ASEAN Engineering Journal

Full Paper

CUSTOM IP DESIGN AND VERIFICATION FOR IEEE754 SINGLE PRECISION FLOATING POINT ARITHMETIC UNIT

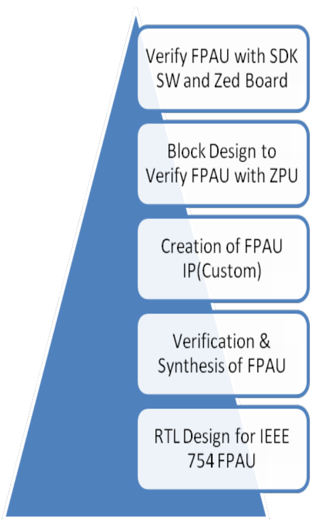
Asha Devi Dharmavaram[a*], Suresh Babu. M[b], Prasad Acharya. G[a].

[a]Sreenidhi Institute of Science and Technology, Ghatkesar, Hyderabad, India
[b]Teegala Krishna Reddy Engineering College, Meerpet, Hyderabad, India

Article history
Received
27 July 2023
Received in revised form
01 November 2023
Accepted
14 November 2023
Published online
31 May 2024

*Corresponding author
ashadevi@sreenidhi.edu.in

Graphical abstract


Verify FPAU with SDK SW and Zed Board
Block Design to Verify FPAU with ZPU
Creation of FPAU IP(Custom)
Verification & Synthesis of FPAU
RTL Design for IEEE 754 FPAU

Abstract

The compact and accurate way of representing numbers in a wide range is the advantage of floating-point (FP) representation and computation. The floating-point digital signal processors offer the IPs that should have the features of low power, high performance, and less area in cost-effective designs. The proposed paper demonstrates the design and implementation of a 32-bit floating-point arithmetic unit (FPAU). The arithmetic operations performed by the FPAU are in the IEEE 754 single precision format for FP numbers. Before performing the 32-bit FP arithmetic operations, the input operands are converted to IEEE 754 single precision. In order to make use of this functional unit in the processor designs, the proposed work discuss about the creation of custom IP. The validation and verification of this IP will be done with the Xilinx Vivado Design software. Here, the verification is performed with VIO hardware debug IP and Zed board. This FPAU IP can be used in DSP applications and can also be used as a floating-point arithmetic block in semi-custom microprocessor and microcontroller designs.

Keywords: Custom IP, FPAU, Hardware Description Language, Single Precision, Zynq Architecture.

1.0 INTRODUCTION

Precision is crucial in today's cutting-edge technologies for applications like digital signal processing. Calculations in engineering and technology use floating-point numbers to represent non-integer values. The IEEE Standard is the most widely used floating point standard. This standard specifies that FP numbers are represented either 32 bits or 64 bits. More applications, including those involving radars, photography, and telecommunications, require floating-point numbers.
This logical approach for the proposed arithmetic operations is to perform them with 32-bit floating-point operands, which are needed in calculation applications. The Verilog hardware description language is used to implement the floating point algorith, which is used to achieve a small area goal. We employ two strategies in Verilog to improve performance. The performance of the circuit is described using the HDL in terms of speed and area. The primary CPU component, the ALU, uses 32 bits to represent floating-point arithmetic operations, logical operations, and other functions.
A technical standard for computing FP numbers was developed by the IEEE and is known as the IEEE 754 standard. It has addressed a number of issues that made various FP implementations challenging to use and less portable. As a result, IEEE Standard 754 is a widely used representation for real numbers on computers

today. This includes PCs and Macs with Intel processors as well as the majority of Unix platforms.

1.1 Literature Survey

The IEEE Standard for Floating-Point Arithmetic [1], 2008, describes formats for binary and decimal FP arithmetic interchange and computation in programming environments. This standard provides the default handling of exceptional circumstances. Implementations of floating-point systems can be done with either software, hardware, or a hybrid usage. The sequence of operations, input data values, and end format are the things the user can control that uniquely define the numerical results and exceptions stated for various operations in this standard's normative section.

P.S. Gollamudi, et al., [2] developed a highly performing 32-bit floating-point adder by using VHDL. They have synthesised them using Xilinx 9.1. They claimed that their system consumed less chip area when compared with other similar systems with a lower combination delay.

Remya Jose and Dhanesh M. S. [3] have developed a statistical analysis for single-precision co-processors. In that work, they designed and implemented a co-processor which can be utilized for analysis. The code was developed in VHDL at RTL level. They used a booth multiplier and a koggestone adder in their work and claimed that it was efficient in terms of delay and area.

At ref. [4], a unique operand-decomposition-based single precision FP- multiplier design has been created by Michael Nachtigal, Himanshu T and others. A reversible revised design for the 8x8-multiplier has also been introduced, which is optimized for latency, garbage output amount, and quantum cost.

Reversible logic gate-based ALU design was discussed by Dhanabal, R., Sarat Kumar Sahoo, and others [5]. TSG is the first reversible gate that functions as a complete adder. This study also presents the creation of a 1-bit alum using a pass transistor and the cadence virtuoso tool. According to the analysis, reversible gates are preferred over irreversible gates in this design.

Niharika et al. [6] proposed the 3-input floating-point adder/subtractor. In this paper, the design was developed using fast adders. Optimized LZA and LZC were used to develop the floating point in this work. The proposed system used an area of 13.89% with a 49.44 ns delay and 91 MW of power dissipation on the Spartan3E FPGA board. They declared that it can be optimised using a fused FP ALU.

Due to the poor precision of fixed-point adders when representing integers, Alaghemand J, Fatemeha, and et.al [7] proposed the concept of a reversible floating-point adder. Conditional exchanges, converters, alignment units, additions, and normalisations are some of the components that make up the proposed design. They tried to improve the quantum costs, output, and fixed input properties of sections before comparing the design with the current design. Compared to existing approaches, their proposed design reduced quantum costs by 78% and 30%, trash emissions by 78% and 26%, and constant inputs by 79% and 30%.

Kahan et al. [8] provided 12 commercially important arithmetic expressions, showing different word sizes, precisions, false positive techniques, and overflow and underflow behavior. Additional formulas have been developed. The cost of creating a "portable" software solution that addresses such numerical variety has risen to intolerable proportions.

Major semiconductor manufacturers embraced IEEE 754 13 years ago, notwithstanding the difficulty it presents to implementers. Hardware designers took on the task with new selflessness in the hope that they might facilitate and promote the enormous growth of numerical software systems. They did achieve significant success. However, misjudging abnormalities that plagued everyone in the 1970s currently only affect CRAY X-MPs and J90s.

Nikhil S.S., et al. (9), in their work, designed a FPGA-based floating-point ALU. They used Xilinx 13.2 for simulation and synthesis purposes. All the logic operations are done with the help of a multiplier. A Vedic multiplier is used instead of a regular multiplier. Using field-programmable gate arrays, the presented design can be used to implement the ALU.

Quinnell et al. [10] offer several new designs for the x87 units of microprocessors' floating-point amalgamate multiplier adders. Modern amalgamate multiply-add units have implementation problems that these new architectures are intended to address while improving performance and lowering power consumption. Each new design is created and implemented using Advanced Devices' ultra-small 65nm dielectric junction transistor technology tools. This work uses a set of modern floating-point arithmetic units as the standard of comparison for ordinal number 14. All designs leverage AMD's "Barcelona" native quad-core standard cell library to develop and highlight new architectures for very modern and realistic industrial technology.

Kodali, Gundabathula S.K, and L. Boppana [11] looked into floating-point math in particular. A frequent computing process is multiplication. Numerous applications exist in science and signal processing. The IEEE-754 single-precision multiplier often calls for a double-precision and the mantissa multiplication of 23 by 23; a big 52 by 52 mantissa multiplier is needed to obtain the outcome. This calculation is a cap on the boundaries of this operation's area and performance. Over the years, multiplication algorithms have been created. In this study, two widely used algorithms are Booth and Karatsuba multipliers, and there is also a comparison of their performances.

The algorithms were developed on an identical FPGA architecture, allowing for a comparison of the number of FPGA resources used and the speed at which they execute. The algorithm that performs the best overall is the recursive Karatsuba.

A.P. Ramesh et al.'s research explores the widespread usage of floating-point multiplication in scientific computations [12]. The high-speed double-precision multiplier is constructed on a Virtex-6 FPGA.

D.A. Devi et al. proposed a 64-bit ALCCU with low-power, high-speed processing that performs code conversion, arithmetic, and logical operations. They have demonstrated a variety of clock frequencies at acceptable low power levels. These constraints were applied to various input and output standards. The proposed system can be used as an IP [13] in high-speed controllers and processors.

D.A. Devi, et al., presented a work on a 32-bit ALU that is planned, simulated, and tested through VIO [14]. The significance of this idea is that, in cases where physical access to I/Os is not possible, we may dynamically test the operation of our design by using the VIO hardware debug IP. It will supply the user inputs to the design and monitor the immediate results, which are processed by the FPGA and can be viewed virtually through the VIO hardware debug window. It is nothing but an emulation result.

D.Asha Devi, et al., developed a power-efficient 32-bit processing unit. For the verification process, virtual I/O was used. It carried out 32 operations, including code conversions and parity generation. With frequencies ranging from 10 MHz to 100 GHz, the power analysis was tested for various Input and outputs [15] and [16].

A double-precision hybrid ALU with high performance that consists of two architectures, the first of which is a semi-FP unit (Semi-FPU) and the second of which is a floating point unit (FPU), has been proposed by S. Ravi [17] in their design. While FPU uses a 64-bit DPU input, semi-FPU uses a 32-bit input, both of them give a 64-bit DPU output. Rounding and exception handling were also produced by their FPU. Verilog HDL was used for the modules' writing. The synthesis is carried out using the Cadence Encounter RTL compiler and 45nm technology, while the simulation is carried out in Xilinx ISE. With a delay of 49735 ps, their design generated 168.44 mW of dynamic power and 168.467 mW of total power.

2.0 METHODOLOGY

The IEEE 754 depicts a single precision number that has a sign bit of 1 bit length, an exponent bias of 8-bit length, and a mantissa, or significant precision, of 23 explicitly stored bits.
The sign bit specifies the sign of the binary number, i.e., 0 for positive and 1 for negative. The exponent field represents both the positive and negative exponents [8]. A bias is added to get the stored exponent. The mantissa consists of significant digits in a floating-point number.

The proposed work was established with 64 bit operating system, windows 10. The CAD tools, Xilinx Vivado version 2018.3 has been used for RTL design of FPAU, synthesis and creation of custom IP. The IP creation will be done after functional verification of the design with RTL simulation. After creation of the custom IP, the IP functionality is verified by integrating it with Zynq processing unit in a block level design. This complex design is implemented on Artix 7 FPGA [20], known as Programmable Logic (PL). To check the FPAU functionality, ARM 9 dual core processor is used and is known as Processing System (PS). These PL and PS are sandwiched on a single chip known as Zynq architecture [21] & [22]. The processing is done with SDK software environment and Zed board. At the SDK environment, the processing logic is developed in C language. The functionality has been verified on SDK terminal.

2.1 A. Conversion of Binary Number into IEEE format

The binary number can be converted into IEEE 754 format with the following steps:

Step 1: Represent the binary number in scientific notation.

Step 2: Make the sign bit (the 31st bit of the IEEE equivalent) 1 if the number is negative and 0 if it is positive.

Step 3: The exponent (30th to 23rd bits of IEEE equivalent) is calculated by subtracting the position of the first one obtained in the algorithm. A bias of 127 is added to the above exponent value.

3.0 IMPLEMENTATION

The floating-point AU is implemented by using the switch-case function. Two select lines are given to the unit, and accordingly, the desired function is executed.
In the proposed design, addition operation will perform with select signal value 00, subtraction operation will perform with 01 select signal value, multiplication operation will perform with 10 value of select signal, and division operation is carried out with 11 value of select signal.

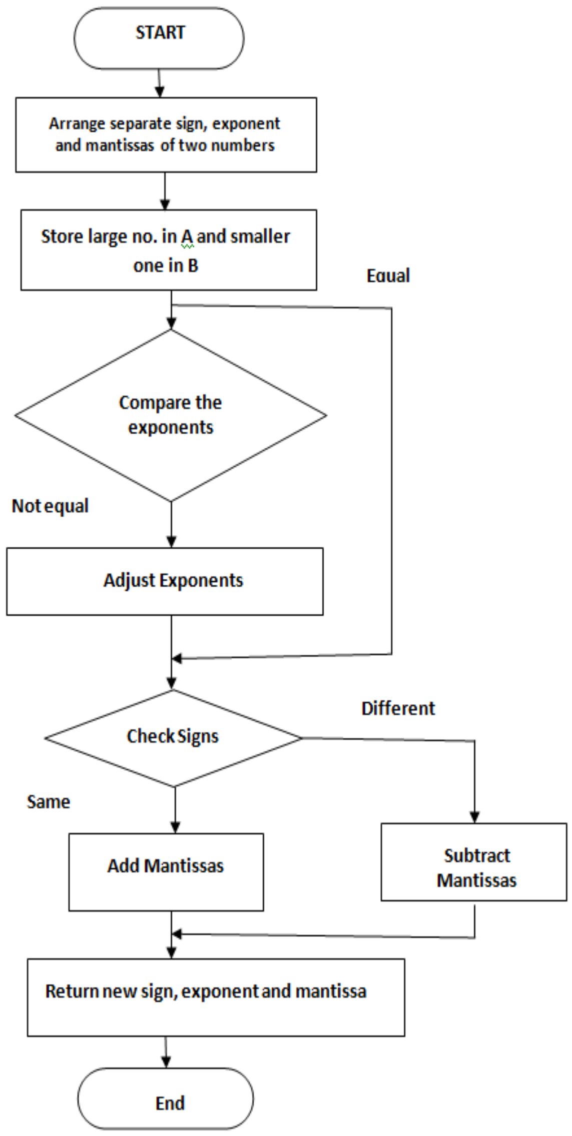

START
Arrange separate sign, exponent and mantissas of two numbers
Store large no. in A and smaller one in B
Equal
Compare the exponents
Not equal
Adjust Exponents
Different
Check Signs
Same
Add Mantissas
Subtract Mantissas
Return new sign, exponent and mantissa
End

Figure 1 Flow chart for floating point IEEE 754 standard addition

3.1 . Addition and Subtraction

Both operands of MSB bits might be either "1" or "0" if they have the same sign. When the operands differ in signs, the MSB of one operand is "1" (positive), and the other is "0" (negative). First, we need to verify the signs of two numbers.

Perform two's complement for the MSB number with "1" if the signs of the two numbers vary. On both numbers, a later addition procedure will be carried out as shown in Figure 1, and the subtraction procedural steps are shown in Figure 2.

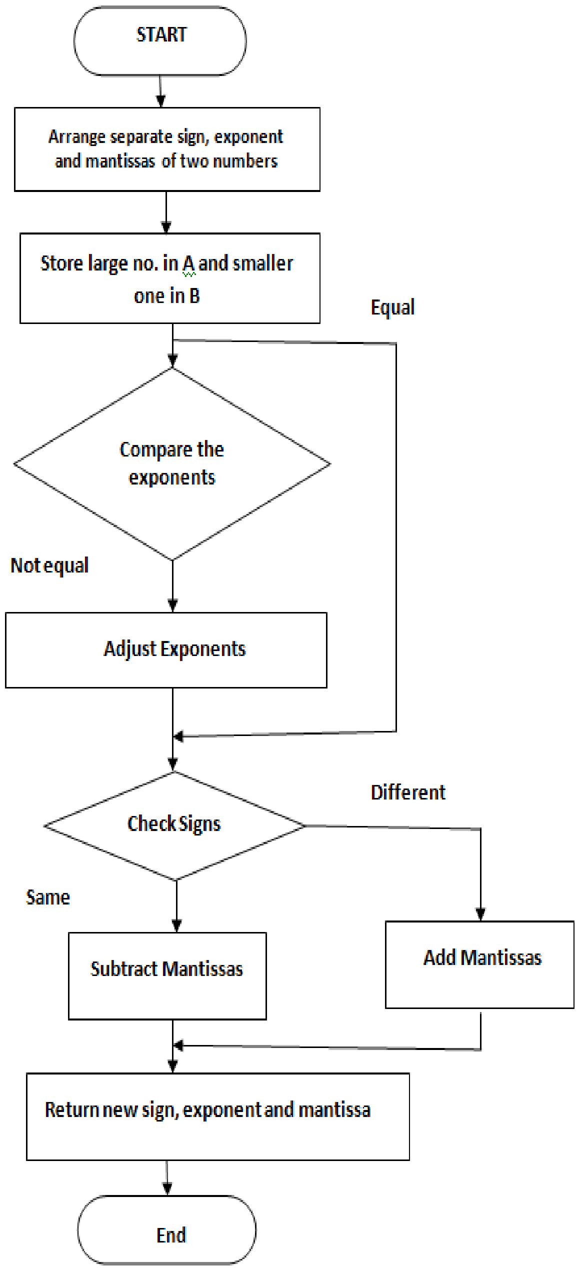

START
Arrange separate sign, exponent and mantissas of two numbers
Store large no. in A and smaller one in B
Equal
Compare the exponents
Not equal
Adjust Exponents
Different
Check Signs
Same
Subtract Mantissas
Add Mantissas
Return new sign, exponent and mantissa
End

Figure 2 Flow chart for floating point IEEE 754 standard Subtraction

This algorithm is used to carry out the addition and subtraction functions in the IEEE 754 standard.

3.2 Multiplication

The two numbers must first be normalised using the IEEE 754 standard. Next, two exponent values are added, and then the bias is subtracted. The sign bit is calculated by performing an XOR operation on both MSB bits. Mantises are multiplied. The mantissa is normalised to provide the desired outcome.

The multiplication of two floating-point numbers can be realized by using the IEEE single precision format, as illustrated in Figure 3.

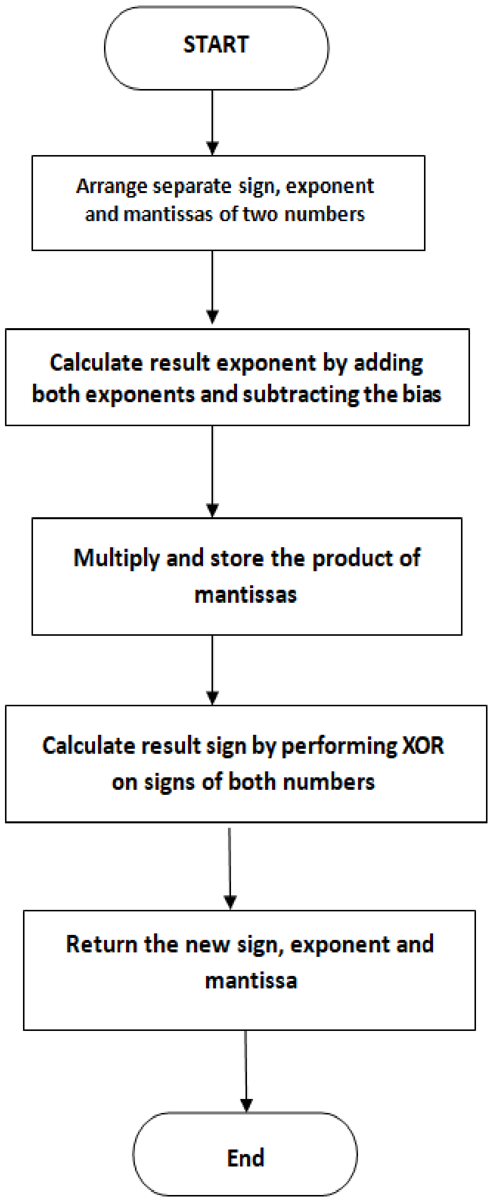

START
Arrange separate sign, exponent and mantissas of two numbers
Calculate result exponent by adding both exponents and subtracting the bias
Multiply and store the product of mantissas
Calculate result sign by performing XOR on signs of both numbers
Return the new sign, exponent and mantissa
End

Figure 3 Flow chart for floating point Multiplication in IEEE 754 standard

3.3 Division

The two numbers must first be normalised using the IEEE 754 standard. Next, two exponent values are subtracted and the bias is added, and the sign bit is calculated by performing an XOR operation on both MSB bits. Mantises are multiplied. The mantissa is normalised to provide the desired outcome. Figure 4 illustrates the floating-point division algorithm.

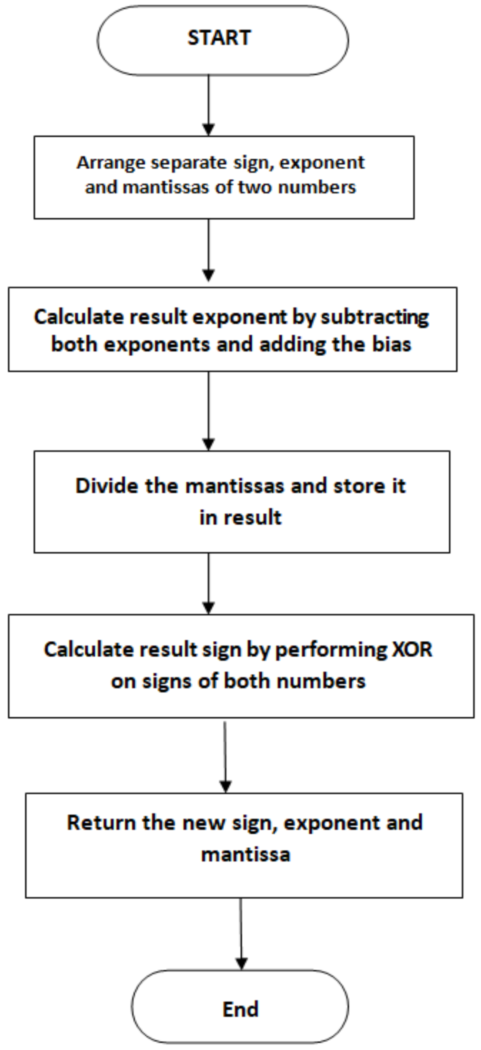

START
Arrange separate sign, exponent and mantissas of two numbers
Calculate result exponent by subtracting both exponents and adding the bias
Divide the mantissas and store it in result
Calculate result sign by performing XOR on signs of both numbers
Return the new sign, exponent and mantissa
End

Figure 4 Flow chart for floating point Division in IEEE 754 standard

3.5 The FPAU Custom-IP Design Steps

Step-i: The FPAU is created in Xilinx Vivado Version 2018.3 using the Verilog HDL, and its functionality is validated through simulation.

Step-ii: Synthesize the design after the simulation process.

Step-iii: Create new IP and packages of IP from the tools available in the Xilinx Vivado software. Then an IP repository will be created.

Step-iv: This created IP can be verified with the block-level design integrated with the Zynq processing unit and AXI interconnect with the custom IP, as shown in Figure 5.

Step v: Prior to completion of the block-level design, verify and generate the output products.

Step-vi: If the validation of the design is complete, generate the HDL wrapper, followed by synthesis, implementation, and bit stream generation.

Step vii: Download the hardware, including the bitstream, and start the SDK software.

Step viii: Create an application project and the necessary application software in the SDK environment. The board support package will be constructed and connected to the Zed board here.

Step-ix: Configure the SDK serial port with a 115200 baud rate, programme the device, and run as a hardware debugger.

Step-x: Now at the SDK terminal, this is available in the tool environment; we can monitor the results of the FPAU with the specified test inputs.

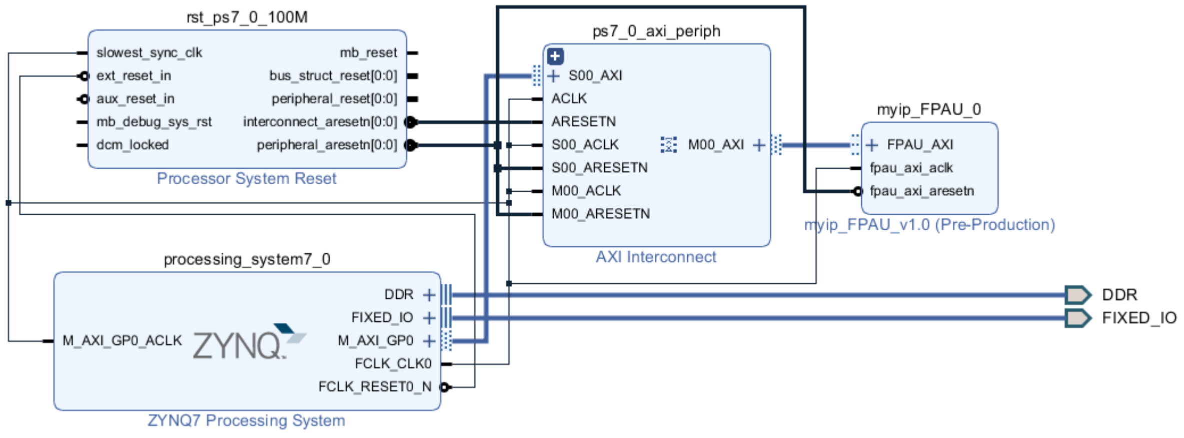


Figure 5 Zynq7 Processing unit and the custom IP : my_FPAU_v1.0

4.0 RESULT ANALYSIS

The functional verification of the FPAU is simulated by writing a test bench module in Verilog HDL, and the waveform results can be observed using the Xsim software tool.

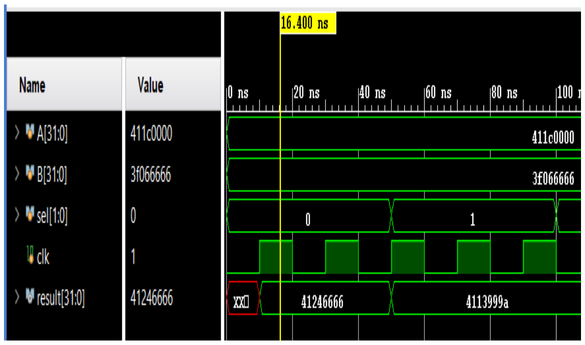


Figure 6 Simulation Result for addition operation

The tested inputs are A = 411c0000 (the hexadecimal equivalent of 9.75 is 411c0000 in the proposed single precision format). and B = 3f066666 (the hexadecimal equivalent of 0.52 is 3f066666).

The FP addition of 9.75 and 0.52 should give the result of 10.27. The hexadecimal equivalent of 10.27 in IEEE 754 32-bit format is 41246666 as illustrated in Figure 6.

The subtraction of 9.75 and 0.52 should give 9.225. By using the IEEE single precision format, the hexadecimal equivalent of the A-B value is 4113999, as illustrated in Figure 7.

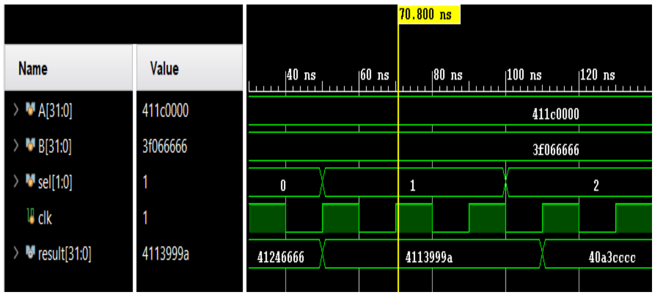


Figure 7 Simulation Result for subtraction operation

The decimal equivalent of 411C0000 is 9.75, and for 3F066666 it is 0.52. In general mathematics, the multiplication of 9.75 and 0.52 should give the result 5.11. By using the IEEE single-precision algorithm, the hexadecimal equivalents of 5.11 are generated as output. The hexadecimal equivalent of 5.11 is 40A3CCCC, as shown in Figure 8.

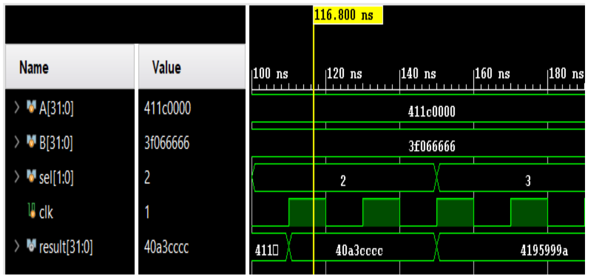


Figure 8 Simulation Results for multiplication operation

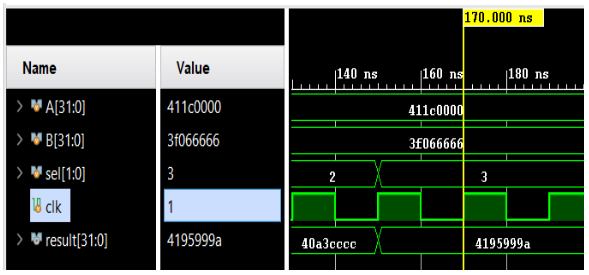
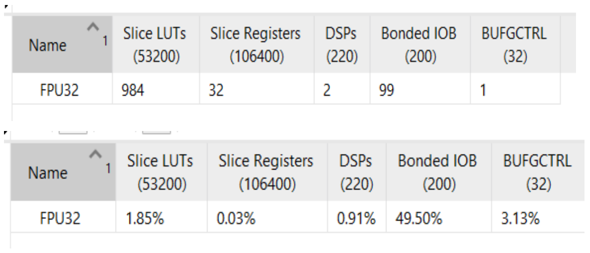


Figure 9 Simulation Result for Division operation

The division of two hexa-decimal numbers, 411C0000 and 3F066666, can be realised by using the division algorithm. The decimal equivalent of 411c0000 is 9.75, and for 3F066666 it is 0.52. Therefore, the division of 9.75 and 0.52 should give the result 18.75. By using the IEEE 754 single precision arithmetic for division, the hexa-decimal equivalent of 18.75 is generated as output, which is 4195999a, as illustrated in Figure 9.

4.1 Utilization Report of 32-Bit FPAU

The utilization report for the 32-bit FPAU is shown in Figure 10. The target used in the work is an Artix7 FPGA. As a result, the proposed design's corresponding slice LUTs, slice registers, DSPs, bound IOBs, and BUFGCTRLs are depicted in Fig. 10 in terms of number and percentage of utilization. The BUFGCTRL is a primitive clock buffer used for glitch-free output [18].



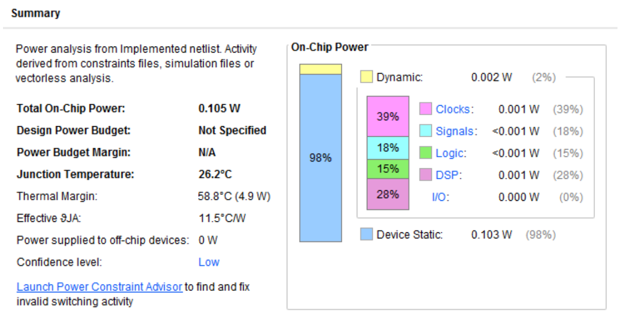
Name 1	Slice LUTs (53200)	Slice Registers (106400)	DSPs (220)	Bonded IOB (200)	BUFGCTRL (32)
FPU32	984	32	2	99	1

Name 1	Slice LUTs (53200)	Slice Registers (106400)	DSPs (220)	Bonded IOB (200)	BUFGCTRL (32)
FPU32	1.85%	0.03%	0.91%	49.50%	3.13%

Figure 10. Utilization Report generated at Implementation Process

4.2 Power Report of 32-Bit FPAU

The power report specified in Figure 11 is generated under a clock constraint of 100 MHz with an IO standard of LVCMOS 33. The total on-chip power is 0.105 W. The device's static power is 0.103 W, and its dynamic power is 2 mW, as shown in Figure 11.



Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.105 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.2°C**
Thermal Margin:	58.8°C (4.9 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

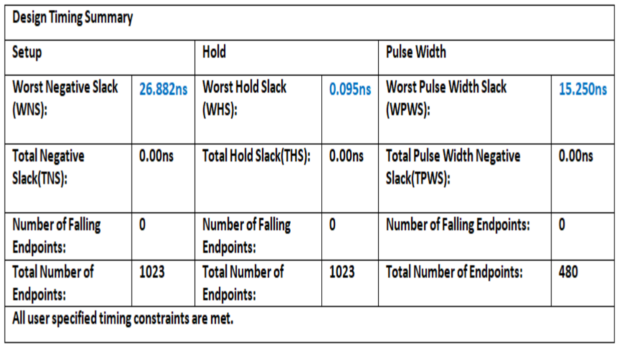
On-Chip Power

Dynamic:	0.002 W	(2%)
Clocks:	0.001 W	(39%)
Signals:	<0.001 W	(18%)
Logic:	<0.001 W	(15%)
DSP:	0.001 W	(28%)
I/O:	0.000 W	(0%)
Device Static:	0.103 W	(98%)

Figure 11 Power Report generated at Implementation Process.

4.3 Timing Report of 32-Bit FPAU

The 32-bit FPAU design timing summary report is shown in Figure 12. The setup time is 26.882 ns, the hold time is 0.095 ns, and the pulse width is 15.250 ns, respectively. All these timing constraints were met without any timing violations.



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	26.882ns	Worst Hold Slack (WHS):	0.095ns	Worst Pulse Width Slack (WPWS):	15.250ns
Total Negative Slack(TNS):	0.00ns	Total Hold Slack(THS):	0.00ns	Total Pulse Width Negative Slack(TPWS):	0.00ns
Number of Falling Endpoints:	0	Number of Falling Endpoints:	0	Number of Falling Endpoints:	0
Total Number of Endpoints:	1023	Total Number of Endpoints:	1023	Total Number of Endpoints:	480

All user specified timing constraints are met.

Figure 12 Power Report generated at Implementation Process

4.4. Comparison Analysis

Some of the earlier researchers have discussed about reversible logic floating point adders of different types. And some researchers have implemented 8-bit, 16-bit, 32-bit and 64-bit adders and multipliers. However, the proposed work covers four arithmetic operations in IEEE754 single precision.
The proposed work is compared with similar IEEE 754 single precision 32 bit operations implemented with other researchers as referred in Ref. [2] and Ref. [6] as illustrated in Table I.

The proposed custom IP design is validated with a Zynq architecture-based development board named the Zed board, with Xilinx Vivado software and SDK platforms. The enhancement of the proposed work over the existing work is that four arithmetic operations are included in a single unit, whereas the existing work did not include all these operations in one unit. The advantage of this custom IP is that it can be used in semi-custom floating-point ALU SoC designs.

The enhancement of the proposed work over the previous work is customized the design of FPAU in 28nm Technology. In the earlier work referred [13],[14],[15] and [16], the design is normal binary 32 bit and 64 bit arithmetic, logic and code conversion operations. All the works are verified with virtual input and output debug IP. However, the floating point arithmetic operations have not done.

Table I Comparison analysis with Ref.2 and Ref.6

S.No.	Parameter	Available	Ref [2]	Ref [6]	Proposed Method
1	Number of Global clks	32	1	2	1
2	Number of bonded IOBs	240	97	99	99
3	Number of LUTs	10944	504	710	984
4	Number of Slice Flip Flops	10944	32	72	32
5	Number of Slices	5472	281	401	246
6	DSPs	-	-	-	2
7	Setup time	-	-	-	26.882ns
8	Hold time	-	-	-	0.095ns
9	Pulse width	-	-	-	15.250ns
10	Power Dissipation	-	-	91mW	105mW
11	Number of operations performed	-	Adder	Adder and Subtractor	Adder, Subtractor, Divisor and Multiplier
12	Methodology	-	IEEE 754 Single precision 32 Bit	IEEE 754 Single precision 32 Bit	IEEE 754 Single precision 32 Bit
13	Custom IP Design	-	-	-	Done with 28nm Technology

The comparison and enhancement of the proposed work with the earlier is illustrated in Table 2.

i. In reference [14], a 32-bit ALU was designed and implemented on a Nexys 4 DDR development board and VIO hardware debug IP.
ii. In the reference [13], 64-bit arithmetic, logic, and code conversions are implemented with the Artix 7 FPGA and VIO hardware debug IP.
iii. In the reference [15], a power-efficient 32-bit ALU with different I/O standards (LVCMOS-12, 15, 25, and 33) and a frequency range of 10 MHz to 8 GHz was verified with an Artix 7 FPGA.
iv. In reference [16], an 8-bit ALU with 16 operations was implemented and verified with CAD tools on both the front end and back end.

In the proposed work, four 32-bit arithmetic operations are implemented with the IEEE 754 standard. In the earlier work, normal arithmetic operations were performed, whereas in the proposed work, FP arithmetic operations are performed. In addition, the enhancement in the proposed work is custom IP design of the FPAU of IEEE 754 32 bit format.

Table 2 Comparison of Ref. 13, 14, 15 and 16 with the proposed work

S.No.	Parameter	Ref[13]	Ref[14]	Ref[15]	Ref[16]	Proposed
1	Size	64-Bit	32-Bit	32-Bit	8-Bit	32-Bit
2	Number and type of operations	Total 32-operations including integer arithmetic, logical and code conversion operations	8-integer arithmetic and 8-logical operations	Total 32-operations including integer arithmetic, logical and code conversion operations	8-integer arithmetic and 8-logical operations	4-Arithmetic Operations With IEEE 754 Single Precision
3	Operating Frequency	100MHz	20GHz	Up to 10GHz	100MHz	100MHz
4	Number of I/Os	201	102	104	45	99
5	Total On chip Power	0.113W	0.085W	0.095W	0.096W	0.105W
6	Timing Constraints	Met	Met	Met	Met	Met
7	Custom IP Design	Not Done	Not Done	Not Done	Not Done	Implemented
8	I/O Standard	LVCMOS12	LVCMOS33	Verified with LVCMOS12 to LVCMOS33	LVCMOS33	LVCMOS33

5.0 CONCLUSION

The IEEE 754 32-Bit (single precision) FPAU custom IP design was successfully designed and verified on Zed Board and Xilinx Vivado System Design 2018.3V software. The utilization, area, and timing reports are generated for the 28 nm technology node. This IP can be used in 32-bit processor, controller, and DSP block designs in semi-custom technology-based designs. The proposed work can be enhanced with beyond-28nm technology and also with higher bits, like 64-bit IEEE 754 (double precision) FP designs. The future scope of the work can be integrated with integer and FPAU operations in one processing design unit.

Acknowledgement

The authors of this paper acknowledge the management of Sreenidhi Institute of Science and Technology for providing the necessary support and encouragement in doing the research and publication of this paper.

References

[1] “IEEE Standard for Floating-Point Arithmetic”,2008. in IEEE STD 754- 1-70, Aug. 292008.

[2] Preethi Sudha Gollamudi, M. Kamaraju, 2013, Design of High Performance IEEE- 754 Single Precision (32 bit) Floating Point Adder Using VHDL, *International Journal Of Engineering Research & Technology (IJERT)* 2(7): 2264-2275

[3] Remya Jose , Dhanesh M S, 2015. Single Precision Floating Point Co-Processor for Statistical Analysis, *International Journal Of Engineering Research & Technology (IJERT) NCETET* –3(5): 1-4

[4] Nachtigal, Michael, Himanshu Thapliyal, and Nagarajan Ranganathan. 2010 "Design of a reversible single precision floating point multiplier based on operand decomposition." In *Nanotechnology (IEEE-NANO), 10th IEEE Conference* on, 233-237. IEEE, 2010.

[5] Dhanabal, R., Sarat Kumar Sahoo, V. Bharathi, V. Bhavya, Patil Ashwini Chandrakant, and K. Sarannya. 2016. "Design of Reversible Logic Based ALU." In *Proceedings of the International Conference on Soft Computing Systems*, pp. 303-313. Springer India.

[6] A. Niharika, G. Naresh, Neelima K, 2021, Design of Three-Input Floating Point Adder/Subtractor, International Journal Of Engineering Research & Technology (IJERT) ICACT – 2021. 9(8): 51-53.

[7] Alaghemand, Fatemeh, and Majid Haghparast. 2015"Designing and Improvement of a New Reversible Floating Point Adder."’

[8] Kahan,William. 1996. "IEEEstandard754 for binary floating point arithmetic" Lecture Notes on the Status of IEEE 754.94720-1776:11.

[9] Nikhil S. S , Sheela Devi Aswathy Chandran , , 2014, FPGA based Floating Point Arithmetic and Logic unit (ALU), International Journal Of Engineering Research & Technology, Vol. 2, Issue 08, pp-52-57, 2014.

[10] Quinnell, Eric, Earl E. Swartzlander Jr, and Carl Lemonds. 2007. "Floating-point fused multiply-add architectures." Signals, Systems and Computers, 2007 ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on. IEEE,

[11] Kodali, R.K.; Gundabathula, S.K.; Boppana, L., 2014 "FPGA implementation of IEEE-754 floating point Karatsuba multiplier," Control, Instrumentation, Communication and Computational Technologies (ICCICCT), International Conference on 10-11 July 2014.300-304.

[12] Ramesh, A.P.; Tilak, A.V.N.; Prasad, A.M., 2013."An FPGA based high speed IEEE-754 double precision floating point multiplier using Verilog," Emerging Trends in VLSI, Embedded System, *Nano Electronics and Telecommunication System (ICEVENT), 2013 International Conference* on7-9 Jan pp 5.

[13] Dharmavaram Asha Devi, M. Suresh Babu, 2019. “Design and Analysis of Power Efficient 64-Bit ALCCU”, *International Journal of Recent Technology and Engineering (IJRTE*), 8(2): 162-166. ISSN: 2277-3878. DOI: 10.35940/ijrte.A1993.078219,

[14] D. A. Devi and L. S. Sugun, 2018. "Design, implementation and verification of 32-Bit ALU with VIO," *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, Coimbatore, India, 495-499, doi: 10.1109/ICISC.2018.8399122.

[15] Dharmavaram Asha Devi, Sandeep Chintala, Sai Sugun L, 2018. “Design of Power Efficient 32-Bit Processing Unit” *International Journal of Engineering & Technology,* 7 (2.16): 52-56

[16] Dharmavaram Asha Devi, 2016. “FPGA Design Flow for 8-bit ALU using Xilinx ISE,” *International Journal of Modern Electronics and Communication Engineering (IJMECE)*, 4(2): 1-4. ISSN: 2321-2152, -

[17] S. Ravi, Adig and H. M. Kittur, &quot; 2017. Design of high performance double precision hybrid ALU for SoC applications,&quot; 2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS), Vellore, India, 1-6, doi:10.1109/ICMDCS.2017.8211610.

[18] “Versal Adaptive SoC Clocking Resources” Architecture Manual AM003 (v1.5) May 16, 2023. url: https://docs.xilinx.com/r/en-US/am003-versal-clocking-resources/Clock-Buffer-Resources.

[19] Roberto R. Osorio et. All, 2023, “Floating Point Calculation of the Cube Function on FPGAs”, *IEEE Transactions on Parallel and Distributed Systems,* 34: 372-382. DOI: 10.1109/TPDS.2022.3220039

[20] Dharmavaram Asha Devi, Niharika Reddy Kathula, Gopinath Kalluri, and Leela Sai Bondalapati, 2023. "Design and Implementation of Image Processing Application with Zynq SoC", *International Journal of Computing and Digital Systems*, 14(01): 377-385.

[21] Devi, D.A., Savithri, T.S., Babu, M.S. 2021. Monitoring and Controlling of ICU Environmental Status with WiFi Network Implementation on Zynq SoC. In: Suma, V., Chen, J.IZ., Baig, Z., Wang, H. (eds) Inventive Systems and Control. Lecture Notes in Networks and Systems, 204. Springer, Singapore. https://doi.org/10.1007/978-981-16-1395-1_48

[22] M. Veldurthi and A. D. Dharmavaram, 2022. "Automatic Vehicle Identification and Recognition using CNN Implemented on PYNQ Board," *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, Coimbatore, India. 1302-1306, doi: 10.1109/ICECA55336.2022.10009054.