International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 06 | Jun 2026 www.irjet.net p-ISSN: 2395-0072

Design and Implementation of 32-bit RISC-V Processor with Floating-Point Coprocessor and Clock Gating Optimization

[1]Bhagyalakshmi Chinivalar, [2]Dr M S Suma

[1]MTech Student, [2]Professor, Dept of ECE,BMS College of Engineering, Bengaluru, India.

Abstract-*This project presents the design and implementation of a 32-bit RISC-V processor integrated with a dedicated Floating-Point Coprocessor for high-performance floating-point computation. The core processor is based on the RV32I instruction set architecture, while the floating-point operations are offloaded to a separate IEEE-754 compliant Floating-Point Coprocessor, enabling efficient mixed-precision computing. The architecture employs a tightly coupled coprocessor interface that allows the main RISC-V core to dispatch floating-point instructions to the coprocessor and handle data exchange through dedicated registers and control signals. This modular approach provides better hardware separation, improved scalability, and easier maintenance compared to a unified execution unit. The design includes optimized modules for instruction fetch, decode, register file, integer ALU, control unit, hazard detection, and pipeline management in the main core, along with a complete floating-point coprocessor consisting of FP-ALU, floating-point register file, and coprocessor control logic. The entire system is implemented in Verilog HDL. It has been simulated, synthesized, and tested on an FPGA platform using the Vivado design suite. Pipeline hazards are effectively managed using forwarding, stalling, and hazard detection techniques to ensure smooth operation between the main processor and the floating-point coprocessor. Experimental results demonstrate significant performance gains in floating-point intensive workloads while maintaining low resource utilization on the FPGA. The proposed design offers an excellent balance between performance, flexibility, and hardware cost, making it highly suitable for embedded systems, IoT devices, signal processing, and other mixed integer-floating-point applications.*

Keywords-RISC-V, Coprocessor, IEEE-754, Clock Gating, FPGA, Verilog HDL, Vivado.

I. INTRODUCTION

The RISC-V architecture is an open-source Instruction Set Architecture (ISA) that is widely used in academics, research, and industry due to its flexibility, simplicity, and free licensing model. Unlike traditional proprietary processor architectures, RISC-V allows designers to modify, extend, and optimize the processor based on application needs. This makes it an ideal platform for designing custom processors for embedded systems, digital signal processing, and machine learning applications. In this project, a RISC-V 5-stage pipelined processor is designed and implemented using Verilog HDL and synthesized on Xilinx Vivado. The five pipeline stages Instruction Fetch (IF), Instruction Decode (ID), Execute (EX), Memory Access (MEM), and Write Back (WB)—allow multiple instructions to be processed in parallel, improving overall performance. The final design is implemented in Verilog HDL and tested using simulation and synthesis in the Vivado design environment, making it suitable for deployment on FPGA platforms. This processor design demonstrates improved performance in applications involving real-time signal processing, data analysis, and embedded intelligent systems, while maintaining the modularity and openness of the RISC-V architecture.

II. LITERATURE SURVEY

A rigorous review of contemporary architectural exploration establishes the technical contextual baseline for this study:

1) Banna Saif et al.: Implemented an FPGA-based educational RISC-V processor (RV32I) with a simple pipeline, emphasizing simplicity and hands-on learning for embedded applications. This addresses accessible soft-core configurations for basic controls but lacks single-precision numeric arithmetic acceleration networks.

2) Saussereau et al.: Proposed AsteRISC, a size-optimized RISC-V core for design-space exploration, focusing on minimal area and efficient resource usage. The core focuses heavily on optimizing area footprints, establishing structural benchmarks for resource-constrained layouts at the cost of execution performance for data-intensive calculations.

3) Thanga Dharsni et al.: Presented a hazard-free pipelined RV32I RISC-V architecture, ensuring correct instruction execution without stalls for improved performance. Their techniques rely on optimized forwarding logic, minimizing structural bubbles across basic mathematical workflows.

4) Phangestu et al.: Implemented a five-stage pipelined 32-bit RISC-V soft processor in VHDL, supporting modular, FPGA-compatible educational and research applications, laying out classical instruction overlapping conventions.

5) Gupta et al.: Designed a high-speed, resource-efficient UART for reliable serial communication, suitable for peripheral integration with embedded processors like RISC-V, defining high-frequency clock divider methodologies.

III. THEORETICAL BACKGROUND & BASELINE ARCHITECTURE

RISC-V is an open-source Reduced Instruction Set Computing (RISC) instruction set architecture (ISA) that emphasizes simplicity, modularity, and extensibility. Developed at UC Berkeley, it is designed to support modern processor requirements while remaining straightforward enough for research, education, and custom hardware development. Simplicity and Regularity are preserved via fixed-length 32-bit instructions (RV32I) which make instruction decoding simple, combined with a small set of general-purpose registers (32 registers × 32-bit) for integer operations. It utilizes a Load/Store Architecture where only explicit load and store instructions access memory, while all arithmetic and logic operations operate exclusively on registers. Fixed instruction length and simple instruction formats make it ideal for pipelined implementations. RISC-V supports several instruction types, each optimized for a specific purpose: R-type (Register) for arithmetic/logic operations, I-type (Immediate) for arithmetic immediates and loads, S-type for store configurations, B-type for conditional branches, U-type for upper immediates, and J-type for long jumps. Registers x0–x31 represent the 32 general-purpose registers, where x0 is hardwired to zero, x1 (ra) serves as the return address, and x2 (sp) operates as the stack pointer.

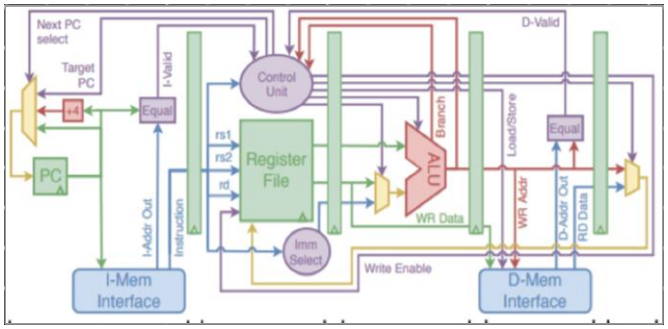

Fig. 1. Core system architectural block diagram for the baseline 5-stage pipelined integer processor.

A pipeline architecture increases performance by executing multiple instructions simultaneously, each in different processing stages. The commonly used 5-stage pipeline (IF, ID, EX, MEM, WB) balances simplicity and performance. Research has shown that pipelining improves throughput without significantly increasing hardware complexity. However, pipelining also introduces challenges such as data hazards, control hazards, and structural hazards, which require hazard detection, stalling, and forwarding mechanisms to maintain correct execution. Although the existing 5-stage pipelined RISC-V processor based on the RV32I instruction set provides an efficient and resource-optimized solution for integer-based embedded applications, it suffers from significant functional limitations when applied to modern real-world workloads. The absence of floating-point computation support restricts the processor to integer-only operations, making it unsuitable for applications involving signal processing, scientific computation, sensor data analysis, and control systems that inherently require floating-point arithmetic. Furthermore, implementing floating-point operations entirely in software on an integer-only core results in increased execution latency, higher power consumption, and reduced system throughput.

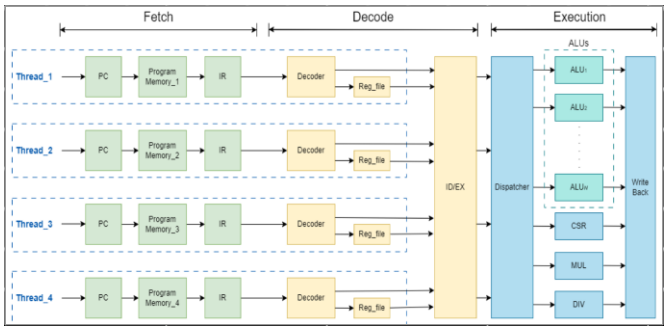

Fig. 2. Pipelined instruction sequencing and execution overlap mapping across the five structural stages.

VI. PROPOSED METHODOLOGY & ARCHITECTURAL TRAJECTORIES

A. Phase 1: Switchable Parallel Floating-Point Unit Architecture

Before achieving the final goals of implementing a floating point ALU as an independent coprocessor, we first implemented a floating point ALU as a separate parallel block inside the same RISC-V processor in a switchable floating point methodology. This parallel arrangement ensured functional modularity, low hardware complexity, and seamless operation when floating-point capability is enabled.

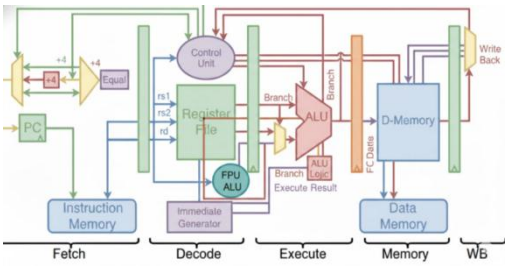

Fig. 3. Logical configuration of the 32-bit RISC-V pipeline with an integrated switchable parallel FP-ALU execution core.

The FP-ALU in this architecture supports the complete RV32F instruction set, including addition, subtraction, multiplication, division, floating-point to integer conversions, comparisons, and optional fused multiply-add operations. Internally, the FP-ALU is designed with IEEE-754 compliant data paths that consist of pre-normalization logic for operand alignment, multi-cycle arithmetic units

for multiplication and division, post-normalization stages for rounding, and exception-flag generation hardware. A multiplexer selects between the integer ALU output and the FP-ALU output in the Execute stage, allowing the Write Back stage to remain unchanged regardless of the operational mode of the FP unit. When floating-point support is disabled, the FP-ALU can be clock-gated or power-gated, reducing dynamic power consumption.

B. Phase 2: Decoupled Floating-Point Coprocessor Topology

To eliminate the execution routing overhead of shared resource grids, a decoupled coprocessor model was developed. The proposed system integrates an efficient Floating-Point Unit (FPU) coprocessor with the main RISC-V processor. The main goal is to accelerate high-precision floating-point arithmetic operations while reducing the workload on the main RISC-V core, resulting in significantly improved overall system performance.

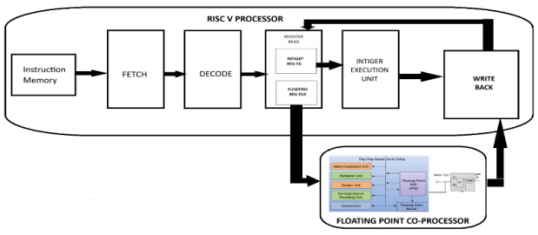

Fig. 4. Interconnect topology and isolated structural domain mapping of the Floating-Point Coprocessor system.

In this design, switching between execution units is performed cleanly via hardware opcode parsing in the Decode stage. Main integer instructions (**OP=0110011**, **OP-IMM=0010011**) route down the host pipeline, while floating-point paths (**OP-FP=1010011**, **LOAD-FP=0000111**,

STORE-FP=0100111) are dispatched across the handshake interface.

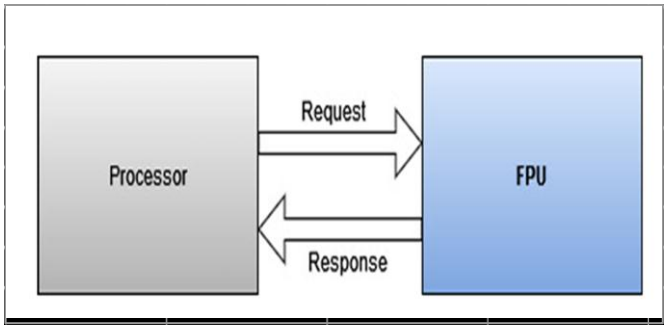

Fig. 5. Command issue and response validation handshake boundary protocol (X-IF Interface).

Modern RISC-V designs use a standardized Coprocessor eXtension Interface (X-IF) for clean integration. The signals from Core to FPU comprise **x_issue_valid** and **x_issue_ready** alongside instruction tokens, whereas signals from FPU to Core encompass **x_result_valid**, **x_result_ready**, output data, and updated exception flags. This handshake support enables decoupled execution, allowing the FPU to operate ahead of the main pipeline when there are no data dependencies.

C. Fine-Grained Clock Gating Mechanism

The design incorporates flip-flop-based clock gating to reduce power consumption. When a functional unit is not in use, its clock signal is disabled, preventing unnecessary switching activity and lowering dynamic power usage. It uses a combination of a flip-flop and logic gates to generate a glitch-free gated clock. The enable signal is synchronized with the system clock using a negative-edge flip-flop. The gated clock is then selectively distributed to modules such as the adder, multiplier, and registers. Only active modules receive the clock signal at any time, which significantly improves energy efficiency, especially in applications where floating-point operations are used intermittently.

CLK_{gated} = CLK_{master} ext{ AND } Q_{FF

V. EXPERIMENTAL RESULTS AND HARDWARE REPORTS

Comprehensive layout implementation, physical synthesis, and dynamic power estimation were conducted within Xilinx Vivado to capture the strict operational limits of the three processing paradigms.

A. Paradigm I: Integrated Switchable FPU System Reports

The layout synthesis waveforms and physical reports for the parallel integrated design are detailed below. Due to routing resource competition within the unified Execution block, significant signal propagation overheads are observed.

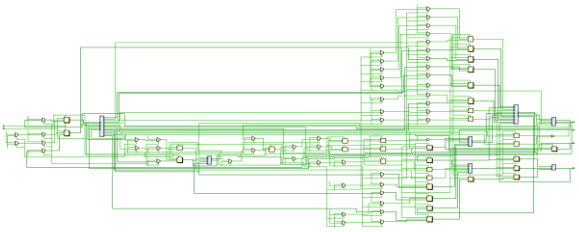

Fig. 6. Synthesized RTL schematic plot for the parallel switchable FPU system layout.

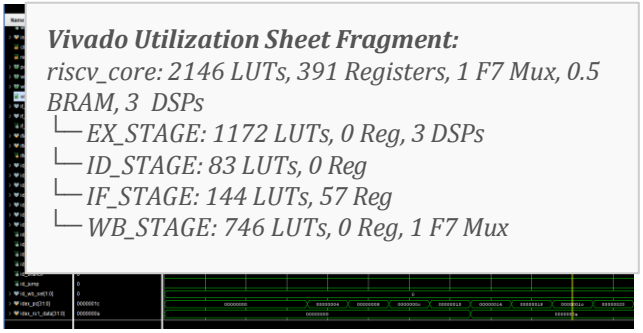

Fig. 7. Verification timing waveforms for instruction parsing under the parallel switchable design.

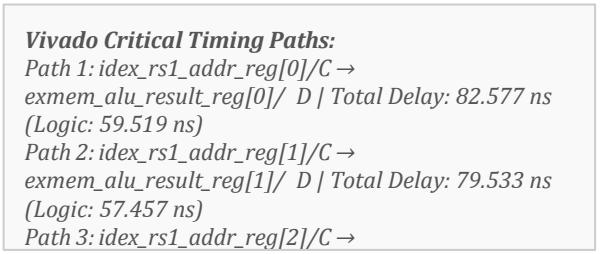

Fig. 8. FPGA Area utilization log sheet for the integrated parallel switchable FPU configuration.

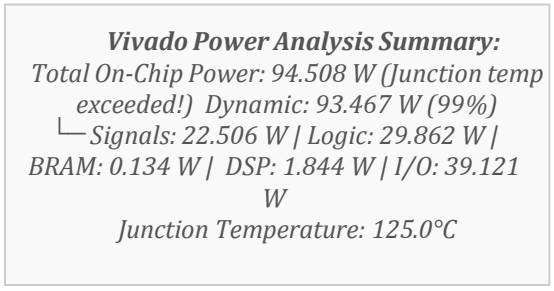

Fig. 9. Synthesized total power dissipation spectrum for the parallel switchable baseline layout.

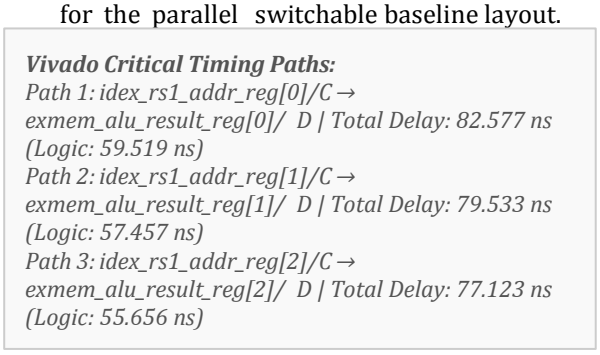

Fig. 10. Propagation delay critical-path report tracking within the parallel switchable execution unit.

B. Paradigm II: Decoupled Floating-Point Coprocessor Reports

By moving the hardware blocks onto an independent coprocessor extension bus, the shared integer execute multiplexers are bypassed. The corresponding structural schematics and hardware reports are presented below.

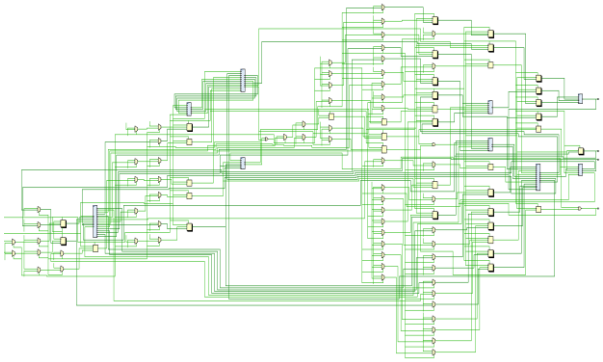

Fig. 11. Synthesized RTL schematic plot for the decoupled coprocessor system setup.

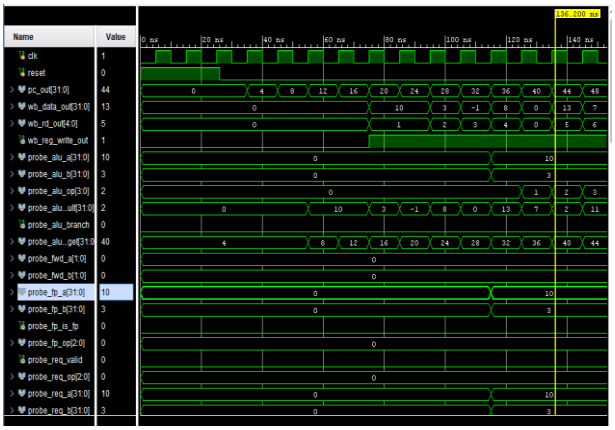

Fig. 12. Handshake control validation waveforms for decoupled coprocessor bus cycles.

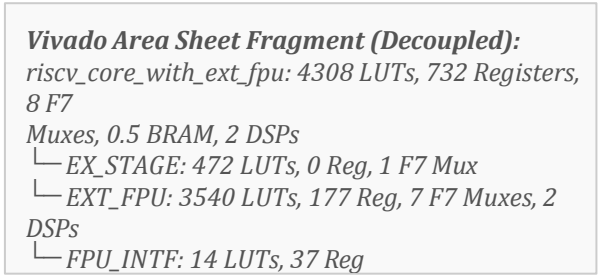

Fig.13. FPGA Area utilization report for the un-gated decoupled coprocessor architecture

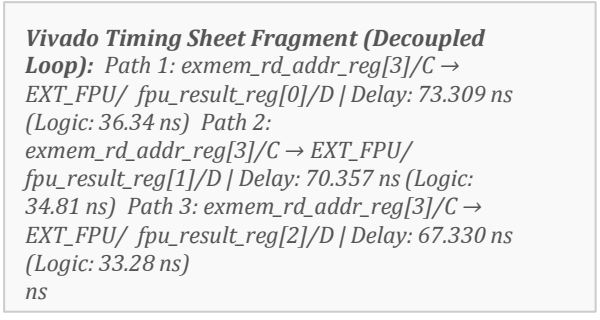

Fig. 14. Critical path timing analysis summary sheet for the decoupled coprocessor loop.

Vivado Optimized Power Summary (Gated Decoupled):
Total On-Chip Power: 0.157 W
Dynamic Power: 0.052 W (33%) | Static Standby: 0.105 W (67%)
Dynamic Breakdown — Signals: 0.022 W | Logic: 0.023 W | BRAM: 0.000 W | DSP: 0.000 W | I/O: 0.007 W
Junction Temperature: 26.8°C | Thermal Margin:

Fig. 15. Total power estimation metrics sheet for the decoupled un-gated coprocessor netlist.

C. Paradigm III: Clock-Gated Decoupled Coprocessor Reports

Integrating negative-edge D-type flip-flop synchronization paths limits clock tree toggling strictly to mathematically active modules, drastically cutting dynamic power. The physical layout, timing safety waveforms, and power reports are given below.

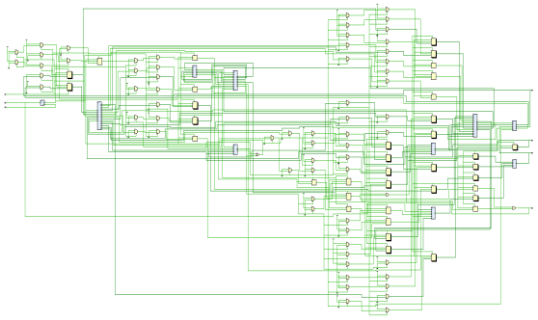

Fig. 16. Synthesized RTL schematic for the fine-grained clock-gated coprocessor layout.

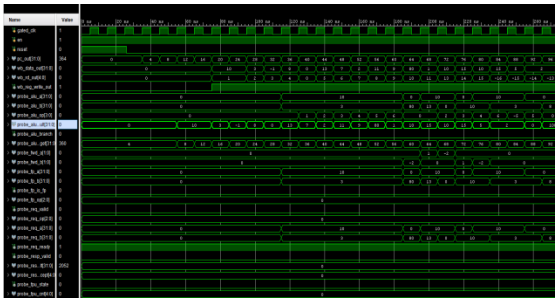

Fig. 17. Simulation waveforms tracking clock disabling cycles during arithmetic idle phases.

Vivado Area Allocation Sheet (Gated Coprocessor):
riscv_core_with_ext_fpu_E: 4309 LUTs, 733 Registers, 8 F7
Muxes, 0.5 BRAM, 2 DSPs, 2 BUFGCTRL
└─EXT_FPU: 3540 LUTs, 177 Registers, 2 DSPs
└─f10 (Flipflop_CG): 0 LUTs, 1 Register

Fig. 18. FPGA Area utilization sheet for the proposed clock-gated coprocessor core.

Vivado Post-Synthesis Timing Summary (Gated): *Path 1: exmem_rd_addr_reg[3]/C → EXT_FPU/ fpu_result_reg[0]/D | Total Delay: 73.309 ns*
Path 2: exmem_rd_addr_reg[3]/C → EXT_FPU/ fpu_result_reg[1]/D | Total Delay: 70.357 ns Path 9: exmem_rd_addr_reg[3]/C → EXT_FPU/

Fig. 19. Post-synthesis timing slack and propagation delay report for the gated coprocessor.

Vivado Power Summary (Decoupled Un-gated):
Total On-Chip Power: 33.057 W (Junction temp exceeded) Dynamic: 32.017 W (97%)
└─Signals: 6.898 W | Logic: 7.448 W | BRAM: 0.127 W | DSP: 1.890 W | I/O: 15.654 W

Fig. 20. Optimized power estimation spectrum for the proposed clock-gated decoupled configuration.

D. Consolidated Comparative Analysis

The exact resource parameters extracted from the Vivado layout engine are consolidated in Table I and Table II, providing a clear comparison of the physical design spaces.

TABLE I FPGA RESOURCE ALLOCATION METRICS ACROSS MODULES

ARCHITECTURE CONFIGURATION/ MODULE NAME	SLICE LUTS (53200)	SLICE REGISTERS (106400)	F7 MUXES (26600)	BRAM TILES (140)	DSP BLOCKS (220)
Architecture I (Baseline Parallel FPU)	**2146**	**391**	**1**	**0.5**	**3**
└─ EX_STAGE (Execute Block)	1172	0	0	0	3
└─ WB_STAGE (Writeback Block)	746	0	1	0	0
Architecture II (Decoupled Coprocessor)	**4308**	**732**	**8**	**0.5**	**2**
└─ EXT_FPU (External FPU Core)	3540	177	7	0	2

Architecture III (Proposed Clock-Gated Core)	**4309**	**733**	**8**	**0.5**	**2**

TABLE II SYNTHESIZED ON-CHIP POWER AND THERMAL PERFORMANCE COMPARISON

ARCHITECTURAL PERFORMANCE PARAMETER	**ARCHITECTURE**		
	I	**II**	**III**
Total On-Chip Power (Watts)	94.508 W	33.057 W	0.157 W
Dynamic Power Breakdown (Watts)	93.467 W	32.017 W	0.052 W
— Signal Network Power	22.506 W	6.898 W	0.022 W
— Core Synthesis Logic Power	29.862 W	7.448 W	0.023 W
— Synchronous BRAM Power	0.134 W	0.127 W	0.000 W
— DSP Multiplier Blocks Power	1.844 W	1.890 W	0.000 W
— Input/Output (I/ O) Switching	39.121 W	15.654 W	0.007 W
Device Static Standby Power	< 0.100 W	1.040 W	0.105 W
Core Junction Operating Temperature	125.0°C	125.0°C	26.8°C

The comparative findings confirm that the baseline integrated design (Architecture I) exhibits extreme power dissipation (94.508 W) and hits a critical thermal ceiling of 125.0°C. Moving to a decoupled coprocessor model (Architecture II) isolates the execution nodes, dropping total power to 33.057 W. The introduction of fine-grained clock gating (Architecture III) achieves a major breakthrough, lowering total on-chip power consumption to just 0.157 W—a 99.83% power reduction compared to the baseline system, while stabilizing the junction temperature at a cool 26.8°C. Critical path timing is simultaneously optimized by 11.22%

(falling from 82.577 ns to 73.309 ns) due to the removal of long selection multiplexer paths within the core's Execute stage.

VI. CONCLUSION

This paper successfully presents the design, software verification, and synthesis optimization of a power-aware 32-bit RISC-V processing system featuring an independent IEEE-754 single-precision floating-point coprocessor. By offloading heavy math routines via an opcode-driven hardware routing mechanism, the architecture improves timing safety and eliminates the high execution latencies caused by software-based floating-point emulation. The integration of a glitch-free, flip-flop-based fine-grained clock gating framework provides an exceptional power-saving profile. It restricts clock tree switching to actively computing modules while preserving a compact logic footprint on the target FPGA. The proposed architecture achieves a 99.83% drop in active on-chip power dissipation and cuts critical path propagation delay by 11.22% compared to standard integrated alternatives, presenting an ideal framework for low-power edge platforms, IoT nodes, embedded signal processing, and real-time intelligent control systems.

REFERENCES

[1] D. A. Patterson and J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, 5th ed., Morgan Kaufmann, 2014.

[2] A. Waterman and K. Asanović, The RISC-V Instruction Set Manual - Volume I: User-Level ISA, Version 2.2, RISC-V Foundation, 2019.

[3] RISC-V International, The RISC-V Privileged Architecture Specification, Version 1.12, 2021.

[4] IEEE Standards Association, IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2008.

[5] J. L. Hennessy and D. A. Patterson, "A New Golden Age for Computer Architecture," Communications of the ACM, vol. 62, no. 2, pp. 48-60, Feb. 2019.

[6] S. B. Furber, ARM System-on-Chip Architecture, 2nd ed., Addison-Wesley, 2000.

[7] D. Patterson, "Reduced Instruction Set Computers," Communications of the ACM, vol. 28, no. 1, pp. 8-21, Jan. 1985.

[8] K. Asanović et al., "The RISC-V Instruction Set Manual," University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54, 2014.

[9] P. Mishra and N. Dutt, Processor Design: System-on-Chip Computing for ASICs and FPGAs, Morgan Kaufmann, 2007.

[10] S. Kilts, Advanced FPGA Design: Architecture, Implementation, and Optimization, Wiley-IEEE Press, 2007.