DESIGN OF DOUBLE PRECISION FLOATING-POINT ARITHMETIC UNIT FOR SPACE SIGNAL APPLICATIONS

DR. V. ANANDI [A]
[a] Associate Professor of ECE, M S Ramaiah Institute of Technology (Autonomous), anandi.v@msrit.edu,Bangalore
DR. M. RAMESH [B]
[b] Professor, School of Management Presidency Universityramandsur@gmail.com., Bangalore

Ponduri Sivaprasad, Research Scholar, Visvesvaraya Technological University, Belgaum

ABSTRACT

Detection of accurate phase estimation, functionality, reliability, accuracy and higher performance of the system are major concerns in space signal processing systems. The data is acquired at high speed and supervised continuously for the validation of correct data. In space applications, milli degree estimation is s major challenge. Fixed point arithmetic operations and single-precision floating-point operations have huge data losses. All existing double-precision floating-point arithmetic operations utilize dual-rail coding to perform complete detections and also require the circuit to receive acknowledgment on completion of execution. It leads to a worst-case delay irrespective of the actual completion time. With the help of modified double-precision floating-point operations, using memory-based synchronization architecture we can obtain more reliability and highly accurate phase detection. In order to estimate milli degree, the Double Precision Floating Point (DPFP) based arithmetic operations are designed using Verilog hardware description language and synthesized with help of Xilinx Design Suite, and finally implemented on Virtex-5 FPGA development board. All arithmetic operations use ternary logic at lower-level modules to optimize area and latency. The synthesized results show that the proposed floating-point design on the Arithmetic Logic Unit for estimation of milli-degree reduces the delay by 23%, throughput is improved by 13% and power consumption is reduced to 31% as compared to existing works.

Keywords: Space signal, Floating-point, milli degree, half unit, double-precision,

I. INTRODUCTION

The performance and dependability of space-based systems are critical. The primary function of current systems is to transport datarather than doing error checks, and the data is sent to the Earth for error checking. To ensure the authenticity of obtained data, correct system functionality must often be regularly monitored. The data capture system captures signal with Gigasamples per second (GSamp/s). In space-borne systems, phase shift data can be utilized to either confirm correct system performance or highlight potential problems. These phase shifts can be calculated for a single channel input vs a known signal model, as well as for two channels with data collected simultaneously. Microwave signals will be used to monitor the Earth's environment by a number of planned NASA remote sensing satellite projects. The National Research Council's decadal assessment identified three such missions and their importance. While each of these missions will employ signal phase in a different way, the fact that their research products rely on it emphasizes the significance of accurate signal phase measurement. The use of the signal phase to characterize the data obtained is a typical feature of these missions. Phase calculations with a maximum inaccuracy of 5%0.021 degrees (0.006% of 360 degrees) are calculated using our adaptive algorithm.

II. LITERATURE REVIEW

Floating-point implementation on a Field Programmable Gate Array (FPGA) is a newer field that has seen significant achievements because FPGA development takes less time and costs less than ASIC design. This study's main purpose is to look at the area and timing of single and double-precision floating-point units (FPUs) and MAC units. On the Spartan 6 FPGA, the described model is both simulated and implemented [1].This research article introduces hardware topologies based on FPGA for floating-point (FP) multipliers. Single-precision (SP), double-precision (DP), double-extended precision (DEP), and quadruple precision (QP) can all be used with the multiplier designs supplied..A variation to this pipeline division technique was presented in [4] to drastically reduce the LUT size. When compared to [9], this technique reduces the LUT size from 13 KB to 208 B for single-precision and from 470 MB to 56 KB for double-precision.In contrast to [4,] [5] suggests a further adjustment for double-precision floating-point values that greatly decreases chip space, particularly by reducing the LUT size from 56 KB to 2.5 KB.This effective pipeline divider technique is often used to reduce power consumption and latency in multi-media and signal processing applications, as well as quick processing [7]. Instructions in [8] have a latency at the cost of the improved architecture. [9] describes a Taylor series expansion-based high-radix pipeline division algorithm. Despite having a smaller lookup table (LUT) than other techniques, this algorithm's table was still somewhat huge. An array multiplier and a Booth multiplier were used to create the Finite Impulse Response Filter (FIR), and the results were compared for various constraints. It also utilizes the first six terms of Taylor series expansion for the approximation to reduce the size of the LUT [10].

III. METHODS AND ALGORITHM

i. Half Unit 64-bit based (HUB) based double precision 64 bit floating-point addition and subtraction

The comparator, extra shifter, and bit inverter, which were identified as sources of bias are all eliminated in Fig.1. As a result, the operation can have a positive or negative outcome. A leading zero-one detector (LZOD) and a final conditional inverter are required in the new design to detect if the result is negative and to calculate the magnitude of the result if it is negative. The conditional inverter after the swap module in Fig. 1 and the ILSB at the R-input Shifter is responsible for acquiring the two's complement of the right operand when an effective subtraction is required. When an effective subtraction is performed, the conditional upper inverter conducts a bitwise inversion of the significand's MSB. The value ILSB=1 is concatenated at the R-input shifter to produce the right operand to the adder (m + 2 bits), which is two's complement in subtraction. In the case of aligned addition (d=0, Eop=0), the hardware required is the same as in adder A+. The "unbiased" logic box is shown in Figure 4. This "unbiased" box also includes the logic needed to deal with the tie condition that occurs when executing a subtraction with d=1 (case d=1, Eop=1). Consider 5-bit significands as shown in Fig.1 by assuming two HUB numbers A and B with two operands x and y with mantissa M_a = 01.1010 (1) and M_b =01.0100 (1), with the sign bit also included and exponent of both are 0. For subtraction, perform A+(-B) and the results as per normal subtraction is 01.10000 with exp-2

As per the HUB concept, there are two ways to perform floating addition and subtraction such as

A. M_a+(-M_b) as per Fig.1(a)

The detailed explanation for M_a+(-M_b) with example is elaborated below

a. Let consider Ma-M_b and M_a=M_b conditions along with S_a =S_b=0 (sign bit) and Cop (Control operation) is to perform either addition or subtraction with single algorithm as per Fig.1(a) and Fig.1(b).

b. If Cop=0 for addition and Cop=1 for subtraction, here M_a-M_b is called left input of 01.1010(1) and M_a=M_b is called right input whose value is 10.1011(1), and these two inputs added through 2's complement, results in 00.0110(0) shown in Fig,1(a).

c. The output of L-Shifter shown in Fig.1(a) is 01.1000(0) exp-2 (here exact value, sign bits are included) and it is found that result is positive so the inverter condition module does not alter any value so the final results are 01.1000(1) exp-2, the sign bit is 0 and it has 0.00001 exp-2 positive bias.

d. As per the above discussion, the mathematical calculations of M_a+(-M_b) are validated through subtraction as shown below.

```
   01.1010(1)
- 01.0100(1)
-------------
   00.0110(0)
```

This result is left-shifted and the output is 01.1000(0), after rounded up of 01.1000(0), the final result is 01.1000(1)

B. -M_b +M_a as per Fig.1(b)

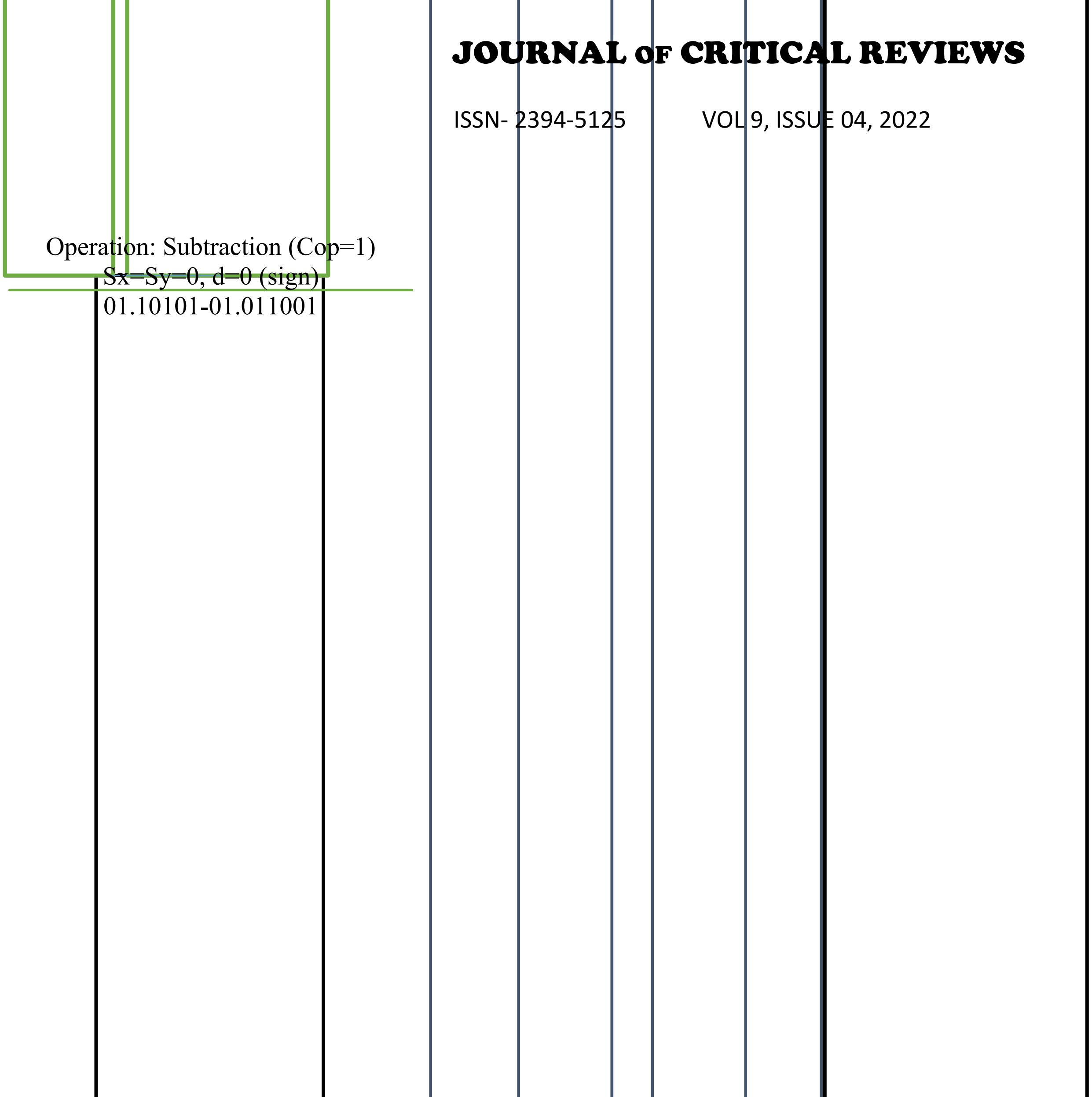
a. To perform - M_b + M_a, consider M_x- M_a as left input and M_y= M_b as right input and S_x =1 and S_y=0 (sign bit) and Cop=0 then both inputs are added as 2's compliments i.e 01.0100(1) and 10.0101(1) then result is 11.1010(0) as shown in Fig.1(b).

b. After the left shifter, the result is 10.1000(0) exp-2 inclusive of sign bit. The result is found to be negative and hence inverter condition module inverts the bits and its output is01.0111(1) exp-2, sign=0, it involves a negative bias of 0.00001.

c. The mathematical validation of the same is shown below:

```
   - Mb + Ma
-01.0100(1)
  +01.1010(1)
-------------
11.1010(0)
```

After left-shifted, the result is 10.1000(0) and after rounding up, it is 01.0111(1). Finally, depending on the operands' positions, rounding up or down is performed.



(a) HUB based Subtraction (M_a+(-M_b)) (b)HUB based subtraction (-M_b +M_a)

Figure .1 Proposed architecture of HUB based 64 bit double precision floating point subtraction

i. The direction of the final rounding (up or down) is not known as the position of the operands is known a priori and therefore the result is not biased (assuming the same chance of having operations (M_a+(-M_b)) and (-M_b +M_a). On the other hand, in the unusual event, if the input pattern ahead of time is known apriori, the swap module of Fig. 2 can provide an alternative option.

ii. When d=0 and subtraction is done, this module can swap the inputs based on the value of the input LSBs, obtaining the necessary unbiased rounding. Because the operation (Ma+(- Mb)) is always done with the biggest absolute number of the left input of the adder due to the comparator, this case of bias cannot be avoided in Fig.2.

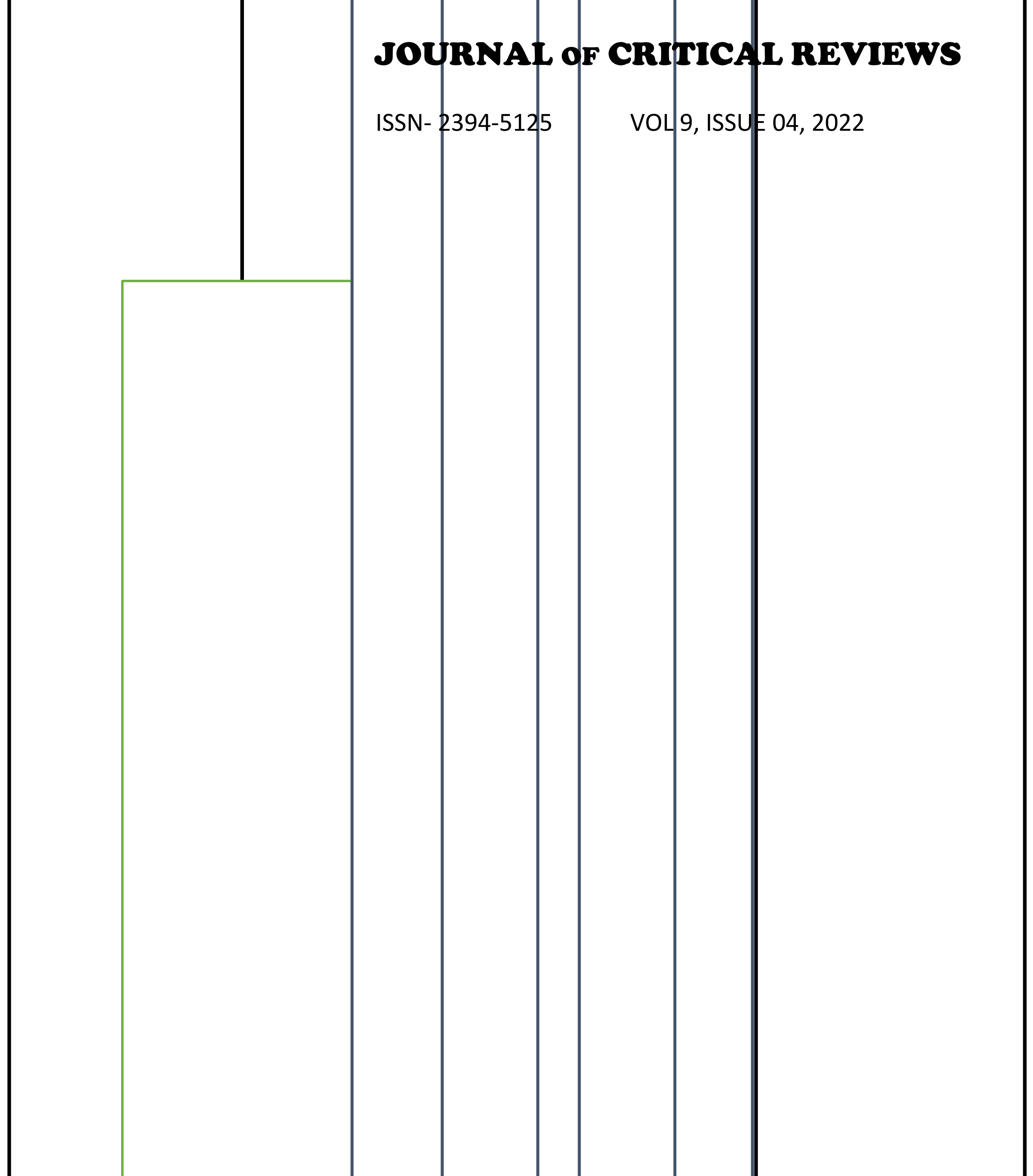


Figure.2: The proposed architecture of HUB based 64-bit double-precision floating-point adder

ii. Double-precision 64 bits floating point Division

The reciprocal generator, as shown in Fig. 3, has three integer multipliers, two of which are responsible for generating a reciprocal after many repetitions. To accomplish the rapid calculations inside the reciprocal generator, the Karatsuba multiplier is used instead of the commonly used integer multiplier. In this multiplier, the partial products are totaled in stages of the half and full adders. Then, standard procedures are used to aggregate the data. A 16-bit custom floating-point format is presented as a comparison to the existing 16-bit brain floating-point format. In addition, a 24-bit unique floating-point format is provided to validate performance comparisons with various floating-point formats. The above format for accumulation is used to improve the accuracy of a 16-bit convolution block.

In existing research works, many algorithms are proposed for accurate division operations, out of which, Newton Raphson algorithm is widely used in many applications. This algorithm requires only one multiplier and subtractor for the calculation of inverse(1/B) of number and its calculations as per equation (1).

$$x_{n+1}=x_n-\frac{f(x_n)}{f^1_{(x_n)}}\text{---------------------}(1)$$

Where $f^1_{(x_n)}$ is derivative of $f(x_n)$ and this equation is recursive with respect to y and the number of iterations are performed to get reciprocal values as expected results. The negative numbers are not allowed due to the nature of the recursive process. As the reciprocal method for divider operation requires many multipliers and iterations it has excessive delay and area overhead. In order to optimize these limitations, the proposed double-precision floating-point divider uses a signed array with one multiplier and minimum delay. It uses the Array-based Processing Unit (APU) as shown in Fig.3 and selects quotient using subtraction module and carry-out by feedback module.

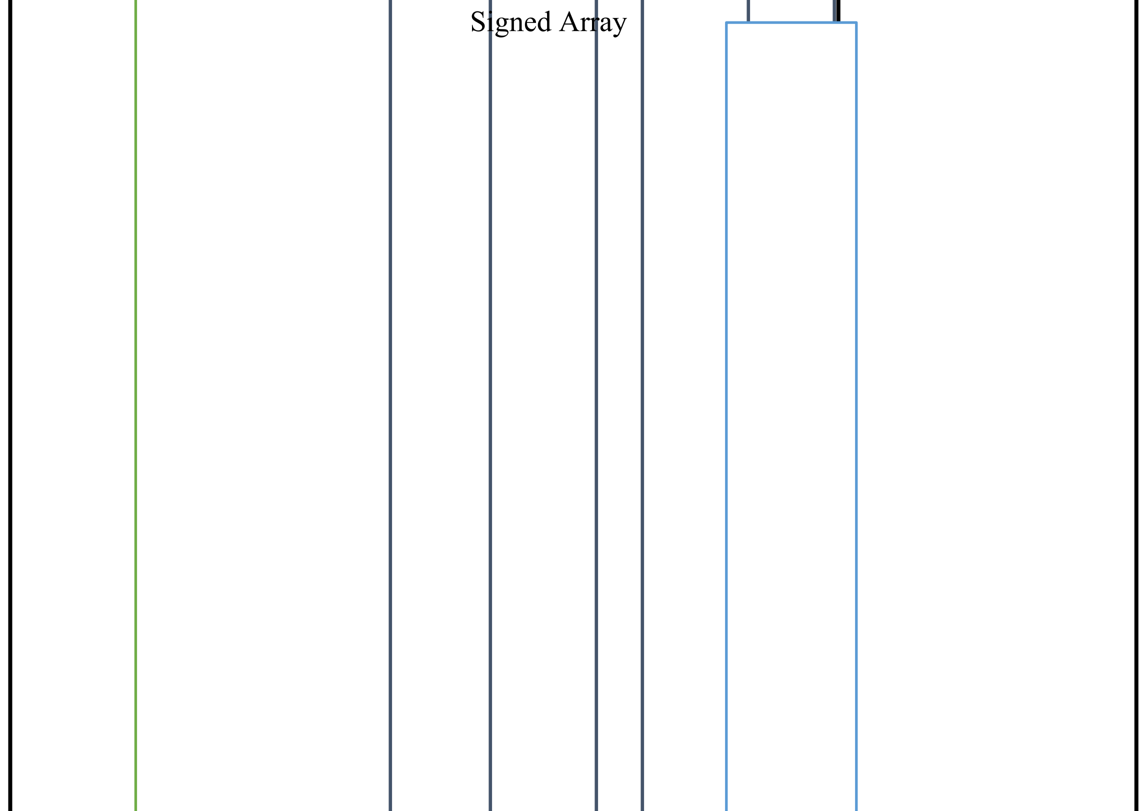
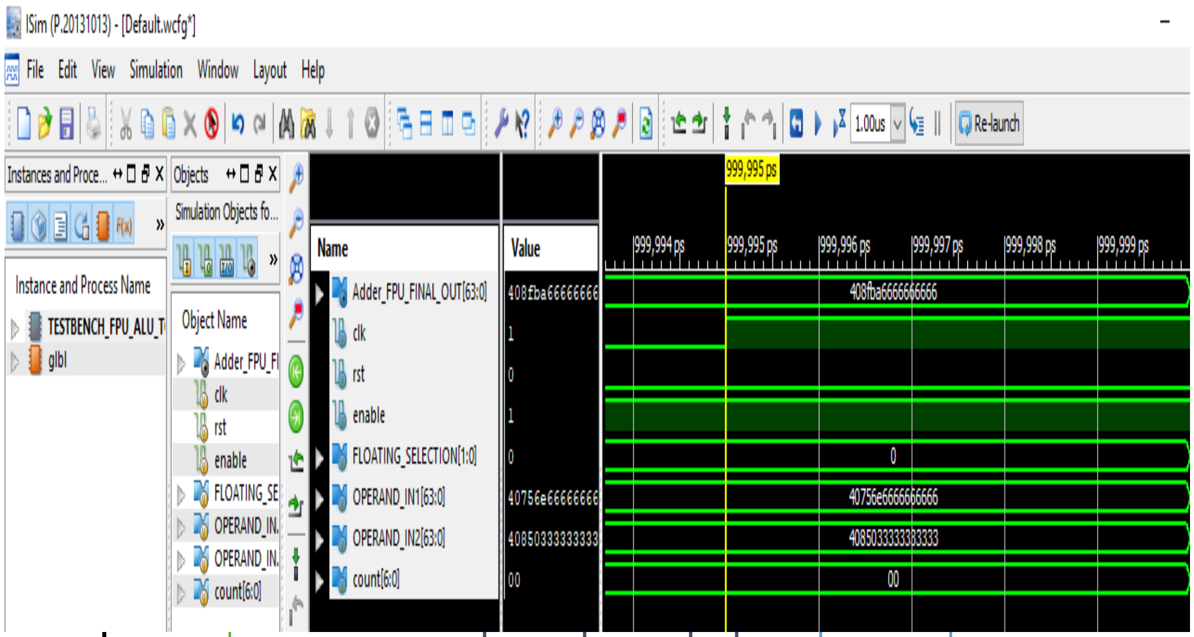


Figure.3. The proposed internal architecture of divider 64-bit double-precision floating-point Divider

IV. RESULTS AND DISCUSSION

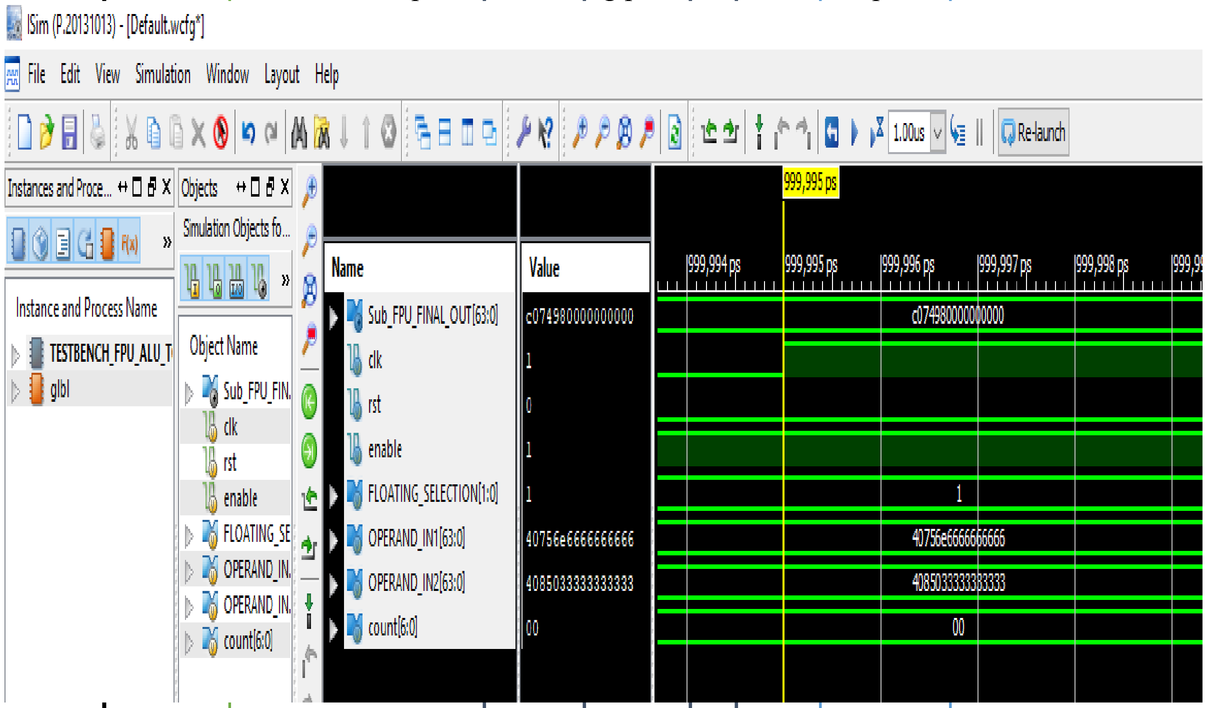
The Xilinx Design Suite Design Compiler is utilized to synthesize Verilog HDL designs for the proposed double-precision arithmetic operations. The Virtex-5 FPGA development board, which is based on 32nm manufacturing technology and has a fixed frequency of 100 Mhz, was chosen for our evaluation. Table 1 compares the synthesis results of floating-point adders/subtractors with varied bit widths. As the number of bits required lowers, the amount of area or energy used by a floating-point adder/subtractor operation reduces. The floating-point format for each bit width is N (S, E, M), where N is the total number of bits, S is a sign bit, E is an exponent, and M is a Mantissa.



Name	Value
Adder_FPU_FINAL_OUT[63:0]	408fba66666666
clk	1
rst	0
enable	1
FLOATING_SELECTION[1:0]	0
OPERAND_IN1[63:0]	40756e66666666
OPERAND_IN2[63:0]	40850333333333
count[6:0]	00

```
408fba6666666666
0
40756e6666666666
4085033333333333
00
```

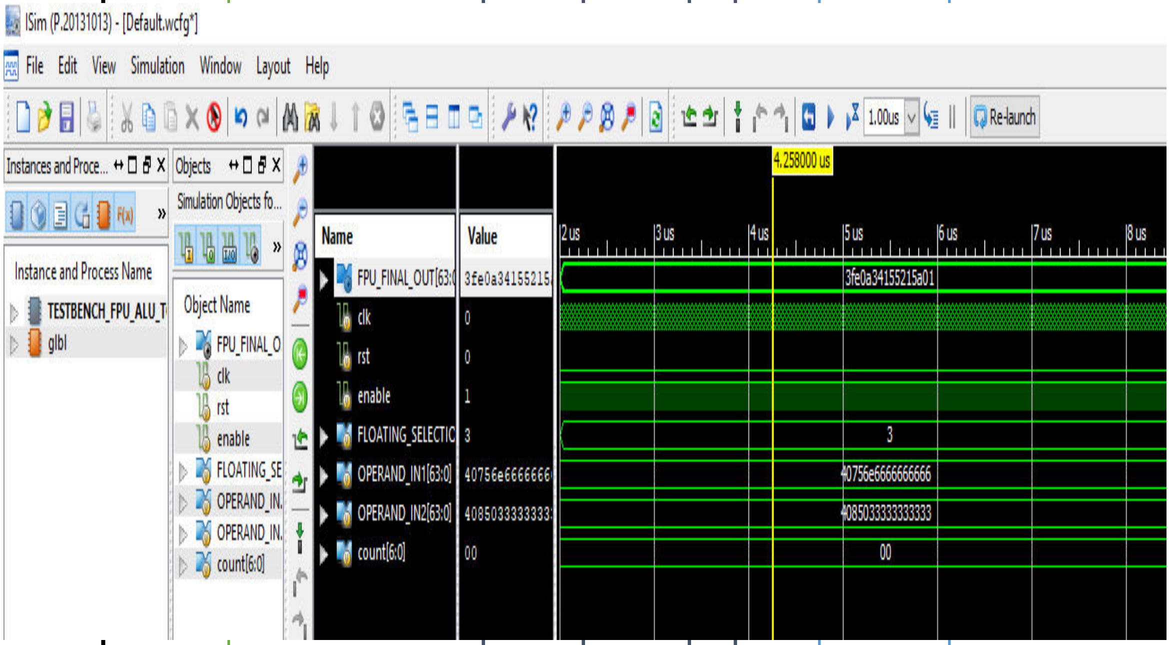
Figure.4. Simulated results of Double precision floating point adder for two operands



Name	Value
Sub_FPU_FINAL_OUT[63:0]	c074980000000000
clk	1
rst	0
enable	1
FLOATING_SELECTION[1:0]	1
OPERAND_IN1[63:0]	40756e6666666666
OPERAND_IN2[63:0]	4085033333333333
count[6:0]	00

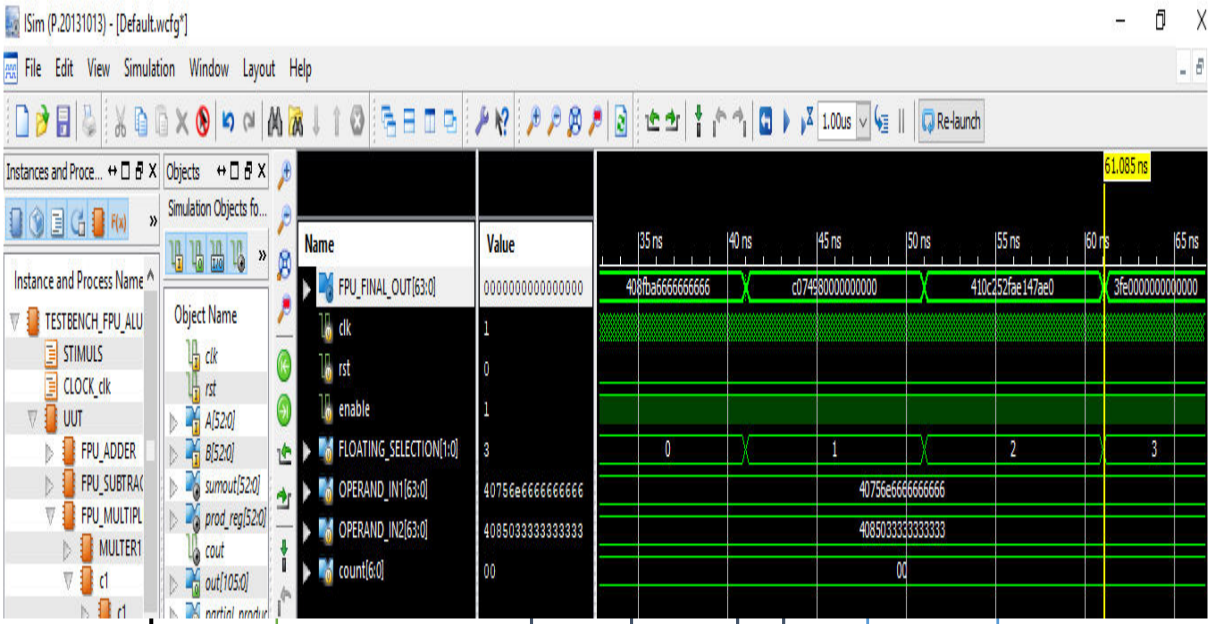
```
c074980000000000
1
40756e6666666666
4085033333333333
00
```

Figure.5. Simulated results of Double precision floating point for subtraction for two operand values



```
Name                    Value
FPU_FINAL_OUT[63:0]     3fe0a34155215a01
clk                     0
rst                     0
enable                  1
FLOATING_SELECTION      3
OPERAND_IN1[63:0]       40756e6666666666
OPERAND_IN2[63:0]       4085033333333333
count[6:0]              00
```

Figure 6.. Simulated results of Double precision floating point for divider



```
Name                    Value               Waveform
FPU_FINAL_OUT[63:0]     0000000000000000    408fba6666666666 | c07498000000000 | 410c252fae147ae0 | 3fe000000000000
clk                     1
rst                     0
enable                  1
FLOATING_SELECTION[1:0] 3                   0 | 1 | 2 | 3
OPERAND_IN1[63:0]       40756e6666666666
OPERAND_IN2[63:0]       4085033333333333
count[6:0]              00
```

Figure.7. Simulated results of Double precision floating point for ALU for two operand values

Finally, Table 1 compares division operators in a variety of floating-point formats. Despite the fact that the operation delay time remains constant when compared to other operators, the smaller the number of bits employed for the floating-point division operation, the lower the area or energy consumption. Many test cases have been run to determine the appropriate arithmetic architecture without sacrificing milli degree estimation accuracy. In the register-transfer-level design, the recommended milli degree estimation training accelerator was constructed in Verilog and functionally verified using the Vivado Verilog Simulator. In the Milli, degree estimation has a Training/Inference engine and involved analyzing several floating-point formats and improving the FP operators. It can operate at frequencies up to 80 MHz, increasing the circuit's throughput. It uses 5.44 times less energy and takes up five times less space than its predecessors., and has great accuracy.

Table.1 Proposed ternary based double-precision floating-point adder

Adders	**Slices**	**Delay (ns)**	**FF's**	**Power (W)**	**Slice LUT's**	**Area (Slices)**	**Memory**	**Frequency MHz**
Kogge-Stone [14]	1140	$2.87e^{-11}$	2021	7.69m	---	---	---	---
Sklansky [15]	1047	$11.15\ e^{-11}$	1724	36.34m	---	---	---	---
Brent-Kung [14]	1268	$9.24\ e^{-11}$	1536	0.4micro	---	---	---	---
Han-Carlson [15]	992	$82.21\ e^{-11}$	4990	13.54m	---	---	---	---
Villalba [16]	1655	12.584	1642	0.6micro	---	---	---	---
Proposed Ternary based double precision floating point adder	**607**	**6.940ns**	**606**	**0.088**	**334**	**208**	**145KB**	**144.09**

V. CONCLUSION

The addition is the primary operation in signal processing, machine and deep learning, such as matrix and vector addition or average computation. This paper investigates the impact of efficient FP ALU for high-speed FP formats on the implementation of FP arithmetic in FPGAs without native FP capability. We developed single-precision and double-precision floating-point arithmetic operations in this study, which were subsequently implemented on an FPGA for signal processing with the MAC module using the Verilog programming language. More efficient structures in FPGA-based designs while maintaining the precision operations can use this strategy to construct efficient arithmetic circuits. When compared to the IEEE-754 standard, the proposed converters to/from any radix allow working at high speeds inside the FPGA and returning output numbers in IEEE-754 format.The results of the implementation suggest that the huge area and latency benefits of utilizing FPGAs to build high-speed FP adders may offset the disadvantages of using FPGAs to implement high-radix FP multipliers. For algorithms that contain both adds and multiplications, this work allows selecting the optimal acceptable radix of the FP arithmetic representation as a function of the ratio between the number of additions and multiplications in the targeted algorithm. This guidance allows the designer to choose between maximum speed, area, or a trade-off choice, depending on the needs of the designer's specific application. The number of additions equals the number of multiplications in numerous processes, including dot product, matrix by matrix multiplications, convolutions, and others.

REFERENCES

1. Srujana B Malkapur et al, “Design of Generic Floating Point Pipeline Based Arithmetic Operation for DSP Processor”, Proceedings of the Second International Conference on Inventive Research in Computing Applications (ICIRCA-2020), 978-1-7281-5374-2/20/$31.00 2020 IEEE.
2. Manish Kumar Jaiswal et al, “SP48E Efficient Floating Point Multiplier Architectures on FPGA”, 2017 30th International Conference on VLSI Design and 2017 16th International Conference on Embedded Systems, DOI:10.1109/VLSID.2017.106, 2380-6923/16 $31.00 2016 IEEE.
3. S¨uleymanSavas et al, “Efficient Single-Precision Floating-Point Division Using Harmonized Parabolic Synthesis”, 2017 IEEE Computer Society Annual Symposium on VLSI, DOI:10.1109/ISVLSI.2017.28, 2159-3477/17 $31.00 2017 IEEE.
4. SoumyaHavaldar et al, “Design Of Vedic IEEE 754 Floating Point Multiplier”, IEEE International Conference On Recent Trends In Electronics Information Communication Technology, May 20-21, 2016, India, 978-1-5090-0774-5/16/$31.00 2016 IEEE.
5. Spoorthi M. N. et al, “Decimal Multiplier with Improved Speed Using Semi-Parallel Iterative Approach”, 978-1-7281-9369-4/20/$31.00 2020 IEEE.
6. Farhad Merchant et al, “Efficient Realization of Table Look-up based Double Precision Floating-Point Arithmetic”, 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems, DOI: 10.1109/VLSID.2016.113, 978-1-4673-8700-2/16 $31.00 2016 IEEE.
7. Deeksha Kiran D K et al, “VLSI Implementation of Braun Multiplier using Full Adder”, International Conference on Current Trends in Computer, Electrical, Electronics and Communication (ICCTCEEC-2017), 978-1-5386-3243-7/17/$31.00 2017 IEEE.
8. Sangeetha P et al, “Comparison of Braun Multiplier and Wallace Multiplier Techniques in VLSI”, Fourth International Conference on Devices, Circuits and Systems (ICDCS'18), 978-1-5386-3476-9/18/$31.00 2018 IEEE.
9. SnehaHiratkar et al, “VLSI Design of Analog Multiplier in Weak Inversion Region”, 978-1-5090-0396-9/16/$31.00 2016 IEEE.
10. ShuchiNagaria et al, “Efficient FIR Filter Design using Booth Multiplier for VLSI Applications” 978-1-5386-4491-1/18/$31.00 2018 IEEE.
11. Minal R. Ghonge et al, “VLSI Design of Fixed Width 2’s Compliment Multiplier”, 2017 International Conference on Innovations in information Embedded and Communication Systems (ICIIECS), 978-1-5090-3294-5/17/$31.00 2017 IEEE.
12. Siva.S.Sinthura et al, “Implementation and Analysis of different 32-bit multipliers on aspects of Power, Speed and Area”, Proceedings of the 2nd International Conference on Trends in Electronics and Informatics (ICOEI 2018), 978-1-5386-3570-4/18/$31.00 2018 IEEE.
13. Shirin Pourashraf et al, “±0.25-V Class-AB CMOS Capacitance Multiplier and Precision Rectifiers”, IEEE transactions on Very large scale integration (VLSI) Systems, DOI: 10.1109/tvlsi.2018.2881249, 1063-8210 2018

14. Ushasree G et al, "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), 978-1-4673-5758-6/13/$31.00 2013 IEEE.
15. Manish Kumar Jaiswal et al, "Configurable Architecture for Double / Two-Parallel Single Precision Floating Point Division", 2014 IEEE Computer Society Annual Symposium on VLSI, DOI:10.1109/ISVLSI.2014.45, 978-1-4799-3765-3/14 $31.00 2014 IEEE.
16. Villalba, J., Hormigo, J. High-Radix Formats for Enhancing Floating-Point FPGA Implementations. *Circuits Syst Signal Process* **41,** 1683–1703 (2022). https://doi.org/10.1007/s00034-021-01855-x
17.