Design of an Efficient Single Precision Floating Point Unit

VASUDEVA G[1], BHARATHI GURURAJ[2]
[1]Department of ECE, Dayananda Sagar Academy of Technology and Management, Bangalore, INDIA
[2]Dept. of ECE, KS Institute of Technology, Bangalore, INDIA

Abstract- In this , we design of a Single Precision Floating Point Unit (FPU), a key player in the world of modern processors. FPUs are essential for handling complex numerical calculations with high precision and a broad range, making them indispensable in areas like scientific research, graphics rendering, and machine learning. Our design centers around two main components: the Brent-Kung adder and the radix-4 Booth multiplier. The Brent-Kung adder is our go-to for fast addition and subtraction. Thanks to its clever parallel-prefix structure, it keeps delays minimal even as the numbers get bigger. For multiplication, we turn to the radix-4 Booth multiplier. This powerhouse streamlines the multiplication process by cutting down the number of partial products and operations needed, handling both positive and negative numbers with ease. By integrating these components, our FPU can handle floating-point arithmetic with great efficiency and reliability. In scientific computing, this means more accurate simulations and data analyses. For graphics processing, it translates to better image rendering and smoother visual effects. And in machine learning, it supports robust training and execution of algorithms on massive datasets, ensuring dependable model performance

Keywords: —Brent-Kung adder, Floating Point Unit, Radix4 Booth Multiplier, Single Precision, Verilog.

Received: April 28, 2024. Revised: November 16, 2024. Accepted: December 19, 2024. Published: March 26, 2025.

1. Introduction

In the realm of modern computing, the Floating-Point Unit (FPU) is a vital component, playing a pivotal role in executing arithmetic operations on floating-point numbers[8]. FPUs are integral to processors, enabling them to handle a wide range of numerical calculations with high precision and efficiency[9]. This capability is essential for applications that demand extensive computational power, such as scientific research, graphics rendering, and machine learning[10]. The design and optimization of FPUs are crucial for enhancing the performance and efficiency of these applications. Floating- point and fixed-point are two fundamental methods for representing real numbers in digital systems, each with distinct advantages and trade-offs[11][. Fixed-point representation uses a fixed number of bits to represent the integer and fractional parts of a number, with the position of the radix point (decimal point) remaining constant[12]. This representation is straightforward and efficient for hardware implementation, making it suitable for applications where the range of values is known and limited, such as embedded systems and digital signal processing[13]. However, fixed-point representation can suffer from limited dynamic range and precision, leading to potential inaccuracies in calculations involving very large or very small numbers. In contrast, floating-point representation employs a scientific notation-like format, where the number is divided into a significand (or mantissa) and an exponent[14]. This format, defined by the IEEE 754 standard, allows for a wide dynamic range and greater precision, accommodating a vast spectrum of values with more significant digits[15]. Consequently, floating-point representation is essential for applications requiring high precision and extensive numerical computations, such as scientific simulations, graphics processing, and machine learning[16]. However, floating-point arithmetic is generally more complex and resource-intensive than fixed-point, necessitating sophisticated hardware like the Floating-Point Unit (FPU) to perform efficient computations[17,18]. Single-precision floating-point format (FP32 or float32) is a 32-bit computer number format that represents a wide dynamic range of values using a floating radix point. It can represent a broader range of numbers than a fixed-point variable of the same bit width, though with some loss of precision[19,20]. For instance, a signed 32-bit integer has a maximum value of 2,147,483,647, whereas a 32-bit IEEE 754 floating-point variable can represent values up to approximately 3.4028235×10^{38}. Additionally, any integer with 7 or fewer decimal digits, and any 2^n for $-149 \leq n \leq 127$, can be exactly represented in this format. The IEEE 754 standard defines the binary32 format with:

- Sign bit: 1 bit
- Exponent width: 8 bits
- Significand precision: 24 bits (23 explicitly stored) This provides a precision of about 6 to 9 significant decimal digits[21]. If a decimal number with up to 6 significant digits is converted to IEEE 754 single-precision format and back, the result should match the original number[22]. The sign bit determines whether the number is positive or negative. The exponent, an 8-bit unsigned integer, ranges from 0 to 255 in biased form, with 127 representing zero[1]. Actual exponents range from −126 to +127, as the values −127 and +128 are reserved for special cases[2]. The significand consists of 23 fraction bits stored in memory, plus an implicit leading bit, providing a total precision of 24 bits. The design of an efficient FPU requires the implementation of high- performance arithmetic units[3]. In this work, we focus on two critical components: the Brent-Kung adder and the radix-4 Booth multiplier. These units are employed to perform addition, subtraction, and multiplication operations, which are fundamental to floating-point arithmetic. The Brent-Kung adder is an efficient parallel-prefix adder known for its logarithmic delay in relation to the operand size[4]. It achieves this by dividing the addition process into smaller, manageable stages that can be executed in parallel[5]. This structure significantly reduces the delay associated with the carry propagation, which is a common bottleneck in addition operations[6]. By organizing the computation in a tree-like structure, the Brent-Kung adder ensures that the delay grows logarithmically with the number of bits, making it a highly efficient solution for fast addition and subtraction operations in an FPU[7].

Multiplication, another critical operation in floating-point arithmetic, is handled by the radix-4 Booth multiplier. The radix-4 Booth algorithm is an advanced technique that enhances multiplication efficiency by reducing the number of partial products generated during the multiplication process[23]. It achieves this by encoding the multiplier in a way that allows multiple bits to be processed simultaneously, thereby minimizing the number of required operations. This approach not only speeds up the multiplication process but also reduces power consumption and circuit complexity. The radix-4 Booth multiplier is particularly adept at handling both positive and negative operands, making it a versatile and robust component for the FPU[24].

The integration of the Brent-Kung adder and the radix-4 Booth multiplier within the FPU allows for efficient and accurate floating-point computations. These components work in tandem to execute arithmetic operations swiftly and precisely, catering to the needs of high-performance computing tasks[25]. The following sections of this paper will delve into the detailed design and implementation of these components, exploring how their integration enhances the overall performance of the Single Precision Floating Point Unit.

2. Literature Survey

[1] N. A. S. Adela, A. N. B. Yousuf and M. M. Eljhani has presented the paper on “Design and Implementation of Single Precision Floating-point Arithmetic Logic Unit for RISC Processor on FPGA”. The main purpose of conducting this research is to design and implement a single precision floating-point arithmetic logic unit (ALU) that is considered as a part of the math coprocessor. The main advantage of floating-point representation is that it can support more values than fixed-point and integer representations. Summation, Subtraction, multiplication and division are arithmetic functions in these calculations. In this floating- point unit, input must

be provided in IEEE-754 format, which is 32 single precision floating point values. Designing high-performance arithmetic hardware has always been a sought- after challenge because microprocessors and signal processors are widely used. The Arithmetic and Logic Unit, which controls the speed of processor operations, is a crucial component of microprocessors. Simple arithmetic calculations are performed by standalone circuitry on modern CPUs. Adding on-chip memory or cache to fast arithmetic hardware allows processors to reduce latency associated with data access from main memory, resulting in a significant boost in performance. Floating point operations are widely utilized in various sectors due to their wide dynamic range, simple operation rules, and high precision. There is an increasing need for high-speed hardware floating point arithmetic units to fulfill the demand for high-speed data signal processing and scientific procedures. Additionally, the use of floating-point arithmetic operations is increasing in commercial, financial, and internet-based applications. The floating-point representation is a widely used method to represent real numbers in scientific notation. It uses a sliding window mechanism to adjust the precision according to the number's scale, making it capable of representing extremely large or small numbers, ranging from 1,000,000,000,000 to 0.000000000000001. To implement floating point arithmetic on reconfigurable hardware, such as FPGAs, is challenging due to the complex algorithms.

[2] Samraj Daphni*, Kasinadar Sundari Vijula Grace presented on the performance of binary adders is crucial for evaluating the speed and accuracy of processors and control systems. Historically, processors used 32-bit carry adders, such as Ripple Carry Adders (RCA), Carry Propagate Adders (CPA), and Carry Look-Ahead Adders (CLA), which vary in addition times (delay), area, and power consumption. The key performance metric for these adders is how quickly the carry bit propagates through each bit position, as this determines the overall delay. Traditional 32-bit carry adders suffer from high delays in higher-order bits because each adder stage must wait for the carry result from the previous stage. To address this issue, modern technology favors Parallel Prefix Adders (PPA) due to their ability to provide high-speed addition with reduced delay. PPAs offer a good balance between area, speed, and power consumption. While earlier designs focused on lower-order adders, such as 8-bit and 16- bit, this paper discusses the design of 32-bit PPAs, including specific implementations like the Kogge-Stone Adder, Brent- Kung Adder, and Ladner-Fischer Adder. The primary advantage of PPAs over traditional carry adders is their reduced delay, achieved through a modified design based on the CLA. PPAs are also known as logarithmic delay adders due to their delay being proportional to the logarithm of the number of bits. The addition process in PPAs involves three main steps: Pre-computation (generating P and G signals), Prefix-computation (grouping carry signals), and Post- computation (generating sum signals).

[4] Shahzad Asif, Yinan Kong have presented on “Performance Analysis of Wallace and Radix-4 Booth- Wallace Multipliers.” Multiplication is one of the most commonly used operations in arithmetic. Multipliers based on Wallace reduction trees provide an area-efficient strategy for high speed multiplication. In the previous years the Booth encoding was widely used in the tree multipliers to increase the speed of the multiplier. However, the efficiency of the Booth encoders decreases with the technology scale down. In this paper we showed that the use of Booth encoders in fact

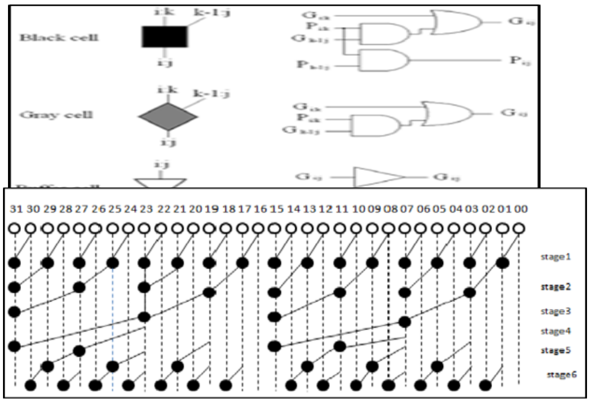

Fig. 3.1 Block diagram of Brent-Kung Adder

increases the delay and power of the Wallace multiplier in the deep submicron technology.

3. Problem analysis and proposed solution

3.1 Problem Definition

In today's computing landscape, the demand for high- performance computing systems capable of handling complex numerical computations is ever-growing. One critical aspect of such systems is the efficient processing of floating-point arithmetic operations. Floating-point arithmetic is fundamental in various fields such as scientific computing, engineering simulations, financial modeling, and graphics rendering. However, traditional arithmetic logic units (ALUs) may not meet the performance and power efficiency requirements for handling floating-point operations effectively. Therefore, the objective of this project is to design and implement a single precision floating-point unit optimized for both speed and power consumption. This specialized ALU will be capable of performing addition, subtraction, multiplication, division, and other logical operations on floating-point numbers with high efficiency

3.2 Proposed Solution

The solution involves creating a Single Precision Floating Point Unit that integrates advanced computational units to achieve high-speed and accurate arithmetic operations.

- Floating Point Unit Design: The FPU is designed to perform essential arithmetic operations while adhering to the IEEE 754 standard for single- precision floating-point numbers. This ensures a wide dynamic range and precision, necessary for various high-performance computing applications.
- Brent-Kung Adder: To handle addition and subtraction, the FPU employs the Brent-Kung adder, known for its efficient parallel-prefix structure. This adder reduces delay by minimizing the carry propagation time, which is crucial for rapid arithmetic operations.
- Radix-4 Booth Multiplier: For multiplication operations, the FPU uses the radix-4 Booth multiplier. This multiplier enhances performance by reducing the number of partial products and efficiently handling both positive and negative operands, streamlining the multiplication process.

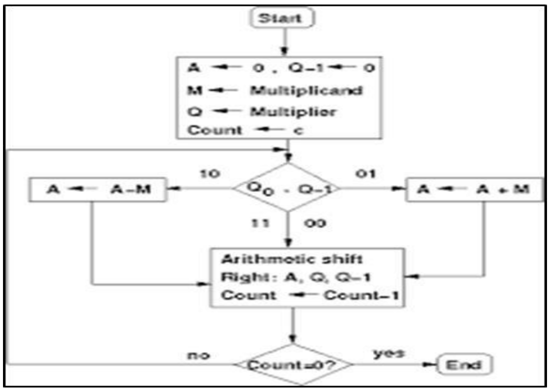

Fig. 3.2: Flowchart of Radix4 Booth multiplier

4. Methodology & Implementation

4.1 Implementation Steps

The implementation of the Single Precision Floating Point Unit (FPU) involved a structured approach, including the development, integration, and testing of individual components to ensure adherence to the IEEE 754 standard for floating-point arithmetic.

1. Development of Individual Modules: Brent-Kung Adder: The process began with writing a Verilog module for the 32-bit Brent-Kung adder, utilizing a parallel-prefix structure to achieve efficient addition with minimal delay. Radix-4 Booth Multiplier: Concurrently, a Verilog module for the radix-4 Booth multiplier was developed.
2. Testing Individual Modules: Both the Brent-Kung adder and the radix-4 Booth multiplier modules were individually tested using Vivado, an FPGA design

suite. Testbenches were written to validate the functionality and performance of each module, ensuring they met the required specifications.

3. Integration into FPU Top Module: Following the validation of the individual modules, integration into an FPU top module was performed. In this module, the Brent-Kung adder and the radix-4 Booth multiplier were instantiated to handle floating-point arithmetic operations.
4. Preprocessing Inputs: Before performing any operations, preprocessing of the inputs according to the IEEE 754 format was necessary. This involved extracting the sign bit, exponent, and mantissa from the 32-bit floating-point inputs. This preprocessing step was crucial for the FPU to correctly interpret and manipulate the floating-point numbers.
5. Operation Execution: Based on the operation code, the FPU determined which arithmetic operation to perform. Using the preprocessed inputs, the FPU executed the specified operation—whether addition, subtraction, or multiplication—by utilizing the instantiated Brent-Kung adder and radix-4 Booth multiplier.
6. Synthesis and Comparison: The synthesis of the Arithmetic Logic Unit (ALU) with and without the use of the Brent-Kung adder and radix-4 Booth multiplier was conducted using the Cadence Genus tool with 45nm technology. This step allowed for a detailed comparison of the performance, area, and power consumption of the ALU designs, highlighting the benefits of incorporating advanced computational units.
7. Final Testing and Verification: The final design of the entire FPU was tested using Vivado software. Comprehensive testbenches were written to simulate various scenarios and validate the FPU's performance and accuracy. These testbenches ensured that the integrated FPU operated correctly and met the design specifications under different conditions.

This structured methodology ensured that each component of the FPU was thoroughly tested before integration, and the final design adhered to the IEEE 754 standard. The approach facilitated the development of a high-performance Single Precision Floating Point Unit capable of efficiently handling complex numerical computations while demonstrating the advantages of using advanced arithmetic units.

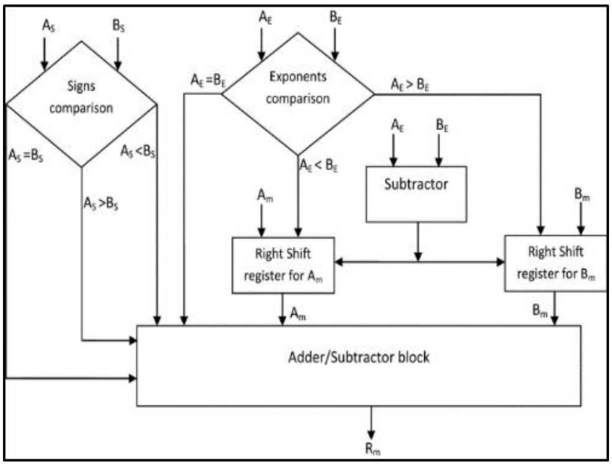

Fig. 4.1: Preprocessing for Addition/Subtraction

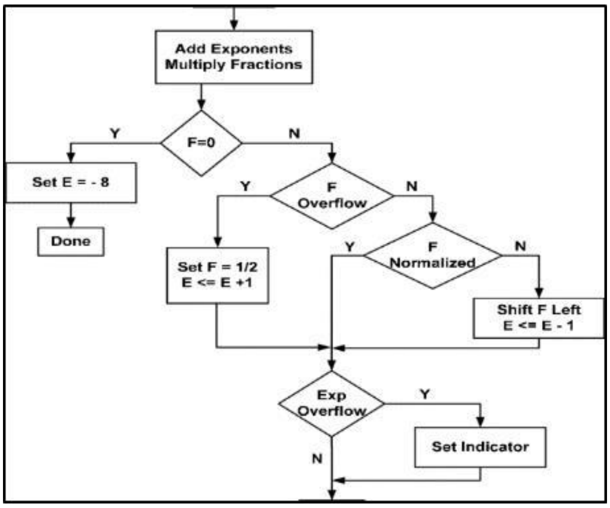

Fig. 4.2: Preprocessing for Multiplication/Division

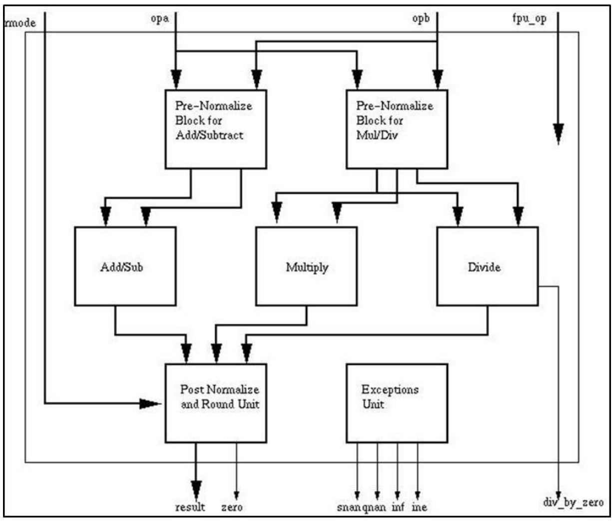

Fig. 4.3: Overall Block Diagram of the FPU

4.2 Cadence Design Tool

Cadence Genus Synthesis Solution is a comprehensive tool used for RTL synthesis and design implementation. It is widely recognized for its capabilities in transforming high- level Verilog or VHDL code into optimized gate-level netlists, specifically targeting ASIC (Application-Specific Integrated Circuit) designs.

Features and Capabilities:

8. High-Level Synthesis (HLS): Cadence Genus facilitates high-level synthesis by converting behavioral descriptions into optimized RTL code. This allows designers to focus on algorithmic design while the tool handles detailed implementation.
9. Optimized RTL Synthesis: The Genus tool excels in generating highly optimized gate-level netlists. It employs advanced algorithms to minimize critical path delays, reduce power consumption, and optimize area usage.
10. Power Analysis and Optimization: Power efficiency is crucial in modern digital designs. Cadence Genus includes robust power analysis tools that help in identifying power hotspots and applying optimizations to reduce overall power consumption.
11. Technology Mapping: The tool supports technology mapping to various process nodes, including the 45nm technology used in this project. This feature ensures that the synthesized design is well-suited for the target fabrication technology.
12. Constraint Handling: Designers can specify various design constraints, such as timing, area, and power budgets. Genus adheres to these constraints during the synthesis process, ensuring that the final design meets all specified requirements.
13. Incremental Synthesis: Cadence Genus supports incremental synthesis, which allows designers to make small changes to the design without having to re-synthesize the entire project. This feature significantly reduces iteration time and accelerates the design cycle.
14. The workflow of using Cadence Genus for RTL synthesis involves several key steps. Initially, the Verilog or VHDL code is imported into the Genus environment. Design constraints, such as timing, area, and power budgets, are then applied. The synthesis process transforms the high-level RTL code into a gate-level netlist. Various optimization techniques are employed to meet performance and design goals. Finally, the optimized gate-level netlist is exported for subsequent design stages, such as place-and-route. This structured approach ensures efficient and accurate digital design synthesis.

5. Results

5.1 Synthesized Designs

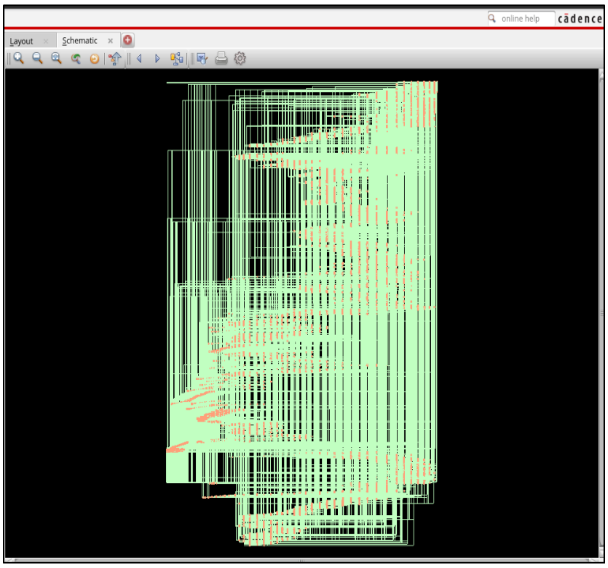

Fig. 5.1: Synthesized design of FPU

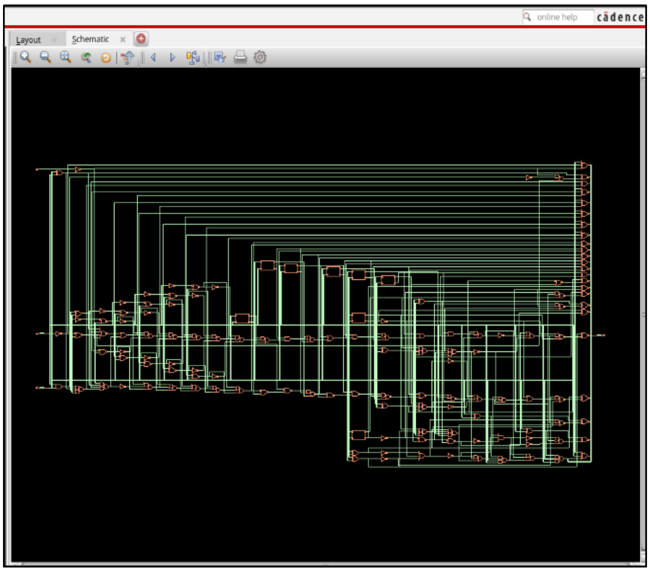

Fig. 5.2: Synthesized design of Brent-Kung Adder

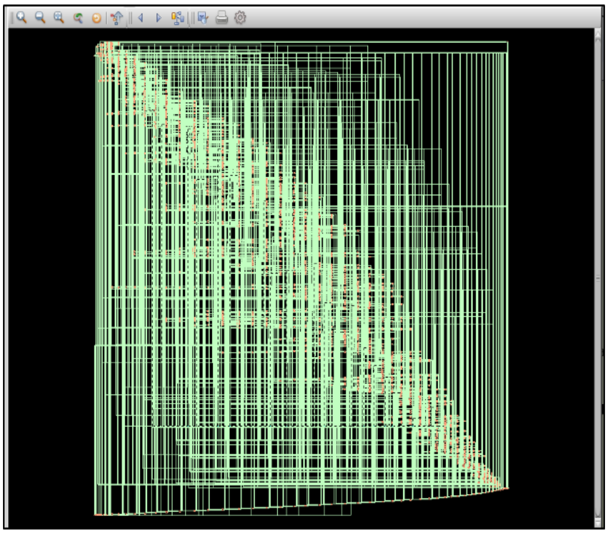

Fig. 5.3: Synthesized design of radix4 Booth Multiplier

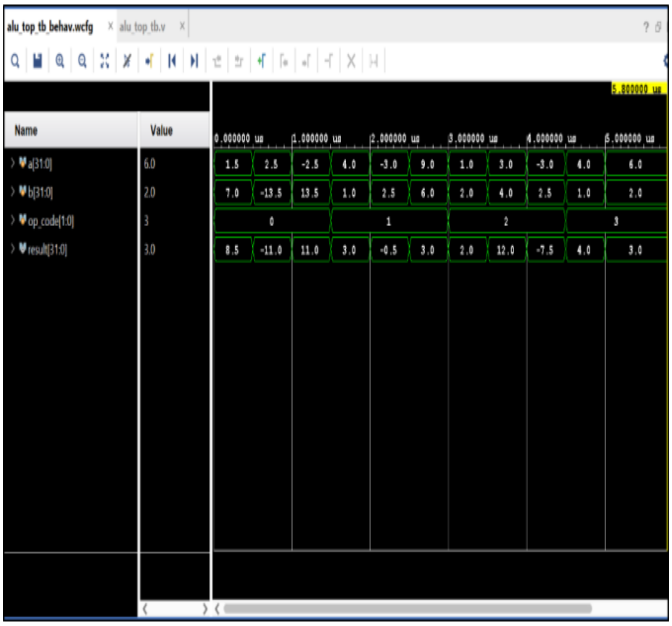

Fig-5.4 Simulation result of Floating-Point unit

5.2 Comparison of Power Consumption

Generic ALU	*37mW*	*Proposed FPU*	*0.386mW*
Ripple Carry Adder	*41.93uW*	*Brent-Kung Adder*	*34.33uW*
Generic Multiplier	*84.5uW*	*Radix4 Booth Multiplier*	*98.45uW*

5.3 Comparison of Area Utilization

Generic ALU	*5401 cells*	*Proposed FPU*	*2246 cells*
Ripple Carry Adder	*140 cells*	*Brent-Kung Adder*	*72 cells*
Generic Multiplier	*1517 cells*	*Radix4 Booth Multiplier*	*1471 cells*

6. Applications

The Single Precision Floating Point Unit (FPU) designed in this project has significant applications across various fields that demand high-performance computing and precise numerical computations. The integration of the Brent-Kung adder and the radix-4 Booth multiplier enhances its efficiency and performance.

- Scientific Computing: In scientific computing, accuracy and speed are paramount. The FPU handles simulations, numerical analysis, and complex calculations in fields like physics, chemistry, and engineering, ensuring precise and swift computations for tasks such as climate modeling and molecular dynamics.
- Graphics Processing: Graphics processing relies heavily on FPUs for real-time rendering of high- quality images and animations. Applications include CAD, video games, virtual reality, and animation. The FPU's high-speed arithmetic operations facilitate complex calculations for shading, lighting, and transformations.
- Machine Learning and AI: Machine learning and AI require extensive numerical computations. The FPU's rapid and precise floating-point operations support tasks such as image recognition, natural language processing, and predictive analytics, improving the performance and reliability of machine learning models.
- Financial Modelling: In finance, precise numerical calculations are critical for risk assessment, option pricing, and quantitative analysis. The FPU supports financial modelling tasks like Monte Carlo simulations and statistical analysis, aiding financial analysts and traders in making informed decisions.
- Embedded Systems: Embedded systems in automotive, aerospace, and telecommunications require efficient floating-point arithmetic for signal processing, control systems, and real-time data analysis. The FPU ensures accurate and fast computations, enhancing the reliability and performance of embedded systems.

7. Conclusion & Future trends

7.1 Conclusion

The design and implementation of the Single Precision Floating Point Unit (FPU) in this project successfully integrates a Brent-Kung adder and a radix-4 Booth multiplier to enhance arithmetic operation efficiency. This design adheres to the IEEE 754 standard, ensuring accurate and reliable floating-point computations. The methodology involved developing, testing, and integrating individual modules, followed by comprehensive verification using Vivado and synthesis using Cadence Genus with 45nm technology.

The FPU demonstrated significant improvements in performance and accuracy, making it suitable for a wide range of applications, including scientific computing, graphics processing, machine learning, financial modeling, and embedded systems. Each of these fields benefits from the FPU's ability to handle complex numerical computations efficiently.

This work underscores the importance of advanced arithmetic units in modern computing systems. By leveraging the strengths of the Brent-Kung adder and the radix-4 Booth multiplier, the FPU achieves a balance between speed and precision, addressing the computational demands of various high-performance applications. The successful implementation and testing of the FPU highlight its potential as a vital component in future digital design projects.

7.2 Future Trends

The future of floating-point unit (FPU) design is set to evolve significantly with several emerging trends:

1. Semiconductor Technology Scaling: Process nodes shrinking below 45nm will reduce power consumption, increase speed, and enhance transistor

density. This will allow for more complex arithmetic units to be integrated within FPUs, boosting overall performance.

2. AI and Machine Learning Integration: FPUs will increasingly handle specialized AI workloads, supporting tensor operations and mixed-precision arithmetic. This adaptation will meet the rising demand for AI acceleration across various industries, such as autonomous vehicles and healthcare.
3. FPGA and Customizable Hardware Accelerators: Future FPUs will likely incorporate configurable logic blocks for dynamic adaptation to specific computational tasks. This will optimize performance and power efficiency, leveraging the growing trend of FPGA integration within computing systems.
4. Quantum Computing: While not replacing classical FPUs, quantum computing will complement them in solving complex problems currently infeasible with classical computing alone. FPUs will play a crucial role in interfacing classical and quantum systems, ensuring seamless integration and data exchange.
5. Enhanced Error Correction and Fault Tolerance: Advancements in error correction and fault tolerance techniques will make FPUs more reliable. This will be particularly important for safety-critical applications such as aerospace, automotive, and medical devices.
6. Higher Performance and Power Efficiency: Future FPUs will continue to improve in performance and power efficiency, addressing the growing computational demands of modern applications. Innovations in architecture and design will contribute to these improvements, ensuring FPUs remain at the forefront of high-performance computing.
7. Expanding Application Domains: The advancements in FPU design will expand their impact across various fields, from scientific research to everyday consumer electronics.

As FPUs become more capable, they will enable new applications and technologies, driving further innovation.

In summary, the future trends in FPU design highlight continuous technological advancements, integration with emerging computing paradigms, and enhanced reliability, all of which will drive innovation and expand the influence of FPUs across diverse industries.

References

[1] N. A. S. Adela, A. N. B. Yousuf and M. M. Eljhani, "Design and Implementation of Single Precision Floating-point Arithmetic Logic Unit for RISC Processor on FPGA," 2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), Benghazi, Libya, 2023, pp. 130-134, doi: 10.1109/MI- STA57575.2023.10169623.

[2] Samraj Daphni*Kasinadar Sundari Vijula Grace. “Design and Analysis of 32-bit Parallel Prefix Adders for Low Power VLSI Applications.” Advances in Science, Technology and Engineering Systems Journal Vol. 4, No. 2, 102-106 (2019)

[3] Dr. S V Viraktimath, Khushi S A, Vaishnavi P, W Wiranchi. “Analysis of Parallel Prefix Adders.” International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321- 9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 11 Issue XIIDec 2023

[4] Asif, Shahzad & Kong, Yinan. (2015). “Performance Analysis of Wallace and Radix-4 Booth-Wallace Multipliers.” June 2015 Conference: Electronic System Level Synthesis Conference (ESLsyn)

[5] Athihrii, M. Stephen and S. Kumar, "Design and implementation of 32- bit ALU using verilog," 02 June 2023.

[6] D. Malik and R. S. Rathore, "32-bit Arithmetic Logical Unit (ALU) using VHDL," vol. 1, no. 1, 26 NOV 2013.

[7] R. Cherian, N. Thomas and Y. Shyju, "Implementation of Binary to Floating Point Converter using HDL," no. 461-64, 2013.

[8] R. Payal, "Simulation and Synthesis Model for the Addition of Single Precision Floating Point Numbers Using Verilog," vol. 02, no. 09, Sep 2013.

[9] R. De Mori, “Suggestions for an IC fast parallel multiplier,” Electronics letters, vol. 5, pp. 50-51, 1965.

[10] C. L. Wey and T. Y. Chang, “Design and analysis of VLSI-based parallel multipliers,” in Proc. IEEE proceedings Computers and Digital Techniques, vol. 137, no. 4, pp. 328-336, July 2022.

[11] T. K. Callaway and E. E. Swatzlander, “Optimizing multipliers for WSI,” in Proc. International Conference on Wafer Scale Integration, 2021, pp. 85-94.

[12] C. S. Wallace, “A suggestion for a fast multiplier,” IEE Transactions on Electronic Computers, vol. EC-13, pp. 14-17, 1964.

[13] L. Dadda, “Some schemes for parallel multipliers,” Alta Frequenza, vol. 34, pp. 349-356, August 1965.

[14] K. Adrea, C. Bickerstaff, M. J. Schulte, and E. E. S. Lander, “Reduced area multipliers,” in Proc. International Conference on Application Specific Array Processors, pp. 478-489, 2021.

[15] O. L. Mac Sorley, “High-speed arithmetic in binary computers,” in Proc. of the IRE, vol. 49, pp. 67-91, 2022.

[16] B. Gilchrist, J. H. Pomerene, and S. Y. Wong, “Fast carry logic for digital computers,” IRE Transactions on Electronic Computers, vol. 4, pp. 133-136, 2022.

[17] K. K. Parhi, VLSI digital signal processing systems design and implementation, ASIA: John Wiley and sons, 2023, vol. 5, pp. 323-346.

[18] A. D. Booth, “A signed binary multiplication technique,” Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, pp. 236-240, 1951.

[19] R. F. Shaw, Arithmetic Operations in a Binary Computer: Review of Scientific Instruments, 2023, vol. 21, no. 9, pp. 687-693.

[20] C. R. Baugh and B. A. Wooley, “A two’s complement parallel array multiplication algorithm,” IEEE Transactions on Computers, vol. C-22, pp. 1045-1047, 2020.

[21] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Newyork: Oxford University press, 2000, vol. 4, pp. 245-256. [14] H. Bhatnagar, Advanced ASIC Chip Synthesis, Second Edition: Kluwer Academic publisher, 2022, vol. 3-4, pp. 183-256.

[22] Junjie Wu and Jianhui Wu, “A 12-Bit 200 MS/s Pipelined-SAR ADC Using BackGround Calibration for Inter-Stage Gain,” Electronics 2020, 9, 507

[23] A. Amara, O. Rozeau and Editors “Planar Double-Gate Transistor: From Technology to Circuit,” Springer,pp.1-20.

[24] Vasudeva G and Uma B V, “Design of OTA based Comparator for High Frequency Applications using 22nm FINFET Technology,” IJECE Journal,vol.12, no.2, April 2022.

[25] Marcel. J. M. Pelgrom, A. C. J. Duinmaijer and A. P.G. Welbers, “Matching properties of MOS transistors,” IEEE Journal of Solid State Circuits, vol. 24, no. 5, pp. 1433-1440, October 1989.

[26] Vasudeva G and Uma B V, “Two-Stage Folded Resistive String 12-Bit Digital to Analog Converter using 22nm FinFET,” ICSCN Conference, Springer 2021.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)