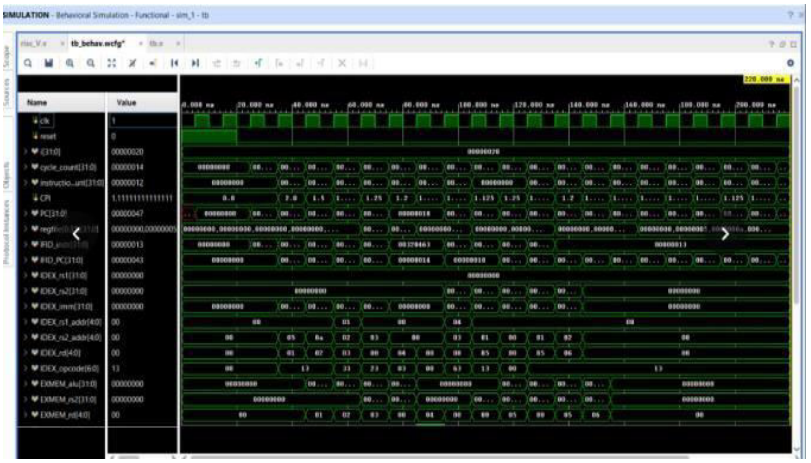


Figure 4.1: Simulation results

Figure 4.2: Simulation of 5-Stage Piplelined results

g) Functional Results

The observed register values (x1 = 5, x2 = 10, x3 = 15, x4 = 15) match the expected results based on the instruction sequence. The correct execution of addi, add, sw, lw, and beq instructions demonstrates that the processor correctly implements RISC-V instruction semantics. Theoretically, each instruction completes after passing through all five stages, and the results confirm that data is correctly written back to the register file. The successful memory read/write operations validate the proper interaction between the processor and data memory.x1 = 5 (from addi x1, x0, 5) x2 = 10 (from addi x2, x0, 10) x3 = 15 (from add x3, x1, x2) x4 = 15 (from lw x4, 0(x0))Additionally, memory location data_mem[0] stores the correct value 15, verifying proper store (sw) and load (lw) operations. The branch instruction (beq) is successfully executed, as seen from the pipeline behavior where the next instruction is skipped, confirming correct control flow handling.

h) Pipeline Operation Analysis

According to pipeline theory, once the pipeline is filled, one instruction completes per clock cycle, significantly improving throughput. The waveform reflects this behavior, where instructions advance stage-by-stage in each clock cycle. The theoretical concept of pipeline latency vs throughput is also observed: although the first instruction takes multiple cycles to complete, subsequent instructions complete faster due to overlapping execution.

© 2026 IJMRSET | Volume 9, Issue 4, April 2026| DOI:10.15680/IJMRSET.2026.0904015

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

i) Hazard Handling Results

The signals in the waveform confirm correct hazard handling:

- **Forwarding signals (forwardA, forwardB)** are activated when dependencies occur.
- **Stall signal** is asserted during load-use hazard conditions.
- **Branch_taken signal** is triggered correctly, and pipeline flushing is observed.

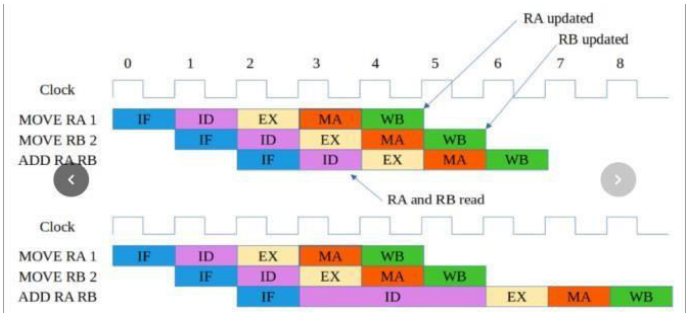


Figure 4.3: Operational Analysis

These results indicate that hazards are properly detected and resolved without affecting correctness.

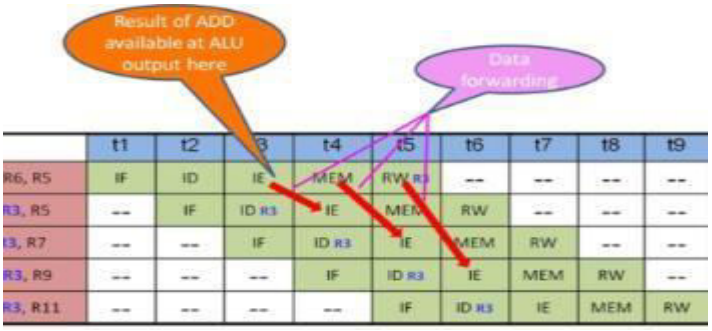


Figure 4.4: Hazard Handling

j) Performance Results

From the simulation:

Cycle Count ≈ 14 Instruction Count ≈ 12 CPI ≈ 1.11

The CPI value close to 1 indicates efficient pipeline performance. Although minor stalls and branch penalties are present, the processor achieves significantly improved throughput compared to a non-pipelined design.

k) Overall Theoretical Findings

- Pipelining increases throughput by enabling parallel instruction execution
- Pipeline registers ensure stage isolationand synchronization
- Hazard handling mechanisms are essential for maintaining correctness
- CPI serves as a key metric for evaluating pipeline efficiency
- The design closely matches theoretical expectations of a 5-stage pipeline.

Analysis

The experimental results strongly align with theoretical principles of pipelined processor design. The implementation successfully demonstrates how instruction-level parallelism, hazard management, and pipeline control contribute to improved processor performance, making it a practical realization of modern CPU architecture concepts.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

V. CONCLUSION

The project focuses on the design and implementation of a 5- stage pipelined RISC-V processor using Verilog HDL. The pipelined architecture improves system performance by enabling the parallel execution of multiple instructions across different stages. This results in 42 higher throughput and efficient utilization of hardware resources compared to single- cycle designs.

The processor is designed with key components such as the Program Counter, Instruction Memory, Register File, ALU, Data Memory, Control Unit, and Pipeline Registers. Additionally, hazard handling techniques such as data forwarding, load-use stalling, and branch handling are implemented to ensure correct execution of instructions and maintain pipeline efficiency.The design is successfully simulated using Vivado, and the results confirm correct execution of arithmetic operations, memory access, and branch instructions. The pipeline stages operate simultaneously, and hazard handling mechanisms effectively resolve data and control dependencies. In conclusion, the pipelined RISC-V processor provides a high- performance and efficient solution for modern computing applications. Although it introduces additional design complexity, the significant improvement in speed and throughput makes it more suitable than single-cycle processors for real-time and high-speed systems.

VI. FUTURE SCOPE

The proposed 5-stage pipelined RISC-V processor offers several opportunities for improvement in performance and efficiency. Techniques such as pipeline balancing, reducing stalls, and advanced hazard handling methods like dynamic branch prediction and out-of-order execution can enhance overall execution. Additionally, extending the instruction set can enable better support for specialized applications such as digital signal processing, cryptography, and embedded systems.

Further enhancements can be made in memory and architecture. Integrating instruction and data cache memory can significantly reduce access time and improve speed, while extending the design to multicore architectures allows parallel task execution for higher performance. Power optimization methods such as clock gating, power gating, and dynamic voltage scaling can also make the processor more energy- efficient and suitable for low-power applications.
Finally, implementing advanced control flow techniques can reduce performance loss due to branching. The processor can be deployed on FPGA platforms or developed into ASICs for real-world applications. Improving toolchain support and integrating the processor with modern technologies like IoT systems and AI accelerators will further enhance its usability and relevance.

REFERENCES

1. Sarah L. Harris and David M. Harris, Digital Design and Computer Architecture: RISC-V Edition, Morgan Kaufmann Publishers, 2017.
2. Hyogeun Kim et al., Single Cycle 32-bit RISC-V ISA Implementation and Verification, International Journal of Engineering Research, 2018.
3. Don K.D., Ayushi P., Virk S.S., Sajal A., Tanuj S., Arit M., Kailash C.R., Single Cycle RISCV Microarchitecture Processor and its FPGA Prototype, Proceedings of the 7th International Symposium on Embedded Computing and System Design, India, 2017.
4. Andrew Waterman, Yunsup Lee, David Patterson, Krste Asanović, The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA, EECS Department, University of California, Berkeley, 2011
5. Krste Asanović et al., The Rocket Chip Generator, Technical Report UCB/EECS-2016- 17, EECS Department, University of California, Berkeley, 2016.
6. Clifford Wolf, PICORV32 – A Size-Optimized RISC-V CPU [Online]. Available: https://github.com/cliffordwolf/picorv32.
7. Dennis A.N.G. and Kwangki R., Selecting a Synthesizable RISC-V Processor Core for Low-Cost Hardware Devices, Journal of Information Processing Systems, 2019.
8. Kim W.Y., The Linux RISC-V Singularity in the CPU World [Online]. Available: https://zdnet.co.kr/view/?no=20201201123035.
9. Swarup Bhunia, Michael S. Hsiao, Mainak Banga, Seetharam Narasimhan, Hardware Trojan Attacks: Threat Analysis and Countermeasures, Proceedings of the IEEE, 2014.

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 8.206| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

10. SiFive Inc., RISC-V Based Processor Developments, [Online]. Available: http://www.thelec.net/news/articleView.html?idxno=1584
11. Patterson, David A., and John L. Hennessy, Computer Organization and Design: The Hardware/Software Interface (RISC-V Edition), Morgan Kaufmann, 2017.
12. Waterman, Andrew, and Krste Asanović, The RISC- V Reader: An Open Architecture Atlas, Strawberry Canyon LLC, 2016.
13. Hennessy, John L., and David A. Patterson, Computer Architecture: A Quantitative Approach, 6th Edition, Morgan Kaufmann, 2019.
14. Yunsup Lee, Design and Implementation of the Rocket Chip RISC-V Processor, University of California, Berkeley, 2016.
15. Mano, M. Morris and Michael D. Ciletti, Digital Design, 5th Edition, Pearson, 2013.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com