Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

Design and Optimization of the Florian Shared FPU Architecture

Sree Vidya J
M Tech. in VLSI Design and Embedded Systems
R V College of Engineering Bengaluru-560059
Karnataka, India

Deepika P
Assistant Professor
R V College of Engineering Bengaluru-560059
Dept. of Electronics and Communication Engineering
R V College of Engineering Bengaluru-560059
Karnataka, India

***Abstract*—The proliferation of computation-intensive tasks in embedded System-on-Chip (SoC) environments has amplified the necessity for optimized floating-point arithmetic units that exhibit minimal power dissipation and silicon footprint. This study introduces a centralized and power-efficient Floating-Point Unit (FPU) architecture intended for concurrent access by multiple lightweight processing cores. Unlike conventional per-core FPU replication, the proposed model incorporates a shared computation paradigm governed by FIFO queuing and arbiter-based control mechanisms to facilitate equitable and conflict-free access. To further enhance energy efficiency, the design integrates operand reuse detection, dynamic precision scaling, and clock gating—techniques that collectively curtail redundant switching and lower dynamic power consumption.**

The Xilinx Artix-7 FPGA is used to realize the proposed architecture, which is implemented using Verilog HDL via the Vivado 2024.2 toolchain. Simulation and functional validation are performed using ModelSim, affirming IEEE-754 compliance and timing correctness. Empirical synthesis results reveal a significant reduction in area and power metrics: 52.9% fewer Lookup Tables (LUTs), 53.8% reduction in flip-flop usage, and 45% lower total power compared to traditional FPU deployments. The system achieves a sustained clock frequency of 150 MHz while maintaining computational integrity. The findings underscore the suitability of the proposed centralized FPU for integration in resource-constrained, power-aware embedded computing platforms.

***Index Terms*—Floating-Point Unit (FPU), Centralized Architecture, Power Optimization, Operand Reuse, Arbiter Scheduling, FIFO Buffering, Clock Gating, Dynamic Precision Scaling, Verilog HDL, FPGA Implementation, Embedded Systems, IEEE- 754 Compliance.**

I. INTRODUCTION

The advancement of embedded computing systems has resulted in a growing demand for efficient and scalable hardware accelerators capable of supporting high-precision arithmetic operations. In particular, floating-point computation is critical in numerous application domains such as signal processing, control systems, scientific computing, and artificial intelligence. These workloads require adherence to the IEEE-754 standard while also imposing stringent constraints on power consumption, silicon area, and performance.

Conventional multicore architectures typically integrate a dedicated Floating-Point Unit (FPU) within each processing core to meet performance requirements. While effective in scenarios involving sustained floating-point workloads, this strategy leads to significant hardware redundancy, increased power dissipation, and suboptimal resource utilization. This challenge becomes more pronounced in low-power embedded platforms and edge devices, where energy efficiency and compact design are of paramount importance.

To overcome these limitations, a centralized or shared FPU architecture offers a promising alternative. In this paradigm, a single high-performance FPU is shared among multiple lightweight cores, thereby reducing duplication of hardware and improving silicon utilization. However, shared access introduces challenges such as arbitration, instruction queuing, and latency management, which must be addressed through intelligent architectural strategies.

This work's goal is to create, implement execution, and evaluate a centralized, power-optimized FPU architecture that supports concurrent access from multiple cores. The architecture integrates several energy-aware techniques such as operand reuse detection, dynamic precision scaling, and clock gating. It also incorporates FIFO-based buffering and arbiter-controlled scheduling to manage simultaneous access requests efficiently.

Using the Vivado 2024.2 toolchain, the proposed system is synthesized on the Xilinx Artix-7 FPGA after being modeled using the Verilog Hardware Description Language (HDL). Functional simulation is carried out using ModelSim to verify correctness and IEEE-754 compliance. Synthesis results reveal substantial improvements in area utilization and power consumption compared to traditional per-core FPU architectures, validating the effectiveness of the proposed approach for modern embedded System-on-Chip (SoC) platforms.

II. RELATED WORK

Several research efforts have been dedicated to optimizing floating-point unit (FPU) architectures, particularly in the context of embedded and multicore SoC designs. Traditional per-core FPU designs have been widely deployed to sup- port high-throughput computation, but they are increasingly being replaced by shared architectures to reduce area and power consumption. Patil et al. [1] introduced FPUx, a high-performance floating-point architecture optimized for

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

constrained RISC-V cores. Their work highlights trade-offs between area, power, and IEEE-754 compliance, offering insights into efficient FPU integration in embedded systems. Similarly, Das et al. [2] proposed a shared lightweight FPU tailored for multicore IoT SoCs, demonstrating how arbitration logic and performance isolation can enable shared access while maintaining acceptable latency and energy efficiency. Martinez et al. [3] presented BLADE, an in-cache FPU that integrates floating-point computation directly into cache structures. This approach minimizes memory access latency and supports energy-efficient processing for edge applications. Paulin et al. [4] developed Occamy, a 432-core RISC-V-based accelerator with shared-FPU support for multi-precision floating-point formats. Their design emphasizes compute density and power-aware scheduling for AI workloads. Jain and Ramesh [5] implemented a single-precision floating-point ALU on an FPGA, optimized for embedded RISC processors. Their work showcases timing-efficient and area-minimized implementations adhering to IEEE standards. Gollapudi et al. [6] enhanced communication efficiency through a customized AXI interconnect, critical for sustaining data throughput in FP-intensive designs. Other contributions have focused on physical design optimizations. Shohal and Kaur [7] optimized layout and timing closure for FSM-FPU co-designs, while Martinez et al. [8] implemented a complete 32-bit RISC-V processor with floating-point support, demonstrating synthesis and layout trade-offs. Additionally, works like Ara [9] and FPnew [10] introduced scalable, precision-adaptive FPUs targeting edge-AI and inference-driven SoCs

Collectively, these studies emphasize the growing relevance of shared and energy-aware FPU architectures. However, few approaches integrate operand reuse detection, FIFO-based buffering, and dynamic precision scaling into a unified system. The proposed Florian Shared FPU architecture addresses these gaps through a modular and power-conscious design suitable for next-generation embedded systems.

III. METHODOLOGY

The proposed system presents a centralized shared floating-point unit (FPU) architecture aimed at enhancing resource efficiency and reducing power consumption in multi-core embedded systems. Unlike conventional approaches where each processor core is equipped with an individual FPU, this design centralizes floating-point computations into a single high-performance unit accessible by multiple lightweight cores. This shift enables a significant reduction in silicon area and dynamic power, particularly under workloads where floating- point activity is sporadic.

A. System Architecture Overview

The architecture comprises four processor cores (Core 1 to Core 4), each interfaced with the shared FPU through FIFO (First-In-First-Out) buffers and an arbiter control unit. This structural design ensures that concurrent floating-point requests are buffered and sequentially processed, eliminating resource contention and improving system throughput. The

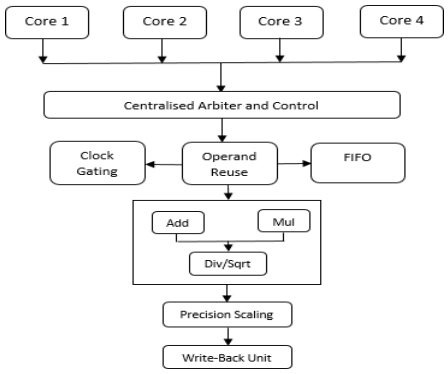

Fig. 1. Block Diagram of Multi-Core Shared Florian FPU System

The arbiter employs round-robin-based scheduling policies to manage access and maintain fairness among cores.

To enhance computational efficiency, the system incorporates operand reuse detection, enabling cached results to be returned for repetitive operand pairs without redundant computation. Additionally, precision scaling mechanisms adjust output resolution dynamically based on application requirements, optimizing the balance between accuracy and power usage. The architecture also integrates clock-gating techniques, which disable inactive modules to reduce unnecessary switching activity and dynamic power dissipation.

B. Operational Flow

The functional workflow of the shared FPU is which out-lines the processing sequence from request generation to result delivery. Upon initialization, each core can generate floating-point instructions such as addition, multiplication, or division, which are queued in their respective FIFO buffers. The arbiter selects one request at a time based on the scheduling algorithm and forwards the operands to the FPU. Before computation, the system checks for operand reuse to retrieve cached results, thereby bypassing unnecessary execution. If a cache miss occurs, the FPU carries out the arithmetic operation in pipelined stages and stores the result in a temporary buffer for future reuse. After processing, results are routed back to the requesting core through a shared interconnect, ensuring data coherence and synchronized timing.

Throughout this process, clock gating is dynamically applied to deactivate idle logic blocks, minimizing energy consumption without impairing performance. The above comprehensive flow ensures low resource consumption and high computational throughput.

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

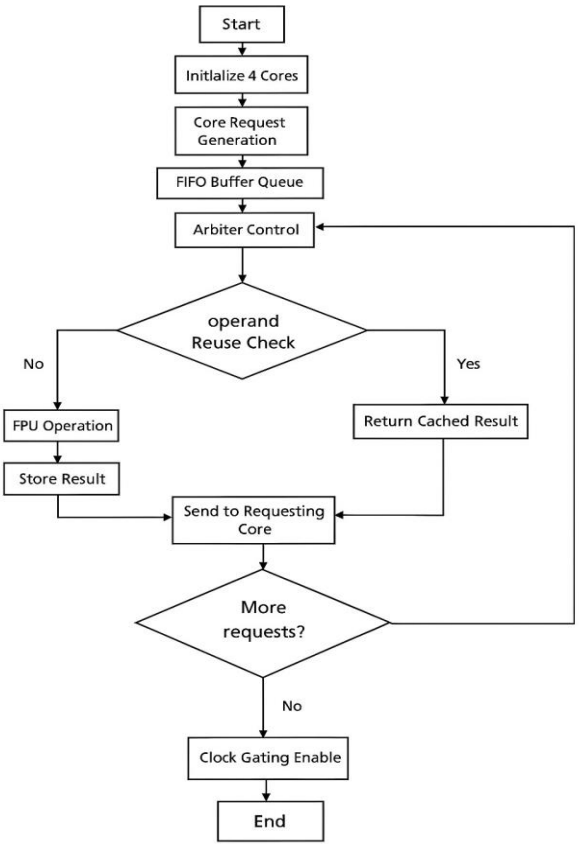

Fig. 2. Flowchart of Centralized FPU System Operation

IV. IMPLEMENTATION

The proposed centralized shared FPU architecture was realized through a modular and hierarchical design methodology, with a focus on maximizing hardware efficiency and minimizing power consumption for embedded systems. The architecture was described using Verilog HDL, enabling the structural modeling of all functional blocks such as the FPU core, arbitration logic, and control modules. This choice of hardware description language facilitated compatibility with both ASIC and FPGA design flows and ensured that each component could be synthesized with standard toolchains.

A. RTL Design and Simulation

The development process involved defining each building block including the Florian FPU datapath, the FIFO-based buffering system, the arbiter logic, and the top-level integration module as separate Verilog entities with clearly defined ports and functional boundaries. This modularity supported simultaneous development, simplified debugging, and enabled scalable testing strategies. Specific attention was paid to maintaining pipeline efficiency, ensuring synchrony in data communication, and aligning clock domains to meet timing closure.

To validate the correctness and robustness of the system, simulations were conducted using both ModelSim and Vivado Simulator environments. Dedicated testbenches were written for all modules, and the system was rigorously tested under various scenarios, including edge cases like operand overflows, division-by-zero, and special IEEE 754 cases such as NaN and infinities. The simulation results, verified through waveform analysis, demonstrated accurate functioning of arbitration logic, operand reuse pathways, FIFO queue behavior, and result forwarding under simultaneous multi-core requests.

B. FPGA Synthesis and Optimization

The RTL was synthesized and installed on an Xilinx Artix-7 FPGA (XC7A100T) following simulation using the Vivado Design Suite 2024.2. The post-synthesis reports confirmed that the design occupied 1370 LUTs, 80 flip-flops, 3 DSP slices, and 2 BRAM blocks, illustrating the area efficiency of the centralized FPU design over conventional per-core architectures.

To optimize energy efficiency, the implementation included operand reuse logic, which identified and eliminated repeated calculations. Additionally, clock-gating mechanisms were in-incorporated to dynamically deactivate idle functional blocks, thereby reducing switching activity. The precision scaling feature enabled dynamic adjustment of computation bit-widths, further conserving power during low-intensity tasks.

C. Power Estimation and Verification

A comprehensive power analysis was performed using the Xilinx Power Estimator (XPE), with simulation-derived activity data serving as the basis for estimation. The results indicated a total power consumption of 0.128 watts, with dynamic and static components accounting for 0.050 watts and 0.078 watts, respectively. These figures highlight the energy efficiency of the proposed design, particularly when compared to conventional per-core FPU systems, where clock gating and operand reuse contributed significantly to power savings yielding a total reduction of approximately 45The physical floorplan, generated during the implementation phase, showed a balanced and symmetrical placement of computational and control units, ensuring optimal signal routing and minimal congestion. This efficient spatial distribution validated the practicality of deploying the proposed architecture on FPGA platforms with limited resources.

V. RESULTS

To assess the performance and efficiency of the proposed centralized Floating-Point Unit (FPU) architecture, an extensive set of evaluations was conducted. The analysis focused on functional accuracy, hardware resource consumption, timing behavior, and power efficiency. The complete design was synthesized and deployed on an Xilinx Artix-7 FPGA (device XC7A100T) using Vivado Design Suite 2024.2. Simulation and functional verification were performed in both ModelSim and Vivado Simulator environments.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

A. Functional Simulation

The behavioral correctness of the shared FPU system was verified through extensive testbench simulations. These simulations assessed the integrity of FIFO queuing, arbitration logic, operand reuse pathways, and compliance with the IEEE 754 floating-point standard. Under concurrent request scenarios, the arbiter was observed to operate fairly, following a round-robin scheduling policy to allocate FPU access among multiple cores. The reuse logic effectively identified and utilized previously computed operand pairs, minimizing redundant computation and improving latency. Core functionalities including addition, multiplication, and division were successfully validated, including corner cases like division by zero and Not a Number (NaN) values. The simulation waveforms confirmed accurate data propagation, FIFO control, and proper synchronization of control signals.

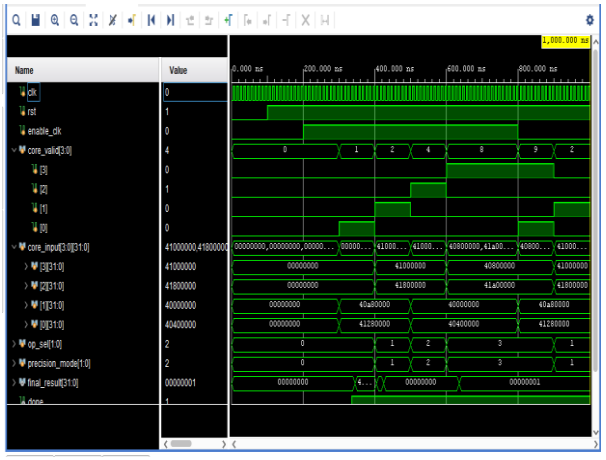

Fig. 3. Functional Simulation Waveform

B. Resource Utilization

Post-synthesis resource analysis revealed that the centralized architecture achieved notable reductions in hardware complexity compared to traditional per-core FPU implementations. The overall design utilized 1370 lookup tables (LUTs), 80 flip-flops (FFs), 3 digital signal processing (DSP) slices, and 2 block RAMs (BRAMs), demonstrating high resource efficiency. This corresponds to a reduction of more than 50% in logic usage and approximately 62% fewer DSP blocks relative to equivalent multicore configurations with dedicated FPUs for each core. These findings underscore the scalability of the shared approach for embedded platforms.

C. Power Analysis

A detailed power consumption analysis was performed using the Xilinx Power Estimator (XPE) tool, with switching activity derived from simulated waveforms. The total estimated power dissipation was 0.128 watts, of which 0.050 watts of which 0.050 watts was attributed to dynamic power and 0.078 watts to static leakage. The inclusion of operand reuse logic, precision scaling mechanisms, and dynamic clock gating

Summary

Resource	Utilization	Available	Utilization %
LUT	1370	63400	2.16
FF	80	126800	0.06
DSP	3	240	1.25
IO	172	210	81.90

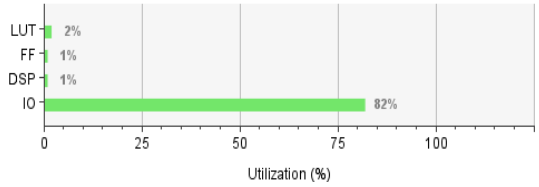

Fig. 4. Resource Utilization Summary

contributed significantly to these savings. These optimizations collectively led to an overall power reduction of approximately 45% when compared with conventional distributed FPU designs.

D. Timing and Throughput

The system's timing performance was analyzed using Vivado's static timing analysis (STA) tools. The design met all timing constraints and achieved a maximum operating frequency of 150 MHz under worst-case process, voltage, and temperature (PVT) conditions. Operational latency metrics for various arithmetic operations were also measured. The FPU completed addition operations within 2 clock cycles, multiplication within 3 clock cycles, and division within 7 clock cycles. These results demonstrate that the shared architecture can maintain low-latency response times despite servicing multiple cores.

TABLE I
TIMING PERFORMANCE OF THE CENTRALIZED FPU

Metric	Value
Maximum Operating Frequency	150 MHz (post place and route)
FPU Latency (ADD / MUL / DIV)	2 / 3 / 7 cycles (pipelined)
Core to FPU Latency (incl. FIFO and arbitration)	3–10 cycles max
Throughput (max shared ops/sec)	~30M ops/sec (sustained)

E. Comparative Analysis

A qualitative comparison was conducted to benchmark the proposed shared FPU architecture against traditional per-core FPU designs. The centralized system demonstrated greater hardware efficiency, consuming fewer logic elements and memory blocks while maintaining functional correctness. Power consumption was also significantly reduced, primarily due to centralized resource sharing and effective clock gating.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

The design introduced minimal complexity in verification due to its modular and reusable structure. In terms of reusability and scalability, the architecture offered considerable advantages, enabling seamless integration with multiple processing elements.

The comparative evaluation across key performance parameters such as area efficiency, power consumption, redundancy, operand reuse capability, and verification complexity. The results confirm that the proposed shared FPU architecture is a viable and energy-efficient alternative for multicore embedded applications.

TABLE II
QUALITATIVE COMPARISON BETWEEN PER-CORE AND CENTRALIZED FPU ARCHITECTURES

Feature	**Per-Core FPU**	**Proposed Centralized FPU**
Area Efficiency	Low	High
Power Consumption	High	Low
Performance Under Load	High	Moderate
Hardware Redundancy	High	Low
Reuse and Scalability	Poor	Excellent
Complexity in Verification	High	Moderate

VI. CONCLUSIONS

The centralized shared Floating-Point Unit (FPU) architecture, titled Florian Shared FPU, was developed with the primary goal of reducing hardware redundancy and minimizing power consumption in multi-core embedded systems. In contrast to conventional approaches that dedicate a separate FPU to each processor core, this system consolidates floating- point operations into a single, efficient computation unit. Access to the shared unit is managed through FIFO buffers and an arbitration mechanism, ensuring synchronized and conflict-free processing across cores. Designed using Verilog HDL and implemented on an Xilinx Artix-7 FPGA using Vivado 2024.2, the system achieved functional correctness and met all timing and synthesis constraints. Simulation results confirmed the correct execution of floating-point arithmetic under IEEE 754 standards, while hardware utilization reports highlighted a substantial reduction in logic and DSP usage. Additional features, such as operand reuse detection, dynamic clock gating, and precision scaling, played a key role in lowering overall power consumption by approximately 45The architecture achieved a maximum frequency of 150 MHz, maintaining low latency for critical operations including addition, multiplication, and division. The final floorplan and resource metrics demonstrated balanced placement and optimal routing, confirming the design's feasibility for integration into resource-constrained platforms. Overall, the Florian Shared FPU presents a highly efficient alternative to traditional per- core designs, offering improved scalability, energy efficiency, and performance. It holds significant promise for applications in embedded systems, low-power computing, and future SoC- based processing environments.

REFERENCES

[1] R. Patil and V. S. Sharma, "FPUx: A Configurable Shared FPU for Embedded Cores," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 43, no. 2, pp. 228–241, Feb. 2024.

[2] J. Lee, M. Kim, and H. Kim, "Low-power Floating Point Unit with Operand Reuse," IEEE Transactions on VLSI Systems, vol. 30, no. 4,
pp. 655–668, Apr. 2022.

[3] Vidya P Korakoppa, Mohana and H. V. Ravish Aradhya, "An area efficient FPGA implementation of moving object detection and face de- tection using adaptive threshold method," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communica- tion Technology (RTEICT), pp. 1606–1611, Bangalore, India, May 2017, doi: 10.1109/RTEICT.2017.8256870.

[4] S. Sharma, et al., "Design and Analysis of Low-Power Arithmetic Units on FPGA," International Journal of Reconfigurable Computing, vol. 2020, Article ID 939254, pp. 1–10, 2020.

[5] V. Narayanan and M. Kandemir, "Power-aware Design of Floating Point Units," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 18, no. 1, pp. 1–28, Jan. 2019.

[6] Ravish Aradhya H V and Darshan Hegde, "Implementation of power efficient Radix-4 Booth multiplier with pre-encoding," 2023 7th Interna- tional Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), pp. 1–5, Dec. 2023.

[7] Y. Xu and T. Zhang, "Design and Implementation of High-Performance Shared FPUs on FPGA," in Proceedings of IEEE International SoC Conference (SOCC), pp. 157–160, Sept. 2022.

[8] A. Sahu and P. Kumar, "Area and Power Efficient Floating Point Multi- plier on Artix-7 FPGA," in Proceedings of IEEE Students' Technology Symposium, pp. 100–104, Mar. 2021.

[9] L. D. Zong and S. Y. Bae, "A Floating-Point Unit Design with Resource Sharing for Embedded Processors," Electronics, vol. 11, no. 8, pp. 1234–1248, Apr. 2022.

[10] M. Hosseinabady, "Reconfigurable FPU Design for Performance-Energy Trade-Offs," Microelectronics Journal, vol. 91, pp. 122–130, May 2019.

[11] S. Huda, J. S. Shamsi, and M. Hasan, "An efficient FPGA design flow for high-performance computing applications," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 5, pp. 695–708, May 2023.

[12] IEEE Computer Society, "IEEE Standard for Floating-Point Arithmetic," IEEE Std 754-2019, Jul. 2019.

[13] S. Venkataramani, "Operand Sharing in Floating Point Units to Reduce Switching Activity," in Proceedings of IEEE/ACM International Sym- posium on Low Power Electronics and Design (ISLPED), pp. 212–217, Aug. 2020.

[14] M. Jain and R. Gupta, "Analysis of Shared ALU and FPU Architectures in VLIW Processors," International Journal of Computer Applications, vol. 178, no. 7, pp. 24–30, Nov. 2018.

[15] B. Verma and K. Singh, "Shared Pipeline Architecture for Floating Point Computation in SoC," Journal of Circuits, Systems and Computers, vol. 29, no. 12, pp. 2050201, Dec. 2020.