Irish Interdisciplinary Journal of Science & Research (IIJSR)
Volume 9, Issue 1, Pages 73-79, January-March 2025

# A power-efficient and high-performance single precision floating point unit for AI applications, designed specifically for cost-effective microcontrollers

Allwin Sanjay D.[1], Akash R.V.[2], Jerin J.[3] & Gopal Ram K.[4*]

[1,2,3,4]*Department of Electronics and Communication Engineering, Stella Mary's College of Engineering, Kanyakumari, Tamil Nadu, India. Corresponding Author (Gopal Ram K.) Email: kgopalram78@gmail.com**

*DOI: https://doi.org/10.46759/IIJSR.2025.9108*



*Article Received: 09 January 2025* *Article Accepted: 21 March 2025* *Article Published: 27 March 2025*

ABSTRACT

The growing interest in AI applications within embedded systems has increased the shortcomings of affordable microcontrollers, which frequently do not include hardware floating-point units (FPUs). Historically, these devices have depended on software emulation for floating-point operations, leading to considerable performance limitations and diminished accuracy. This paper tackles this issue by introducing a power-efficient, high-performance single-precision FPU for resource-limited microcontrollers. Using Verilog, we developed a custom hardware FPU where the sub-modules used combinational 3-stage pipeline structure and the top module used 4-stage pipeline structure with sequential operation. A comparison with a nonpipelined version showed performance improvements. The throughput was boosted with a 56.36% gain in maximum clock speed, gain in throughput equals to 369.34%. The other benefits are a 13.19% reduction in the power consumption of the pipelined design. Dedicated hardware uses shared memory which saves more chip area for co-integration of high-performance computational elements such as hardware implementations of the activation functions used in different AI applications.

**Keywords:** Floating-Point Operations; Single-Precision; AI; Microcontrollers; Power-Efficient; Hardware Design; Custom Hardware FPU.

## 1. Introduction

The growing number of AI applications ranging from edge computing to IoT devices has increased the overall demand for efficient computational capabilities in embedded systems [1]. High-performance microcontrollers (MCUs) equipped with integrated floating-point units (FPUs) can provide the processing power required for AI workloads [2]; however, their associated cost is often a prohibitive barrier to deployment in resource-constrained environments [3]. This is especially important in cases where the sensitivity cost is exaggerated, like consumer electronics, industrial automation, wearable devices [4].

The problem comes from the core computational complexity of AI algorithms, which often depend on floating-point arithmetic in scenarios such as neural network inference and even signal processing [5],[6]. Floating-point emulation via software is flexible but incurs too much performance overhead, causing latency and energy waste [7]. This inefficiency becomes even more critical in real-time applications where responsiveness and power efficiency are important [8]. Hence, Hardware-based FPU designs are of great requirement to achieve high performance within the stringent requirements of cost and power limits inside embedded environment [9].

This study fills this critical gap by introducing a new area-efficient FPU architecture designed for cost-sensitive MCUs. For our FPU, we have created a three-stage pipelined architecture specifically for its sub-modules in order to have better optimization for internal operations while retaining low latency. In addition, a four-step pipeline is considered at the top module to ensure the most throughput and performance. In addition, we used our expertise in understanding the computational requirements of AI applications in developing custom hardware appliances [10] to optimize activation function evaluation in AI workloads. This integration notably boosts the floating-point unit's appropriateness AI tasks, doing efficient operations in environments with limited resources.

The main aim of this research is to illustrate both the feasibility and efficiency of incorporating a high-performance, cost-effective FPU into resource-constrained microcontroller units (MCUs), which in turn will allow for the deployment of AI functionalities across a wider array of applications. This study makes a good contribution to the world of VLSI design by offering a practical and efficient hardware solution that strikes a balance among performance, area, and power consumption. The proposed architecture undergoes thorough evaluation [11] through both simulation and synthesis processes, showing its ability to markedly improve the computational capabilities of embedded systems.

## 2. Preliminary

The world of scientific and engineering computation has experienced an influence from the IEEE 754 standard for floating-point arithmetic [12]. Since it was first introduced in 1985, IEEE 754 has established a dependable and consistent framework for the representation and manipulation of real numbers within digital systems, which in turn provides interoperability among various hardware and software platforms. This standardization has played an important role in making accurate and reproducible numerical computations, an essential requirement for a wide array of applications, including scientific simulations and financial modeling, among others. Nevertheless, the requirements of modern computing continued to evolve, especially with the emergence of high-performance computing and embedded systems, leading to changes to improve upon weaknesses while also introducing emerging concepts. Since standardizing our representation of floating-point numbers for computers on IEEE754-1985, we have always had two basic numbers: single precision (32-bit) and double precision (64 bit), but the details of the functions, including rounding modes, exception handling [13], and special values (e.g., infinity and NaN) were all left to specification. It was revolutionary for its time, but the standard did not support newer formats and capabilities of computers as technology advanced. Because of the rise of applications needing better precision, larger numeric ranges, and better performance on certain computational tasks, the 1985 standard was not sufficient.

To address these demands, improvements and extensions were introduced in the IEEE 754-2008 revision. The standard included new binary and decimal floating-point formats, such as half-precision (binary16) and quadruple-precision (binary128), which can serve a wider range of application needs. Revised according to the findings of the round-ups of 2008 also solved the problems of repetition, and reliability that is in the sections of the guidelines for exception handling and replication. In addition, it defined fused multiply-add operations [14] that provide considerable performance and accuracy enhancement.

## 3. Proposed Architecture

### 3.1. FPU Sub-Modules

The proposed FPU architecture consists of five arithmetic sub-modules; adder, subtractor, multiplier, divider, and comparator. These modules adopt a three-stage pipelined design of stages input fetch, Operation calculation, and result rounding with an eye on hardware efficiency while maximizing performance. This pipelining strategy is also key to increasing throughput and lowering latency, both important factors for AI workloads. And, because the comparison operation is very simple, a non-pipelined implementation is chosen for the comparator module. This

method helps to reduce hardware overhead and latency, since the comparison can be efficiently done in a single clock cycle. A combinational approach reduces control logic and register overhead which is key for cost-sensitive MCUs. This pipelined design considered for arithmetic sub-modules achieves a high-throughput floating-point operations, which accelerates AI computations effectively. This architecture provides a reasonable methodology for incorporating functional, efficient FPU capabilities into hardware resource-constrained embedded solutions by carefully adjusting performance versus hardware complexity.

### 3.2. Top-Level Module

Here, the top module for the proposed FPU architecture used a four-stage sequential logic pipeline, managing the access of data and control signals to the arithmetic sub-modules and AI acceleration unit. The pipelined design is efficient in order to maximize the throughput of the CPU and run floating point arithmetic as efficiently as possible. The 4 stages are, Instruction Decode and operand fetch, Operation execution, Result rounding and exception handling and result write-back and status update.

The architecture of proposed Floating-point unit is given in figure 1.

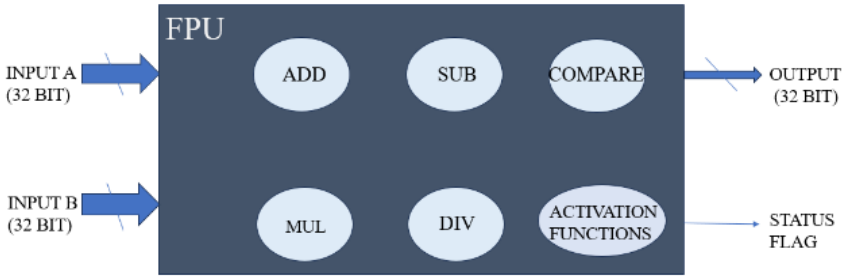


**Figure 1.** Proposed Architecture of Floating-Point Unit

### 3.3. Custom AI Acceleration Module

To improve efficiency for AI workloads, we implement the FPU with specialized hardware modules to provide acceleration for frequently used activation functions. The modules are cost-constrained for resource-optimized MCUs. This design includes hardware implementations of sigmoid, tanh, and ReLU activation functions. Hardware complexity is reduced by using piecewise linear approximation.

### 3.4. Data Formats and Exception Handling

The proposed Floating-Point Unit (FPU) adheres to the IEEE 754 standard for single-precision floating-point data representation. This particular format has been chosen to strike a harmonious balance between numerical precision and the easiness of hardware design, making it well-suited for a diverse array of artificial intelligence applications. The FPU is equipped with exception handling that proficiently manages scenarios involving Not-a-Number (NaN), infinity, zero, and subnormal numbers.

To make effective software management of these exceptions, status flags are employed to signal their occurrence. Also, the comparator module appropriately sets status flags to say the results of comparison operations. Control over the FPU's operations is governed by a 3-bit operation selection code, which dictates whether an arithmetic computation or an AI acceleration task will be executed.

## 4. Implementation

The proposed FPU architecture was designed using the Verilog Hardware Description Language (HDL). For tasks such as simulation, synthesis, and implementation—encompassing floor planning and routing—the Xilinx Vivado tool was used. The evaluation and prototyping were conducted on the Xilinx Artix-7 FPGA platform, serving as the target hardware for this endeavour.

### i) Synthesis and Implementation

The Verilog HDL code undergoes synthesis via the synthesis tools provided by Xilinx Vivado. In this process, two constraint files were implemented: one to delineate the clock and reset signals, while the other served to outline the port connections essential for interfacing with the FPGA board. Throughout the synthesis procedure, efforts were made to enhance performance while concurrently minimizing area, all in light of the resource limitations of the targeted FPGA device. The floor planning and routing stages were performed using Vivado's implementation tools, ensuring efficient utilization of the FPGA's resources and meeting timing constraints.

### ii) Verification and Simulation

A thorough verification process was carried out using testbenches designed in Verilog HDL. Separate testbenches were constructed for each sub-module, providing an exhaustive assessment of their individual functionalities. Also, a main testbench was created to evaluate the behavior of the top-level module and to confirm the integration of the Floating-Point Unit (FPU) and the AI acceleration units. The testbenches were designed to cover a wide range of input values and testcases, ensuring better verification of the design's functionality. Simulations were performed using Xilinx Vivado's simulator.

### iii) FPGA Prototyping

The design was successfully implemented on the Xilinx Artix-7 FPGA to check its functionality and performance in a real hardware setting. This FPGA platform gives us a solid testing ground to assess how well the FPU performs and how it uses resources. The successful run on the FPGA shows that we can indeed integrate this proposed FPU architecture into budget-friendly embedded systems.

## 5. Results

To see how well the new pipelined FPU architecture performs, we compared its efficiency, power usage, and area (PPA) with a traditional non-pipelined FPU design that doesn’t include the special AI activation function block.

**Table 1.** Performance Chart

| System | Proposed FPU | Existing FPU |
|---|---|---|
| Clock Speed | 86 MHz | 55 MHz |
| WNS | 0.162 | 0.191 |
| Throughput | 86 Mega-OPS | 18.33 Mega-OPS |

**Table 2.** Power Consumption Chart

| System | Proposed FPU | Existing FPU |
|---|---|---|
| Static Power | 0.138 W | 0.138 W |
| Dynamic Power | 0.020 W | 0.044 W |
| Total Power | 0.158 W | 0.182 W |

**Table 3.** Resource Utilization Chart

| System | Proposed FPU | Existing FPU |
|---|---|---|
| LUT | 1434 | 1266 |
| FF | 316 | 41 |
| BRAM | 0 | 0 |
| DSP | 2 | 2 |

**1) Performance Analysis**

We took a close look at how the FPU architecture performed by measuring its clock frequency and throughput on the target FPGA. Check out Table-1 for a summary of the results. We found that the pipelined FPU hit a maximum clock frequency of 86 MHz, while the non-pipelined version reached up to 55 MHz. This represents a 56.36% increase in clock frequency for the pipelined architecture. The latency, measured in mega-operations per second (mega-ops), was 86 mega-ops for the pipelined FPU and 18.33 mega-ops for the non-pipelined FPU. This demonstrates a 368.14% improvement in latency for the pipelined design, showing its higher throughput.

**2) Power Consumption**

Power consumption analysis was performed using Xilinx Vivado's power analysis tools. Table-2 presents the estimated power consumption of the FPU architecture on the target FPGA. The results show a total power consumption of 0.158W while the non-pipelined FPU consumed a total power consumption of 0.182 W. This indicates a 13.19% reduction in power consumption for the pipelined architecture, showing its power efficiency.

**3) Resource Consumption**

The hardware resource utilization of the proposed FPU architecture was evaluated using Xilinx Vivado's synthesis reports. Table-3 summarizes the resource consumption on the target Xilinx Artix-7 FPGA (xc7a200tfbv676-2). The results show the number of Look-Up Tables (LUTs), flip-flops (FFs), Digital Signal Processors (DSPs), and Block RAMs (BRAMs) used by the design. The LUT count indicates the amount of combinational logic required to implement the FPU and AI acceleration modules. The flip-flop count represents the number of sequential elements used for registers and pipeline stages. The DSP blocks are used to accelerate multiplication and other arithmetic operations, and the BRAMs are used for lookup tables and data storage. The pipelined FPU required 13.27% more LUTs compared to the non-pipelined version. However, the pipelined FPU used 670.73% more FFs, which is

expected due to the pipelined register stages. Both designs used 2 DSPs, indicating no change in DSP resource usage.

The results clearly demonstrate the performance and power efficiency advantages of the proposed pipelined FPU architecture. The improvement in clock frequency and latency indicates the effectiveness of the pipelined design in enhancing throughput. The reduction in power consumption further validates the design's suitability for power-constrained applications.

While the pipelined FPU uses slightly more LUTs and slightly more FFs compared to the non-pipelined version, the performance gains and power savings justify the increased resource utilization. The increased FF usage is expected due to the pipelined design, and the LUT increase is a reasonable trade-off for the increase in performance. The AI activation modules also contribute to the increase of LUTs.

The comparison with the non-pipelined FPU shows the benefits of the proposed architecture, demonstrating its superiority in terms of performance and power efficiency. These results confirm the feasibility of integrating a high-performance, low-power FPU into cost-constrained MCUs, enabling the deployment of AI capabilities in a wider range of embedded applications.

## 6. Conclusion

This paper presented a pipelined FPU architecture optimized for cost-constrained MCUs, specifically targeting the integration of AI capabilities. Implemented and evaluated on an FPGA, the design demonstrated improvements in performance, achieving a 56% faster clock frequency and a 368% reduction in latency, alongside a 13% decrease in power consumption, when compared to a non-pipelined implementation. The inclusion of custom AI acceleration modules, using piecewise linear approximations, further enhances the FPU's efficiency for AI workloads, making it a viable solution for resource-limited embedded systems. Future research will explore advanced optimizations for the AI modules and investigate broader application domains.

**Declarations**

**Source of Funding**

This study did not receive any grant from funding agencies in the public, commercial, or not–for–profit sectors.

**Competing Interests Statement**

The authors declare no competing financial, professional, or personal interests.

**Consent for publication**

The authors declare that they consented to the publication of this study.

**Authors' contributions**

All the authors made an equal contribution in the Conception and design of the work, Data collection, Drafting the article, and Critical revision of the article. All the authors have read and approved the final copy of the manuscript.

**Availability of data and material**

Not applicable for this study.

**References**

[1] Ankita Tiwari, Gaurav Trivedi & Prithwijit Guha (2021). Design of a Low Power Bfloat16 Pipelined MAC Unit for Deep Neural Network Applications. In IEEE Region 10 Symposium (TENSYMP), Pages 1–5.

[2] Nhut-Minh Ho & Weng-Fai Wong (2017). Exploiting half precision arithmetic in Nvidia GPUs. In 2017 IEEE High Performance Extreme Computing Conference (HPEC).

[3] Zaruba, F., et al. (2020). Snitch: A tiny pseudo dual-issue processor for area and energy efficient execution of floatingpoint intensive workloads. IEEE Transactions on Computers, Pages 1–1.

[4] Mach, S., et al. (2020). Fpnew: An open-source multiformat floating-point unit architecture for energy-proportional transprecision computing. IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

[5] Hockert, N., & Compton, K. (2009). Ffpu: Fractured floating point unit for fpga soft processors. In 2009 International Conference on Field Programmable Technology, Pages 143–150.

[6] Muktai N. Joshi & Dhanashri H. Gawali (2016). Floating point unit core for Signal Processing applications. In Online International Conference on Green Engineering and Technologies (IC-GET), Pages 1–5.

[7] Pimentel, J.J., et al. (2016). Hybrid hardware/software floating-point implementations for optimized area and throughput tradeoffs. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25(1): 100–113.

[8] Park, J., et al. (2023). Florian: Developing a low-power RISC-V multicore processor with a shared lightweight FPU. In Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED), Vienna, Austria, Pages 1–6.

[9] Bertaccini, L., et al. (2021). Tiny-FPU: Low-cost floating-point support for small RISC-V MCU cores. In Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Pages 1–5.

[10] Sundaresan, S., et al. (2021). Artificial intelligence and machine learning approaches for smart transportation in smart cities using blockchain architecture. In Blockchain for Smart Cities, Elsevier, Pages 35–56.

[11] Rupprecht, B., et al. (2022). Performance evaluation of AI algorithms on heterogeneous edge devices for manufacturing. In Proc. IEEE 18th Int. Conf. Autom. Sci. Eng. (CASE), Mexico City, Mexico, Pages 2132–2139.

[12] IEEE (1987). IEEE Standard 754-1985 for Binary Floating-Point Arithmetic. IEEE.

[13] Demmel, J.W., & Li, X. (1994). Faster numerical algorithms via exception handling. IEEE Transactions on Computers, 43(8): 983–992.

[14] Pajany, M., et al. (2023). Enhancing Irrigation Efficiency with AI-Based Instinctive Irrigation System (IIS) in Wireless Sensor Networks. In 2023 International Conference on System, Computation, Automation and Networking (ICSCAN), IEEE, Pages 1–7.