ISSN 2277-2685
IJESR/July-September. 2026/ Vol-16/Issue-3/58-62
Mr. Mohd Abdul Aziz *et. al.*, / International Journal of Engineering & Science Research
Full Length Article

Design And FPGA Simulation Of 4-Term Floating Point Adder With Single Normalization

Mr. Mohd Abdul Aziz[1], Dr. Amairullah Khan Lodhi[2], Mr. H. A. Abdus Samad[3]
[1]PG Student; Dept. of ECE (VLSI System Design), Shadan College of Engineering and Technology, Hyderabad, India.
[2] Professor, R & D Coordinator, Dept. of ECE, Shadan College of Engineering and Technology, Hyderabad, India.
[3]Assistant Professor, Dept. of ECE, Shadan College of Engineering and Technology, Hyderabad, India.
Mail Id; mohammedabdul0205@gmail.com[1], lak_resumes@yahoo.com[2], abdus.samad569@gmail.com[3]
Accepted 23-06-2026

Abstract
Floating-point addition is a fundamental operation in digital signal processing, scientific computing, artificial intelligence, and FPGA-based computing systems. Conventional IEEE 754 floating-point adders provide high numerical accuracy but suffer from increased latency and hardware complexity due to exponent alignment, carry propagation, normalization, and rounding operations. This work presents the design and FPGA implementation of a hardware-efficient 4-term floating-point adder with single normalization and pipelined parallel accumulation. The proposed architecture employs a carry-preserving parallel adder tree to perform simultaneous accumulation of four aligned mantissas, reducing arithmetic delay and improving throughput. A lightweight single-stage normalization technique minimizes hardware overhead while preserving near-monotonic arithmetic behavior. Pipeline registers further enhance performance by reducing the critical path. The architecture is implemented in Verilog HDL, verified using ModelSim, and synthesized on an Intel Cyclone V FPGA using Quartus Prime. Synthesis results show that the design utilizes only 334 Adaptive Logic Modules (ALMs) and 25 registers, occupying less than 1% of the available FPGA resources. The proposed architecture achieves an effective balance between hardware efficiency, computational speed, and numerical accuracy, making it suitable for FPGA-based arithmetic accelerators and high-performance embedded computing applications.

Keywords—*Floating-Point Adder, IEEE 754, FPGA, Verilog HDL, Approximate Computing, Pipelining, Parallel Accumulation, Single Normalization*

I. INTRODUCTION

Floating point arithmetic is a crucial component of today's digital systems with a high dynamic range numerical processing requirements. The floating point addition and accumulation operations are being extensively used in applications like machine learning accelerators, image processing systems, scientific simulations, graphics processing, digital communications, etc. With the rise in computational complexity in present-day high-speed systems, the need for high-speed arithmetic architecture which is also hardware efficient has gained significance.

The conventional IEEE754 floating point adders are designed to perform accurate numerical computation by aligning the exponents, adding the mantissa, normalizing the result and rounding the result. These architectures attain high numerical precision, but can be complex and have a long propagation delay, because of the sequential carry propagation and iterative normalization features. These limitations are more important in multi-operand floating point accumulation systems, where multiple carry propagation operations and normalization stages result in significant arithmetic delay and decrease in throughput.

A few researches have been performed on parallel floating point accumulation, approximate arithmetic, compressor-based addition and pipelined architectures to enhance the arithmetic performance. Approximate arithmetic techniques alleviate the demands for numerical precision, reduce the hardware complexity, and still provide an acceptable level of computation accuracy. Likewise, carry preserve accumulation techniques also avoid carry propagation until the end of the arithmetic stages, thus decreasing the critical path delay. Multiple arithmetic stages can be executed simultaneously and throughput is further improved by pipelining.

Many approximate FPA architectures, however, incur some monotonicity violations and significant normalization overhead, leading to a loss of numerical stability and hardware efficiency. The design of the FPGA-based floating point arithmetic which preserves monotonic arithmetic behavior and has low hardware complexity is one of the problems that have been difficult.

This paper presents an approximate floating point adder with pipelined highlevel architecture to overcome these drawbacks, through using the lightweight single-stage normalization and carry-

preserving parallel accumulation. The proposed architecture will perform parallel addition of four parallel aligned mantissas by using a parallel carry-preserving adder tree which will decrease arithmetic depth and boost throughput. A simple normalisation approach is proposed to reduce area consumption without sacrificing the near-monotonicity of the arithmetic operations. To minimize critical path delay and enhance operating frequency, pipeline registers are added.

The proposed design has been coded in Verilog HDL and synthesized on Cyclone V FPGA platform.Waveform analysis and behavioral simulation were performed using ModelSim to verify the correctness of each design module [11]. The experimental results show that the proposed architecture is effective for implementing high-performance floating-point accumulation applications with low hardware utilization, low normalization overhead, high arithmetic throughput and efficient FPGA implementation.

II. RELATED WORK

Many scientific computing, signal processing, graphics processing, machine learning accelerators and FPGA-based embedded systems are built on the computational backbone of floating point arithmetic. While conventional IEEE754 floating point arithmetic is highly accurate and has uniform computation rules, the high hardware complexity, carry propagation delay of the arithmetic operations and the necessary normalization overhead reduce the arithmetic performance of high speed digital systems [1, 2].

There have been several studies on the design of high speed floating point arithmetic architectures to increase throughput and decrease arithmetic latency. Arithmetic algorithms and efficient floating point computing hardware architectures were extensively discussed by Parhami and Koren [3, 4]. The theoretical basis of parallel arithmetic structures, carry-save accumulation, and compressor-based arithmetic units, which are key components of modern VLSI-based arithmetic systems, were set by their work.

The traditional sequential floating point adders are serial in performing the functions of exponent comparison, normalizing the mantissa, aligning the mantissa and adding the mantissas, rounding, etc. These architectures are correct but have the disadvantage of repeated carry propagation and iterative normalization stages which result in large combinational delay. To address these deficiencies, carry-save and carry-preserving accumulation techniques have been developed to decrease arithmetic depth and increase computation speed [3, 7] respectively.

Because of its ability to decrease hardware complexity and enhance the computational efficiency, approximate computing has become an important research field. In [5] ZHU et al. presented low-power approximate adders for applications where errors are tolerable, and showed that area and propagation delay can be reduced substantially. Likewise, Mahdiani et al. [6] developed imprecise computational blocks for fast VLSI implementation of soft computing applications. These techniques have shown that controlled approximation can be used to get a useful gain in arithmetic efficiency without sacrificing significant accuracy in the computation.

Han and Orshansky [14] also pointed out that approximate computing is an effective solution for the tradeoff between computational accuracy and hardware efficiency in energy limited digital systems. Venkataramani et al. [15] suggested programmable approximate architectures that can dynamically trade-off accuracy of Arithmetic and computational performance. These works together very clearly showed the increasing significance of approximation techniques in contemporary designs of arithmetic accelerators.

In the literature, also a few implementations of floating-point arithmetic using FPGAs have been reported. Bhardwaj and Mane [13] used IEEE754 floating point adder on FPGA platform and showed its effectiveness for accelerating the floating point arithmetic in hardware. Usselmann [12] suggested an open-source IEEE floating point arithmetic unit, that provides standard floating point operation support for FPGA systems. Most existing FPGA implementations, however, are mainly oriented to the correctness of the computation and not to efficient approximation of the function and simple normalization strategies which are hardware-friendly.

However, current architectures still have problems of complexity of the normalizing function, propagation delay, large area usage and monotonicity preservation. Numerous approximate arithmetic architectures suffer from monotonicity violations, leading to numerical instability and unpredictable arithmetic behavior of iterative computation systems.

The proposed work aim to overcome these limitations by proposing a pipelined carry preserving approximate floating point accumulation architecture for a single stage normalization. The proposed design is different from the conventional sequential floating point adders because of the simultaneous accumulation of four operands by adopting a parallel carry-preserving adder tree. In addition, the proposed normalization method has a significantly lower hardware overhead compared to maintaining the IEEE754 behavior while keeping it near-monotonic. The synthesis results on the FPGA shows that the architecture provides very high arithmetic throughput and very low area utilization with only 334 ALMs and 25 registers on a Cyclone V FPGA device

III. PROPOSED ARCHITECTURE

The proposed architecture aims to provide high-speed floating point accumulation whilst maintaining as low an overall complexity of the hardware as possible, and as efficient as possible in terms of FPGA

usage. The architecture implements a carry-preserving parallel accumulation approach and lightweight normalization technique to add four IEEE754 single precision FP operands simultaneously. The goal of the overall design is to minimize carry propagation delay, reduce the normalization overhead, maintain near-monotonic arithmetic operations, and increase throughput by executing the operation in a pipelined fashion. The whole architecture is composed of the following components: exponent extraction, exponent comparison, mantissa alignment, carry-preserving parallel adder tree, pipeline register stage, simplified normalization unit, and rounding circuitry. The arithmetic stage is designed to minimize FPGA resources consumption and minimize combinational delay.

First the exp fields of all the operands of the floating point operations are extracted and passed to the exponent comparison module. This module finds the highest order of all the operands given as inputs with parallel comparator logic. The maximum exponent is used as the reference exponent for alignment of mantissa. The remaining operands are then compared to the reference exponent and the difference(s) calculated, which determines how many right-shift operations are necessary. The mantissa alignment module shifts the mantissas to the right, if necessary, based on the calculated difference in exponents. When floating point numbers are used, the alignment of mantissa allows all floating number to be represented in the same exponent prior to arithmetic accumulation. The FPGA synthesis results showed that the biggest hardware overhead in the proposed architecture is the alignment of mantissa, which is caused by the use of barrel shifting structures and multiplexing circuitry.

Once aligned, the mantissas go to the carry preserving parallel adder tree for processing. This is the main contribution of the work proposed for this module. The proposed architecture is suitable for simultaneous multi-operand accumulation rather than the sequential addition of operands that is implemented in the conventional architecture. The aligned mantissas are processed in pairs and the processing is realized through compressor-based accumulation logic with separate storage of the carry information during intermediate processing steps. Because the carry propagation is delayed to the last arithmetic stage, the critical path delay is greatly minimized.

The final carry-propagating adder is used to combine the intermediate sum and carry vectors that were generated by the adder tree to form the final accumulated result. This stage can be enhanced with carry-lookahead arithmetic structures to provide better arithmetic performance.

A pipeline register stage is added to the accumulation stage and normalization stage to increase throughput and operating frequency. Arithmetic operation is separated into independent timing stages in the pipeline registers, thereby minimizing the combinational delay[9] and allowing the temporal parallelism. The design follows standard synchronous digital design principles [9] while employing pipelining techniques to achieve temporal parallelism and higher throughput, as discussed by Hennessy and Patterson [8].

For the normalization stage, a simplified single stage normalization method is used instead of complicated iterative normalization methods used in traditional IEEE754 adders. The lightweight normalization module reduces hardware complexity without affecting the arithmetic behavior in a non-near-monotonic manner. Quartus synthesis results suggest the normalization module requires only a small hardware overhead, around 0.5 ALMs. This observation confirms the effectiveness of the simple normalization method proposed.

Ultimately, the rounding unit formats the number in IEEE754, and outputs the final single-precision floating point number. A new architecture was synthesized using Verilog HDL and Intel Quartus Prime[10], for a Cyclone V FPGA device..

A. Methodology

The pipelined parallel processing method proposed in the floating point accumulation architecture is to achieve the maximum arithmetic throughput and reduce the complexity of hardware. The technique is a mix of carry preserving arithmetic, approximate normalization, and pipelined execution to achieve an efficient balance of speed/area and numerical accuracy . The Arduino operation starts with the decomposition of the operands following IEEE754. The sign field contains the sign of the 32-bit floating point operand followed by the exponent field, and then the mantissa field. The largest exponent of all operands is then found in the exponent comparison stage. The maximum exponent is chosen as the reference exponent because it is required to represent the exponents of floating point numbers.

The mantissa alignment stage calculates differences of exponents and conditionally right shifts the mantissa. This operation makes all operands have the same exponent as a reference value. The matched mantissas are shown as:

$$M\_aligned = M \times 2\text{^}(-\Delta E)$$

where ΔE is the difference in the exponents of the operands and reference.

The carry preserving parallel accumulation stage then adds simultaneously all aligned mantissas. The proposed design does not carry the carry bits through the intermediate stages but stores them as intermediate bits. This approach substantially decreases the arithmetic depth and it will minimize the critical path delay. The carry-preserving accumulation process can be represented as:

$$S\_total = S\_intermediate + C\_intermediate$$

where S_intermediate and C_intermediate are intermediate sum, and carry vectors produced by accumulator logic implemented using a compressor.

To minimize combinational timing delay, a pipeline register stage is added after the accumulation stage.

The pipeline method is a way of breaking the arithmetic operations in several timing stages, so that independent arithmetic operations can be performed concurrently. Compared to fully combinational sequential architectures, this will improve the operating frequency and the throughput. An approximate lightweight normalization technique is used in the normalization stage. Normalization of conventional floating point adders needs to be performed by using multiple shift operations and iterative leading-zero detection to get the correct answer. These operations make hardware more complex and cause a propagation delay to be greater. The proposed architecture makes the normalization task simple by employing a single-stage normalization approach that is of reduced complexity. Near-monotonic arithmetic behaviour is maintained at the expense of a small approximation error, but FPGA resources are significantly lowered.

IV. Result And Discussion

On the Cyclone V FPGA device, Intel Quartus Prime Lite Edition was used to successfully synthesize the proposed architecture. The experimental result shows that the proposed architecture is able to meet high arithmetic performance while using very small hardware resources.

The entire design only requires 334 Adaptive Logic Modules (ALMs), which is less than 1% of the FPGA resources used in the targeted device. The architecture also needs just 25 registers, showing a very efficient pipelined implementation.

As per the synthesis report, the mantissa alignment stage has the maximum hardware overhead in the overall design. The logic for shifting and computing exponent difference in each mantissa alignment module uses about 67-69 ALMs. By comparison, the simplified normalization strategy only requires 0.5 ALMs for the normalization module, which shows the success of the simplified normalization approach.

The carry-preserving adder tree has the excellent property of minimizing the propagation delay due to carries, which allows for a significant increase in the arithmetic throughput. Pipelined execution allows higher operating frequency and allows temporal parallelism. Based on timing analysis, the proposed design can be used to operate at a high speed and has stable arithmetic properties.

The simulation results validate the architecture and show that the architecture remains monotonic when the operand data values are increasing. The error of approximation is still relatively small for most of the test cases, as revealed by the error analysis. The proposed design thus obtains a practical balance between the efficiency and the arithmetic accuracy.

The FPGA synthesis results are summarized in Table I. The proposed architecture utilizes only 334 ALMs and 25 registers, occupying less than 1% of the Cyclone V FPGA resources. No DSP blocks, PLLs, or on-chip memory are required, demonstrating the hardware efficiency of the design.

TABLE I. FPGA Resource Utilization

Parameter	Value
FPGA Device	Cyclone V
Total ALMs Used	334
FPGA Utilization	< 1%
Registers Used	25
Total Pins	161
PLL Usage	0
DSP Blocks	0
Block Memory Bits	0

TABLE II. Module-wise ALM Utilization

Module	ALMs Used
Exponent Comparison	25
Mantissa Align 1	67.5
Mantissa Align 2	69
Mantissa Align 3	68.5
Mantissa Align 4	62
Adder Tree	23
Pipeline Register	6.5
Normalize Nonmono	0.5

Table II presents the ALM utilization of individual modules. The mantissa alignment units consume the highest resources, while the proposed Normalize Nonmono module requires only 0.5 ALMs, confirming the effectiveness of the lightweight single-stage normalization approach.

As shown in Table III, the proposed architecture achieves lower latency and higher throughput than conventional IEEE 754 adders while maintaining monotonic behavior. This demonstrates an effective trade-off between hardware efficiency, computational performance, and numerical accuracy.

TABLE III. Performance Comparison

Architecture	Latency	Throughput	Accuracy	Monotonicity
Sequential IEEE754 Adder	High	Low	Exact	Yes
Parallel Monotonic Adder	Medium	Medium	Near Exact	Yes
Proposed Approxi	Low	High	Approximate	Maintained

mate Architecture				

V. Conclusion

This paper proposed the architecture of carry-preserving pipelined floating point adder architecture which has a simplified normalization process for the FPGA based arithmetic acceleration. The design will be a mixture of parallel accumulation, pipelined execution, approximate arithmetic and light weight normalization to achieve high throughput with minimum hardware. The experimental results indicate that the proposed architecture uses less than 1% of the resources of the FPGA while retaining the arithmetic monotonicity. The normalization strategy is simplified, hence the hardware utilization for normalization is very low.An efficient balance between computational accuracy, throughput and FPGA area efficiency is obtained. Hence, the proposed architecture is very well suited for high speed signal processing, approximate computing accelerators, machine learning systems and arithmetic processors based on FPGAs. Future work could include expansion of the architecture for double precision arithmetic, adaptive approximation control and ASIC level low-power optimization.

References

[1] D. Goldberg, "What Every Computer Scientist Should Know About Floating-Point Arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, Mar. 1991

[2] IEEE Computer Society, *IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2019, New York, NY, USA: IEEE, Jul. 2019

[3] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. New York, NY, USA: Oxford University Press, 2010.

[4] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed. Natick, MA, USA: A K Peters, 2002.

[5] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant applications," in *Proc. IEEE Int. Symp. Integrated Circuits (ISIC)*, Singapore, 2009, pp. 69–72.

[6] H. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010

[7] M. J. Schulte and E. E. Swartzlander Jr., "Hardware designs for exactly rounded elementary functions," *IEEE Trans. Computers*, vol. 43, no. 8, pp. 964–973, Aug. 1994..

[8] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed. Burlington, MA, USA: Morgan Kaufmann, 2017.

[9] S. Lee, *Digital Circuits and Logic Design*. Upper Saddle River, NJ, USA: Prentice Hall, 2007.

[10] Intel Corporation, *Quartus Prime Lite Edition User Guide*. Santa Clara, CA, USA: Intel Corporation, 2025.

[11] Siemens EDA (formerly Mentor Graphics), *ModelSim User Manual*. Wilsonville, OR, USA: Siemens EDA, 2024.

[12] R. Usselmann, "IEEE Floating Point Arithmetic Unit," OpenCores Project, 2010. [Online]. Available: https://opencores.org/projects/fpu

[13] K. Bhardwaj and P. Mane, "Design and FPGA implementation of IEEE-754 floating point adder," *International Journal of Engineering Research and Technology (IJERT)*, vol. 5, no. 6, pp. 112–117, Jun. 2016.

[14] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Proc. 18th IEEE European Test Symposium (ETS)*, Avignon, France, 2013, pp. 1–6.

[15] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: A unified design paradigm for approximate and quality-configurable circuits," in *Proc. 46th Annual IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Davis, CA, USA, 2013, pp. 136–147.

Author Profile

Mr. MOHD ABDUL AZIZ, M.Tech student in ECE (VLSI System Design) from Shadan College of Engineering And Technology, Peerancheru, Telangana.

Dr. AMAIRULLAH KHAN LODHI, Professor, R & D Coordinator, Dept. of ECE from Shadan College of Engineering and Technology, Peerancheru, Telangana.

Mr. H. A. ABDUL SAMAD, Assistant Professor, Dept. of ECE from Shadan College of Engineering and Technology, Peerancheru, Telangana.