iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** IV **Month of publication:** April 2023

DOI: https://doi.org/10.22214/ijraset.2023.50739

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

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 IV Apr 2023- Available at www.ijraset.com

Implementation of Pipelined Multi_Precision (1, 2 and 4) Floating-point Arithmetic Operations

Prof. Bhagya[1], Ramanagowda G S[2], Rohan S[3], Soundarya D R[4], Yogitha K[5]
[1]*Assistant Professor,* [2, 3, 4, 5]*Student, Electronics and Communication, East West Institute of Technology, Bangalore, India*

Abstract: ***The implementation of pipelined multi-precision-based arithmetic operations are carried out. In the existing system, the floating-point operation is based on single precision and is implemented on a divider. The proposed design has been implemented using single, double and quad precision using the universal piece-wise linear (PWL) approximation method and a modified Goldschmidt algorithm. The proposed design performs addition, subtraction, multiplication, and division using the universal PWL method to reduce maximum error. Small multipliers are used in the modified Goldschmidt algorithm. The pipe-lining process has been used in order to improve the speed of execution and accuracy. This pipelined architecture is described in Verilog and timing performance is measured with Xilinx timing analyser.***
Keywords: Pipelined architecture, precision floating Point numbers

I. INTRODUCTION

In the view of the fact that complexity of the algorithms for the floating-point operations are very hard to implement on FPGA. The computations for floating point operations involve large dynamic range, but the resources way required for these operations is high compared with the integer operations. We have unsigned/signed multiplier for multiplication of binary numbers, for multiplication of floating-point numbers floating point division is used. Floating point numbers are one possible of representing real numbers in binary format; The IEEE 754 standard presents two different floating-point formats, Binary interchange format and Decimal interchange format. Dividing floating point numbers is a critical requirement for DSP applications involving large dynamic range. The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point computation which was established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE IEEE Standard 754 floating point is the most common representation today for real numbers on computers, including Intel-based PC's, Macs, and most Unix platforms.

There are several ways to represent floating point number but IEEE 754 is the most efficient in most cases. IEEE 754 has 3 basic components: The Sign of Mantissa: This is as simple as the name. 0 represents a positive number while 1 represents a negative number. The Biased exponent: The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent. The Normalized Mantissa: The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e., O and 1. So a normalized mantissa is one with only one 1 to the left of the decimal. IEEE 754 numbers are divided into Three based on the above three components: single precision and double precision, Quad Precision.

II. METHODLOGY

MODIFIED GOLDSCHMIDT ALGORITHM is used to find precision values. Hence in Goldschmidt methods, the value of q becomes closer to the accurate value of the result of Y/X with an increasing number of iterations. From the iterative equations, three advantages of the modified Goldschmidt algorithm can be identified, as follows.

1) The effective word length of R is reduced. The smaller multiplication and addition units meet the requirements for the iteration over R.
2) Complementation is avoided by removing subtraction from the recursive equations.
3) The multiplication is replaced by the square operation. the proposed method for multi precision arithmetic operation classified into different modules, namely

XOR Module: XOR is the digital logic gate which provides logic 0 or logic1 as the output. Sign bit performs Xor operation. Goldschmidt block: The precision values are calculated using modified Goldschmidt algorithm.

Exponent bias addition: The resultant exponent value will be subtracted by the bias value to get the normalized value. The bias value varies based on the precision type.

Floating Point Adder and Subtractor: Addition and subtraction of floating-point numbers is carried out using IEEE 754 standard floating point Addition Algorithm. The resultant sign value decides the arithmetic operation.

Floating Point Multiplier: Multiplication of floating-point numbers is performed by addition of exponents and multiplication of the mantissas.

Floating Point Divider: Division of IEEE 754 Floating point numbers is done by dividing the mantissas and subtracting the exponents.

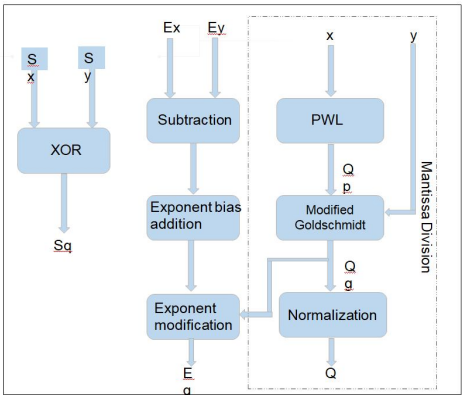


Fig 1: Block Diagram Of Floating-Point Divider

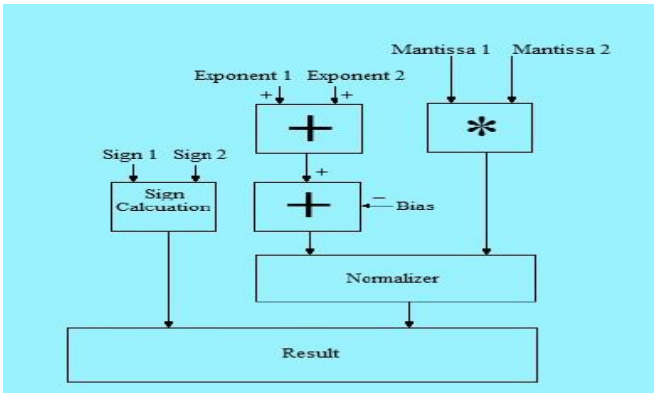


Fig 2: Block Diagram Of Floating-Point Multiplier

III. IMPLEMENTATION

The proposed method for precision floating-point numbers is based on a universal Piece wise linear algorithm method and a modified Goldschmidt algorithm. A state-of-the-art universal PWL method is used to implement the initial approximation of the reciprocal. In the modified Goldschmidt algorithm, full precision multipliers are replaced with smaller multipliers. The design is going to describe a single precision floating point divider for better Area & timing performance. Xilinx Timing Analyzer is used to reduce the power consumption and to increase the speed of execution by implementing certain algorithm for dividing two floating point numbers. By the use of pipelining process this divider is very good at speed and accuracy compared with the previous Dividers. The pipelined architecture is described in Verilog and is implemented on Xilinx Spartan 6 FPGA. Timing performance is measured with Xilinx Timing Analyzer.

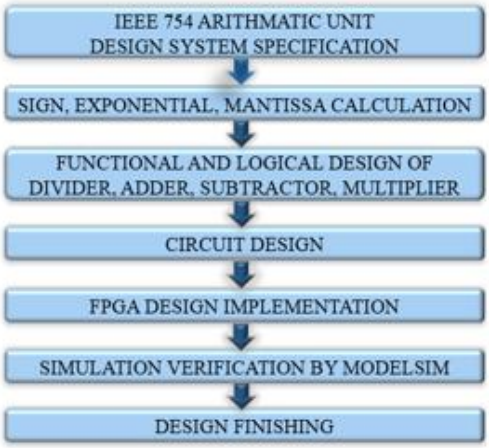


FIG 3:DESIGN FLOW

IV. RESULTS

outputs for the proposed model are explained using an example. Consider 2 inputs namely x=2.1 and y=2.0. double precision floating point subtraction operation is performed with 2 inputs x&y. Thus, the simulated results in Xilinx are shown in the below fig 4. Hence the results are verified using FPGA spartan 6. obtained result in FPGA spartan 6 is shown in the below fig 5. Table 1 describes the total area and dealy utilised in the proposed method,where area and delay are compared with existing model which provides a better performance in terms of area and delay.

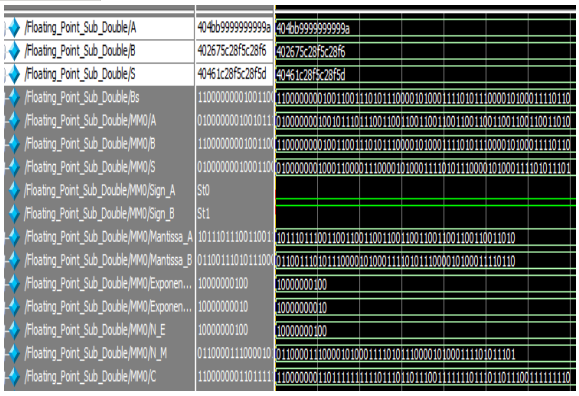


Fig4: simulation of single precision adder

Fig5: snapshot of FPGA

Considering the inputs x=42.3 and y=12.4 single precision floating point division operation is performed. Hence the result are simulated using Xilinx and are verified using FPGA spartan 6. Thus, the obtained outputs are shown below in fig6 and fig7.

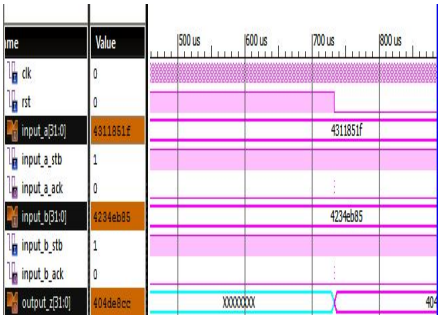


Fig6:simulation of single precision divider

Fig7: snapshot of FPGA

DEVICE NAME	AREA		DELAY
VERTEX 4	LUT	SLICES	OVER DELAY
Single Precision Floating Point Divider	5246	2880	45.6ns
Propose Single Precision Floating Point Divider	862	455	4.692ns

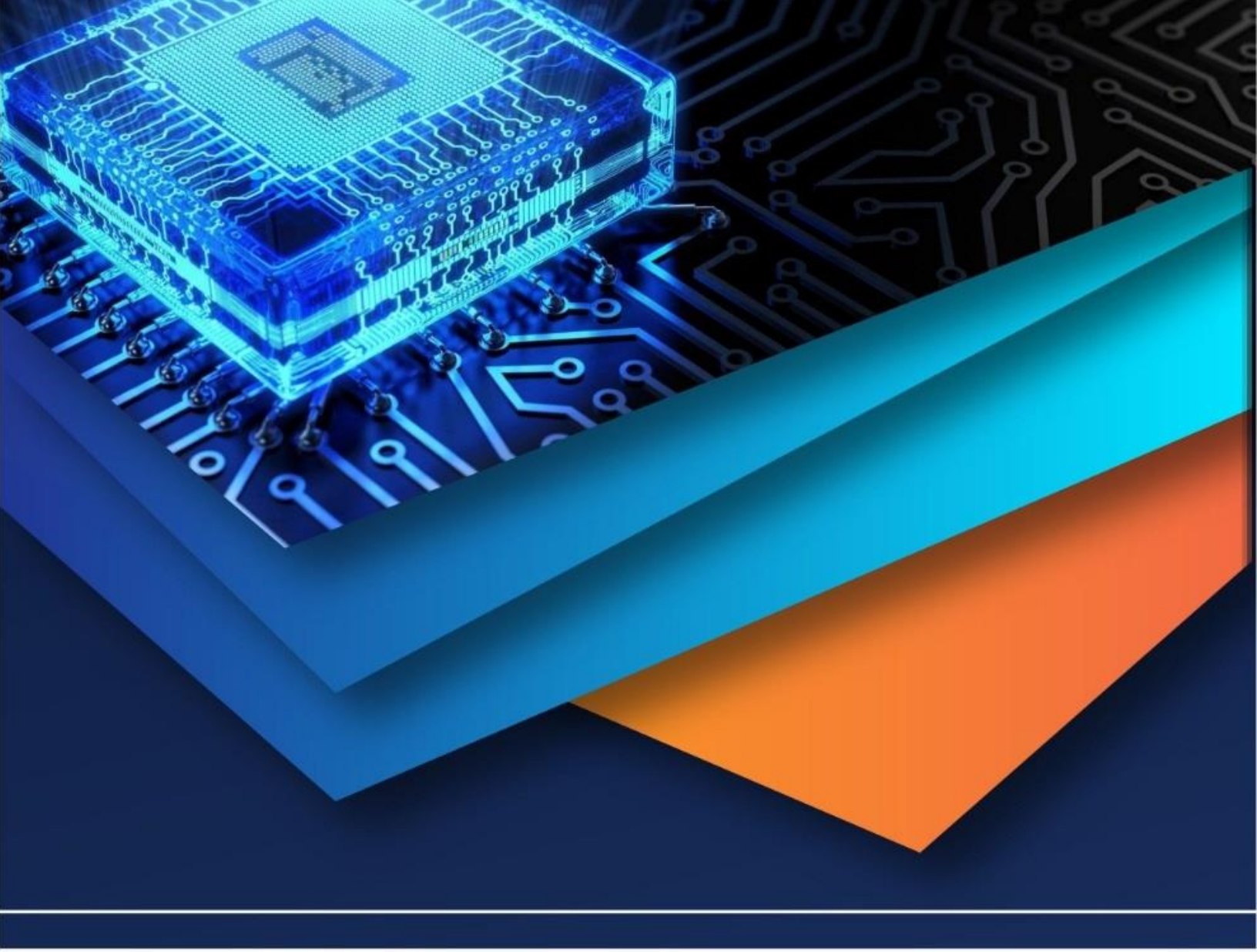
table 1: comparison of synthesized report

V. CONCLUSIONS

The implementation of pipelined multi precision floating point arithmetic operations is executed successfully. The time taken to complete pipelining techniques with delay is noted which is in the order of nano seconds. In comparison to the reference mode, the proposed system obtains better result in terms of area and delay.

REFERENCES

[1] Y. Yang, Q. Yuan, and J. Liu, "An architecture of area-effective high radix floating point divider with low-power consumption," IEEE Access, vol. 9, pp. 40039–40048, 2021.
[2] F. Lyu, Z. Mao, J. Zhang, Y. Wang, and Y. Luo, "PWL-based architecture for the logarithmic computation of floating-point numbers," IEEE Trans. Very Large-Scale Integer. (VLSI) Syst., vol. 29, no. 7, pp. 1470–1474, Jul. 2021.
[3] P. Malik, "High throughput floating-point dividers implemented in FPGA," in Proc. IEEE 18th Int. Symp. Design Diag. Electron. Circuits Syst., Apr. 2015, pp. 291–294.
[4] J. P. Wang, S. R. Kuang, and S. C. Liang, "High-accuracy fixed-width modified booth multipliers for lossy applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 1, pp. 52–60, Jan. 2011.
[5] S. Galal and M. Horowitz, "Energy-efficient floating-point unit design," IEEE Trans. Compute., vol. 60, no. 7, pp. 913–922, Jul. 2011.
[6] K.-N. Han, A. F. Tenca, and D. Tran, "High-speed floating-point divider with reduced area," Proc. SPIE Math. Signal Inf. Process., vol. 7444, Oct. 2009, Art. no. 74440O.
[7] T.-J. Kwon and J. Draper, "Floating-point division and square root using a Taylor-series expansion algorithm," Microelectron. J., vol. 40, no. 11,pp. 1601–1605, Nov. 2009.
[8] M. D. Ercegovac, L. Imbert, D. W. Matula, J. M. Müller, and G. Wei,"Improving Goldschmidt division, square root, and square root recipro-cal," IEEE Trans. Comput., vol. 49, no. 7, pp. 759–763, Jul. 2000.
[9] G. W. Bewick, "Fast multiplication: Algorithms and implementation,"Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA,1994.
[10] P. Surapong and F. A. Samman, "Floating-point division operator basedon CORDIC algorithm," ECTI Trans. Comput. Inf. Technol. (ECTI-CIT),vol. 7, no. 1, pp. 79–87, Jan. 1970.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY