DESIGN AND IMPLEMENTATION OF A SINGLE PRECISION FLOATING POINT ARITHMETIC UNIT ON FGPA

Safiya[1], Dr.M.Asha Rani[2]

[1,2]Department of Electronics and Communication Engineering/Jawaharlal Nehru Technological University Hyderabad/India

Abstract

This work presents the design and implementation of a single precision floating-point unit (FPU) on a Field-Programmable Gate Array (FPGA). FPGAs offer key strengths like hardware reconfigurability, parallel processing, and low-latency execution, making them ideal for custom arithmetic acceleration. Floating-point representation brings major benefits such as wide dynamic range and high numerical precision, allowing both very large and very small values to be handled efficiently. The FPU performs arithmetic operations including addition, subtraction, and multiplication, conforming to the IEEE-754 single precision (32-bit) standard. To speed up multiplication, the design uses Booth's radix-4 multiplication algorithm, which reduces partial products and improves throughput with lower hardware overhead. The unit is integrated as a dedicated math coprocessor to accelerate floating-point computations, which are essential for applications requiring high numerical accuracy and dynamic range, such as digital signal processing, scientific simulations, and real-time data analysis. Floating-point representation enables handling of very large or very small values with precision that fixed-point or integer units cannot achieve, making it suitable for iterative and computationally intensive tasks. The design is modelled using Verilog Hardware Description Language (HDL) and implemented on an Intel Altera FPGA platform. Synthesis results demonstrate efficient resource utilization and high throughput, validating the FPU's capability to deliver fast and accurate arithmetic processing for embedded systems.

Keywords: Floating point, IEEE-754, FPGA Design, Verilog HDL, Arithmetic Logic Unit, Single Precision, Booth's Radix-4 Multiplier

I. Introduction

Floating-point arithmetic units are fundamental components in modern digital systems that require high numerical precision and wide dynamic range computation. Applications such as digital signal processing, image and video processing, scientific computing, and embedded systems extensively rely on floating-point operations to achieve accurate and reliable results. As computational demands increase, the performance of arithmetic units—particularly multiplication—has become a critical factor influencing overall system speed, power consumption, and hardware efficiency.

Several research efforts have focused on improving the design of floating-point arithmetic units to meet performance and efficiency requirements. Earlier works primarily concentrated on optimizing floating-point addition and subtraction by reducing exponent comparison delay, mantissa alignment complexity, and normalization overhead. Other studies emphasized improving floating-point multiplication performance, as it is one of the most computation-intensive operations within an FPU. Booth encoding techniques, especially Radix-4 Booth multiplication, have been widely adopted in multiplier designs to reduce the number of partial products and improve computational speed while maintaining reasonable hardware complexity. Additionally, FPGA-based implementations of floating-point units have been explored due to their inherent parallelism, flexibility, and suitability for prototyping custom arithmetic architectures.

Despite these advancements, the implementation of a complete IEEE-754 compliant single precision floating-point arithmetic unit on low-cost FPGA platforms remains challenging. Floating-point operations involve multiple stages such as exponent calculation, mantissa processing, normalization, rounding, and special case handling, which significantly increase design complexity. Moreover, many reported designs focus mainly on simulation-level validation, with limited emphasis on real-time hardware verification. Practical constraints such as limited input/output resources on FPGA development boards further complicate the implementation and testing of full 32-bit floating-point operations.

To address these challenges, this paper presents the design and implementation of a 32-bit single precision floating-point arithmetic unit compliant with the IEEE-754 standard. The proposed design supports floating-point addition, subtraction, and multiplication, with the multiplication unit implemented using Booth's Radix-4 algorithm to enhance efficiency by reducing partial products. The entire architecture is modeled using Verilog HDL and functionally verified through simulation using the Xilinx Vivado Design Suite. Following verification, the design is synthesized and implemented on a Basys-3 FPGA development board. To accommodate hardware interface limitations, segmented input application using switches and push buttons is adopted, and the 32-bit output is displayed in two 16-bit phases using on-board LEDs.

The proposed work demonstrates a practical and efficient approach to implementing single precision floating-point arithmetic units on FPGA platforms. By combining IEEE-754 compliance, optimized multiplication, and real-time hardware validation, the design offers a balanced solution in terms of performance, accuracy, and feasibility, making it suitable for embedded and real-time digital applications.

II. Literature Review

Floating-point arithmetic has gained significant importance in modern digital systems due to its ability to represent a wide dynamic range of numerical values with higher precision compared to fixed-point arithmetic. As applications such as digital signal processing, image processing, scientific computation, and embedded systems continue to grow, the need for efficient floating-point hardware implementations has increased considerably. Consequently, many researchers have focused on the design and optimization of Floating Point Units (FPUs) to improve performance metrics such as speed, area, power consumption, and accuracy.
Several early implementations of arithmetic units on FPGA platforms were primarily based on fixed-point representations because of their simplicity and reduced hardware complexity. However, fixed-point arithmetic suffers from limited dynamic range and precision, making it unsuitable for applications requiring high numerical accuracy. To overcome these limitations, researchers began exploring IEEE-754 compliant floating-point arithmetic implementations on reconfigurable hardware platforms.

Malkapur and Rajput et al. proposed a generic pipeline-based floating-point arithmetic architecture targeted for DSP processors. Their work focused on implementing both single-precision and double-precision floating-point operations on FPGA using Verilog HDL. The design emphasized area and timing comparison between different precision levels, demonstrating that pipelining improves throughput while maintaining acceptable hardware utilization. However, the increased pipeline stages resulted in higher design complexity.

Sangwan and Angeline et al. presented a pipelined single-precision floating-point co-processor aimed at enhancing operational frequency. Their design employed optimized arithmetic modules and demonstrated improved performance through pipelining. While the approach achieved higher clock frequencies, the architecture required additional hardware resources and careful control logic.

Ushasree et al. developed a high-speed single-precision floating-point unit compliant with the IEEE-754 standard. Their work incorporated pre-normalization and post-normalization stages along with exception handling mechanisms. The design supported addition, subtraction, multiplication, division, and square root operations. Although the throughput was improved, the inclusion of multiple arithmetic operations increased overall circuit complexity.

Considerable research has also been conducted on floating-point multiplication, as it is one of the most time-critical operations in an FPU. Conventional multiplication techniques generate a large number of partial products, leading to increased delay and area. To address this issue, Booth encoding techniques were introduced. Radix-2 Booth multiplication reduced the number of partial products, while Radix-4 Booth multiplication further minimized them by processing multiple bits of the multiplier at a time.

Several researchers reported that Radix-4 Booth multipliers provide an efficient balance between speed improvement and hardware cost, making them suitable for FPGA-based floating-point multipliers.

Yadav and Chaudhary et al. presented a 32-bit floating-point unit for advanced processors using IEEE-754 representation. Their work detailed the implementation of addition, subtraction, and multiplication operations using HDL and analyzed the design in terms of area and delay. The results showed improved delay performance, but hardware validation on development boards was limited.

Grover and Soni et al. implemented a 32-bit floating-point arithmetic unit on FPGA and verified the functionality using simulation tools. Their work highlighted the advantages of FPGA-based floating-point implementations over ASICs in terms of development time and flexibility. However, practical input/output constraints of FPGA boards were not extensively addressed.

From the literature survey, it is evident that significant efforts have been made to improve floating-point arithmetic performance using optimized algorithms, pipelining techniques, and efficient multiplier designs. However, many existing works focus either on simulation-level validation or complex architectures that increase hardware overhead. There remains a need for a practical, IEEE-754 compliant single-precision floating-point arithmetic unit that combines efficient multiplication techniques with real-time FPGA implementation and verification.

Motivated by these observations, the present work focuses on the design and implementation of a 32-bit single-precision floating-point arithmetic unit using Verilog HDL. The design incorporates floating-point addition, subtraction, and multiplication, with multiplication implemented using Booth's Radix-4 algorithm to reduce partial products and improve performance. The complete design is synthesized, simulated, and implemented on the Basys-3 FPGA development board using the Vivado tool, with functional verification performed through test inputs and on-board hardware resources.

III. Problem Statement

The proposed system addresses the limitations of existing floating-point implementations by presenting an IEEE-754 compliant single precision floating-point arithmetic unit optimized for FPGA realization. To improve multiplication efficiency and reduce hardware complexity, Booth's Radix-4 algorithm is employed. The design supports floating-point addition, subtraction, and multiplication using Verilog HDL and is verified through simulation in the Xilinx Vivado environment. Hardware implementation on the Basys-3 FPGA board enables practical validation while ensuring improved performance, accuracy, and resource efficiency.

IV. Methodology

The methodology of the proposed system focuses on the design and implementation of an IEEE-754 compliant 32-bit single precision floating-point arithmetic unit optimized for FPGA realization. The design process consists of three main stages: operand decomposition, arithmetic computation, and result normalization and rounding. Floating-point addition and subtraction are performed through exponent comparison, mantissa alignment, and normalization, while multiplication is implemented using Booth's Radix-4 algorithm to reduce partial products and improve computational efficiency. The complete architecture is modeled using Verilog HDL, verified through functional simulation in the Xilinx Vivado tool, and implemented on the Basys-3 FPGA board for hardware validation.

A. Operand Decomposition

Operand decomposition is the initial stage of the floating-point operation. In this stage, the two 32-bit input operands are unpacked according to the IEEE-754 single precision format. Each operand is divided into three fields: a 1-bit sign, an 8-bit exponent, and a 23-bit mantissa. For normalized numbers, an implicit leading '1' is appended to the mantissa to restore the actual significand.

Fig A : Single Precision Format

This stage prepares the operands for arithmetic processing by separating their components and enabling independent handling of sign, exponent, and mantissa. Proper decomposition is essential to ensure correct exponent comparison, mantissa alignment, and arithmetic accuracy in subsequent stages.

B Arithmetic Computation

Arithmetic computation forms the core of the proposed system and includes floating-point addition, subtraction, and multiplication operations.

B.1 Floating-Point Addition and Subtraction

In floating-point addition and subtraction, the exponents of both operands are first compared to identify the operand with the larger exponent. The mantissa corresponding to the smaller exponent is right-shifted until both exponents match, ensuring proper alignment. Based on the operation type and sign bits, the aligned mantissas are either added or subtracted.

After computation, the intermediate mantissa result may not be in normalized form. Therefore, normalization is required to shift the mantissa and adjust the exponent accordingly. This ensures that the result conforms to IEEE-754 representation before rounding and packing.

B.2 Floating-Point Multiplication Using Booth's Radix-4 Algorithm

Floating-point multiplication is one of the most computation-intensive operations in an FPU. In the proposed system, multiplication is optimized using Booth's Radix-4 algorithm. The sign of the result is obtained by XORing the sign bits of the operands, while the exponents are added and bias corrected.

The mantissa multiplication is performed using Radix-4 Booth encoding, which processes the multiplier bits in groups of two. This technique significantly reduces the number of partial products compared to conventional multiplication methods. As a result, the computation delay and hardware resource usage are reduced, making the design suitable for FPGA implementation.

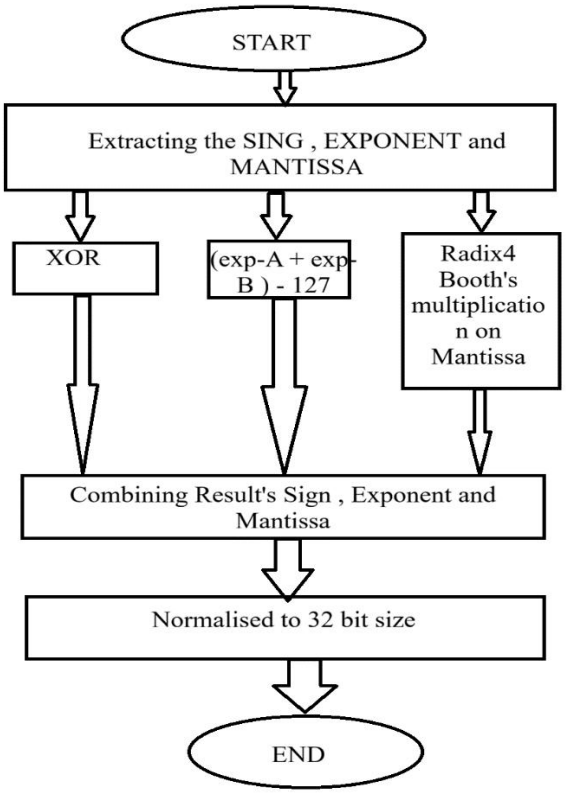

Fig B: Flow chart of Multiplication

The generated partial products are accumulated to produce the final mantissa product, which is then forwarded for normalization and rounding.

Encoding Table (Radix Booth Encoding):

Group of 3 bits (Yi+2Yi+1Yi)	Operation
000 / 111	0 × M (no operation)
001 / 010	+1 × M
011	+2 × M
100	-2 × M
101 / 110	-1 × M

Table B : Booth's Multiplication

B. Result Normalization and Rounding

Result normalization ensures that the mantissa is in standard IEEE-754 normalized form. If the mantissa exceeds the allowable range, it is right-shifted, and the exponent is incremented. If the mantissa is smaller than the required range, left-shifting is performed with appropriate exponent adjustment.

Rounding is applied to fit the mantissa within 23 bits while preserving numerical accuracy. After rounding, the final result is packed by combining the sign bit, adjusted exponent, and mantissa to form a 32-bit IEEE-754 single precision floating-point output.

C. FPGA Implementation and Verification

The complete floating-point arithmetic unit is implemented using Verilog HDL and simulated using the Xilinx Vivado Design Suite. Functional verification is carried out using test benches to validate correctness for various input combinations. The verified design is synthesized and implemented on the Basys-3 FPGA board.

Due to limited on-board input and output resources, operands are applied using switches and push buttons in multiple stages. The 32-bit output is displayed in two 16-bit segments using on-board LEDs, enabling real-time hardware verification of the design.

V16	V17	W16	W15	V15	
0	0	0	0	0	Reset (all input set as zero)
1	0	0	0	0	Input A Lowerbit selected
1	0	0	1	0	Input A Upperbit selection
0	1	0	0	0	Input B Lowerbit selected
0	1	0	1	0	Input B Upperbit selection
0	0	1	0	0	Output of Lowerbits
0	0	1	0	1	Output of Upperbits

Pins set as :-

- V16 & V17 as input selectors
- W15 is mode selector as lower bits or upper bits
- W17 is for output generator
- V15 is for bit selection of output (lower bits or upper bits)

V. RESULTS

This chapter presents the simulation, synthesis, and hardware implementation results of the proposed 32-bit single-precision floating-point arithmetic unit. The performance of the designed FPU is evaluated in terms of functional correctness, resource utilization, and timing characteristics. The obtained results are analyzed to verify compliance with the IEEE 754 standard and to assess the effectiveness of the proposed architecture.

5.1 Functional Simulation Results

Functional simulation is performed to verify the correctness of floating-point addition, subtraction, and multiplication operations. The simulation is carried out using Xilinx simulation tools by applying various test vectors, including normal values, boundary conditions, and special IEEE 754 cases**.**

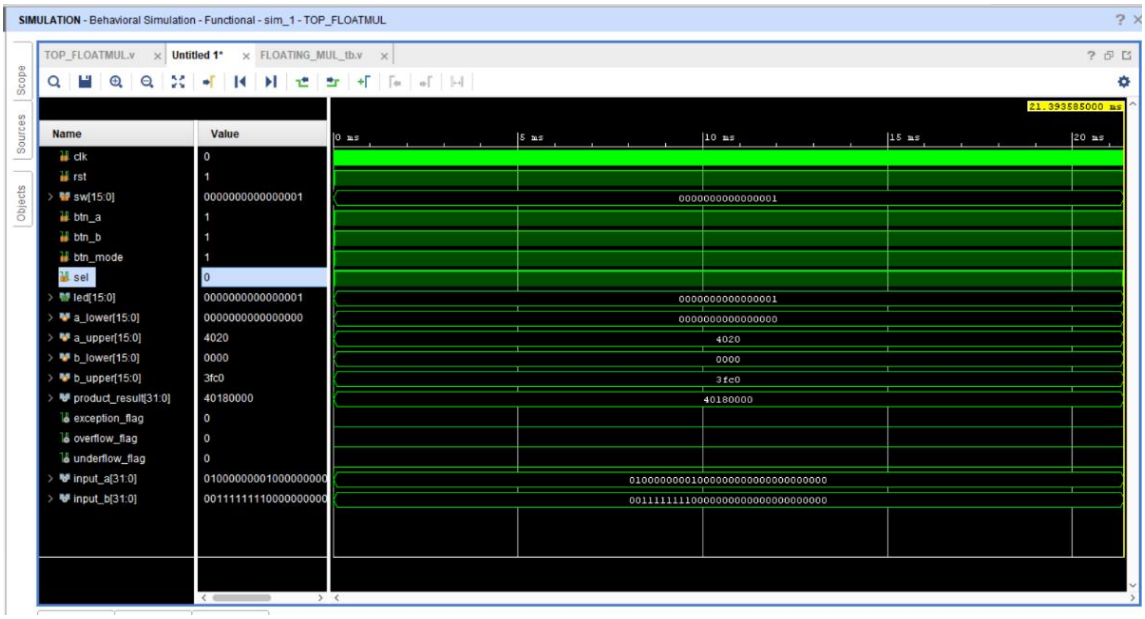

Fig 5.1 : Simulation Results

For floating-point addition and subtraction, operands with different exponents and signs are tested to ensure correct exponent alignment, mantissa computation, normalization, and rounding. The simulation results confirm that the proposed FPU correctly performs arithmetic operations and produces accurate results in accordance with IEEE 754 rules.

For floating-point multiplication, the mantissas are multiplied using the Radix-4 Booth multiplication algorithm. The simulation results demonstrate correct multiplication behaviour with reduced computation steps due to the Booth encoding process.

5.2 Synthesis Results

The proposed floating-point arithmetic unit is synthesized using Xilinx design tools targeting the Basys3 FPGA development board. Post-synthesis reports are analyzed to evaluate hardware resource utilization and timing performance.
The synthesis results indicate that the proposed design efficiently utilizes FPGA resources such as Look-Up Tables (LUTs), flip-flops, and DSP blocks. The use of Radix-4 Booth multiplication reduces the complexity of the multiplication unit, contributing to improved area efficiency compared to conventional multiplication approaches.

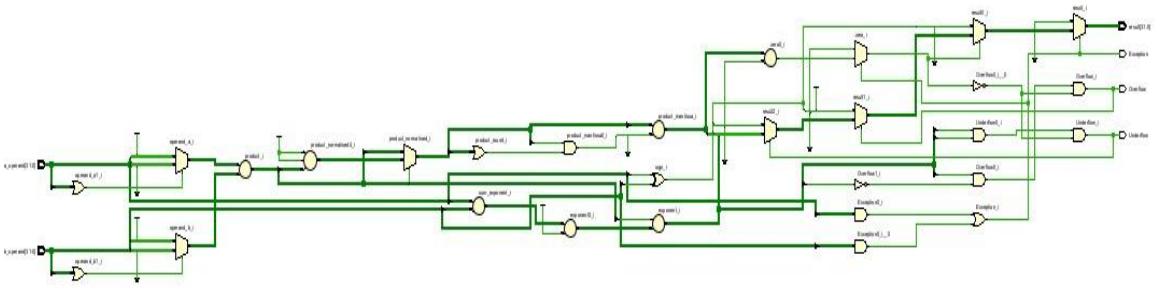

Fig 5.2 : Synthesis Results

5.3 Timing Analysis

Timing analysis is performed to determine the maximum operating frequency and critical path delay of the proposed FPU. The results show that the design meets timing constraints and supports high-speed operation suitable for real-time and computation-intensive applications. The reduced number of partial products in the multiplication unit significantly contributes to improved timing performance.

5.4 Hardware Implementation Results

After successful synthesis and timing verification, the design is implemented on the Basys3 FPGA development board. The implemented FPU is tested using different input combinations, and the outputs are observed to validate real-time functionality. The hardware implementation confirms that the proposed system operates correctly and reliably under practical conditions.

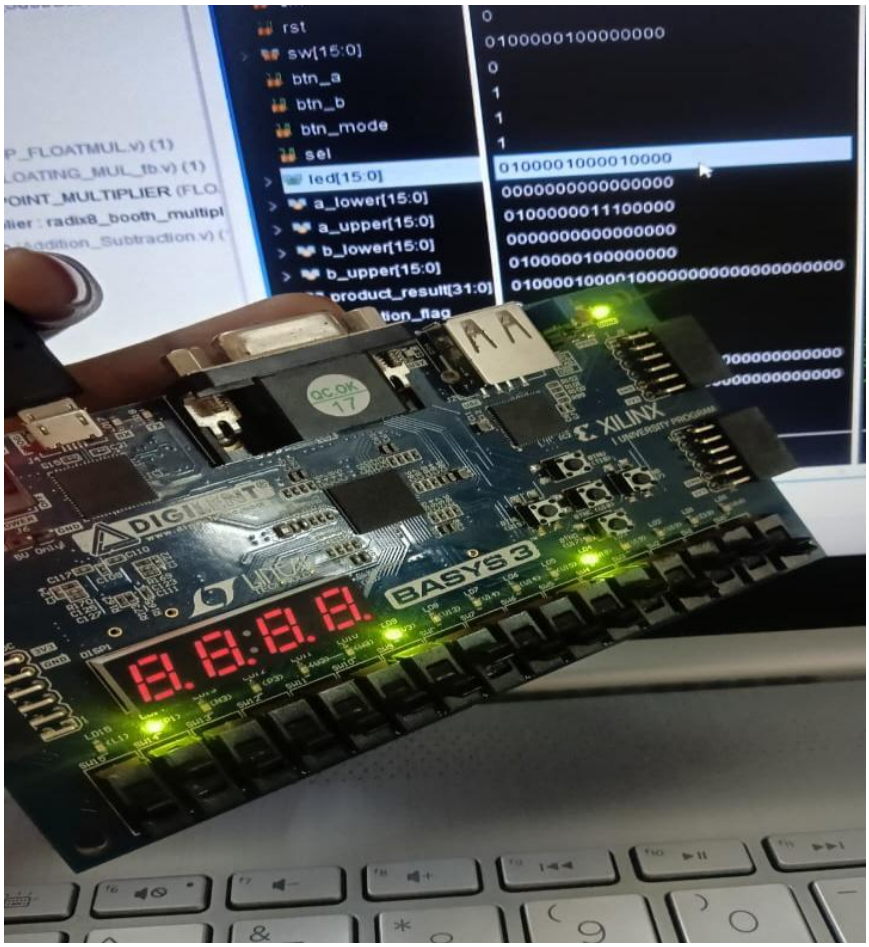

Fig 5.4 : Implementation Results

5.5 Performance Discussion

The overall results demonstrate that the proposed 32-bit floating-point arithmetic unit achieves high accuracy, reduced computational delay, and optimized hardware utilization. The incorporation of the Radix-4 Booth multiplication algorithm improves multiplication speed while maintaining IEEE 754 compliance. Compared to conventional floating-point designs, the proposed system provides a balanced trade-off between performance and hardware complexity, making it suitable for embedded systems, DSP applications, and FPGA-based computing platforms.

VI. CONCLUSION

This work presented the design and FPGA implementation of an IEEE-754 compliant 32-bit single precision floating-point arithmetic unit. The proposed system supports floating-point addition, subtraction, and multiplication, with multiplication optimized using Booth's Radix-4 algorithm to reduce partial products and improve computational efficiency. The complete architecture was modeled using Verilog HDL, verified through functional simulation in the Xilinx Vivado environment, and successfully implemented on the Basys-3 FPGA board.
The results demonstrate that the proposed design achieves accurate floating-point computation while maintaining efficient hardware utilization and practical feasibility under FPGA input/output constraints. The use of Booth's Radix-4 multiplication contributes to reduced computation delay and improved performance compared to conventional multiplication techniques. Overall, the implemented floating-point arithmetic unit provides a reliable and efficient solution suitable for embedded and real-time digital applications.

REFERENCES

1. Dr. Ravindra P. Rajput Srujana B Malkapur "Design of Generic Floating Point Pipeline Based Arithmetic Operation for DSP Processor" IEEE Xplore, 978-1-7281-5374-2/20/$31.00 ©2020 IEEE, pg.no 1059-1064.

2. Manisha Sangwan, An Anita Angeline, Design and Implementation of Single Precision Pipelined Floating Point Co-Processor, 2013 International Conference on Advanced Electronic Systems (ICAES).

3. Ushasree G, R Dhanabal, Dr Sarat Kumar Sahoo, VLSI Implementation of a High Speed Single Precision Floating Point Unit using Verilog, proceedings of 2013 IEEE Conference on Information and Communication Technologies(ICT 2013).

4. F. Mhaboobkhan, K. Kokila, R. Jothikha, and K. L. Preethikha, " Design of Pipelined Parity Preserving Double Precision Reversible Floating Point Multiplier Using 90 nm Technology," 2020 6th Int. Conf. Adv. Comput. Commun. Syst. ICACCS 2020, no. 2, pp. 739-744,2020,doi:10.1109/ICACCS48705.2020.9074209.

5. A. Yadav and I. Chaudhary, "Design of 32 -bit Floating Point Unit for Advanced processors," Int. J. Eng. Res. Appl., vol. 07, no. 06, pp. 39–46, 2017, doi: 10.9790/9622 0706053946.

6. Shanthala. N1, Nayana. M, Chandrashekar.C, Dr. Siva Yella-MP-al-li "Basic operation performed on Arithmetic Logic Unit (ALU) For 32-Bit Floating Point Numbers", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 12 (2017) pp. 3248-3252 Research India Publications.

7. Naresh Grover, M, K Sony, “Design of FPGA based 32-bit Floating Point Arithmetic Unit And verification of its VHDL code using MATLAB”, I.J Information Engineering and Electronics Business, 2014, 1, 1-14 published Online February in MECS.

8. H.H. Saleh, ―H.Fused Floating-Point Arithmetic for DSP, ‖ PhD dissertation, Univ. of Texas, 2008.

9. Swathi.A, G.Srinivasulu “ASIC implementation of a High speed double Precision (64) floating point unit using Verilog”, International journal and magazine of engineering, technology, management and research ISSN 2348-4845.

10. Prashanth B, P.Anil Kumari, G Sreenivasulu,” Design & Implementation of Floating point ALU on a FPGA Processor”, 2012 International Conference on Computing on Computing, Electronics and Electrical Technologies[ICCEET].