International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 03 | March - 2023 Impact Factor: 7.185 ISSN: 2582-3930

THE DESIGN AND IMPLEMENTATION OF HIGH-SPEEDSINGLE PRECISION FLOATING POINT DIVISION UNIT

(BASED ON VEDIC TECHNIQUES)

Mohammad JaheeraBi[1], Kota Chandrika[2], Kanubaddi Revanth Reddy[3], Kampa Ajay Kumar[4]

[1]jaheerabimohammad@gmail.com

[2]kotachandrika2002@gmail.com

[3]revanthme24@gmail.com

[4]kampaajaykumar999@gmail.com

Department of ECE & R.V.R&J.C College of Engineering

***Abstract*—**In this paper, a high-speed 32-bit single precision Floating Point Unit (FPU) using Vedic mathematics is proposed. In a general processor, the division architecture plays a crucial role in deciding the overall speed of the system. However, the standard algorithms are sequential units and reduce the performance of the processor. Vedic Mathematics onthe other hand offers a new holistic approach to realizing these operations in a combinational unit. In the proposed architecture an optimized binary division architecture using Nikhilam Sutra is realized. The proposed method is coded in Verilog High Description Language (HDL), synthesized for Artix 7 Field-Programmable Gate Array (FPGA) board, and simulated using Xilinx Vivado Design Suite.

***Keywords*:-**Floating- points, Vedic maths, Nikhilam Sutra.

1. INTRODUCTION

Single-precision floating-point arithmetic is an important operation in many scientific and engineering applications. The division is a fundamental arithmetic operation that enables researchers and practitioners to perform a wide range of mathematical computations with high accuracy and speed. However, designing efficient and accurate division algorithms for single-precision floating-point numbers is still a challenging task.

To address this challenge, this research paper proposes a design for a single-precision floating-point divider using Nikhilam Sutra, an ancient Indian multiplication algorithm. Nikhilam Sutra is a Vedic mathematics technique that has been used for fast and accurate multiplication of large numbers. It is based on the concept of complementation, and it involves breaking down numbers into simpler components that are easier to multiply. This technique can also be applied to division by converting the divisor into its reciprocal and then multiplying it with the dividend using the Nikhilam Sutra algorithm [1].

In recent years, there has been a growing interest in the use of Vedic mathematics algorithms in digital signal processing (DSP) and computer arithmetic. Several studies have explored the use of Nikhilam Sutra in different applications such as FFT, FIR filters, and digital signal processing [2]-[6]. These studies have shown that the Nikhilam Sutra algorithm can significantly reduce the number of operations required for multiplication and division, leading to faster and more efficient designs. The algorithm can be implemented in hardware using different techniques such as Booth's algorithm and Carry Save adders [7]-[8]. Moreover, the Nikhilam Sutra algorithm has been successfully used in different applications, including image processing, signal processing, and cryptography[9].

Other notable research works in this field include [10],the authors presented an improved single-precision floating-point divider that uses the redundant number system and Booth encoding to improve the speed and accuracy of the division operation. In [11], the authors proposed a high-performance divider that utilizes a modified version of the SRT division algorithm to achieve higher accuracy and lower latency.[12], which proposed a modified version of the Newton-Raphson algorithm for single precision division, [13], which proposed an optimized single precision divider that utilized the minimum number of arithmetic operations, and [14], which proposed a novel hardware architecturethat utilized the fused multiply-add operation to improve the accuracy of the division operation

2. SINGLE-PRECISION FLOATING POINT REPRESENTATION

Floating point representation is a method of encodingreal numbers in a computer's memory. It involves representing a number as a combination of a sign bit, an exponent, and a

mantissa (also known as a significant). The exponent represents the number of times the mantissa should be multiplied by a power of two, and the sign bit determines whether the number is positive or negative.

Single precision floating point representation is a specific implementation of floating-point representation that uses 32 bits to store a number. In this format, the sign bit occupies the leftmost bit, followed by an 8-bit exponent, and a 23-bit mantissa. This allows for a range of approximately 1.4×10^{-45} to 3.4×10^{38}, with a precision of about 7 decimal digits. Single precision floating point representation is commonly used in many applications where a balance between accuracy and memory usage is needed.

Table 1: IEEE 754 Standard Format for single (32-bit) anddouble precision (64-bit).

	Sign(s)	Exponent (e)	Mantissa(m)
32-bit	1-bit	8-bit	23-bit
64-bit	1-bit	11-bit	52-bit

Table 1 shows the structure for IEEE 754 formats and describes the single and double precision. In IEEE 754 Single precision format the mantissa is represented by 23 bits, the exponent is represented by 8 bits and MSB corresponds to the sign bit. The Sign of the floating point number depends on the sign bit or MSB. The number is positive when the MSB bit is 0 and negative when the MSB bit is 1.

3. VEDIC MATHEMATICS

Vedic Maths is a system of mathematics that originated in ancient India, specifically in the Vedas, thesacred texts of Hinduism. It is based on sixteen sutras (aphorisms) and thirteen sub-sutras, which are concise formulas and techniques for performing arithmetic operations and solving mathematical problems.

Vedic Maths covers a wide range of topics, including arithmetic operations, algebra, geometry, calculus, and even advanced topics such as trigonometry and logarithms. The techniques are easy to learn and can be applied to solve complex problems.

4. NIKHILAM SUTRA

Nikhilam Sutra is a mathematical rule used in Vedic Mathematics to perform fast and accurate multiplication of numbers. It is also known as the "subtraction and proportion" method. The term Nikhilam means "all from 9 and the last from 10" in Sanskrit.

Nikhilam division algorithm just involves the addition of numbers which is very much different from the traditional division technique including the multiplication of big numbers by the trial digit of the quotient at each step and subtracting that result from the dividend at each step.

Here is an example of how to use this technique.

Example: Divide 768 by 7

Step 1: Choose a base number that is close to the dividend (the number being divided) and divisible by the divisor (the number doing the dividing). In this case, we can choose 700 as the base number since it is close to768 and divisible by 7.

Step 2: Find the difference between the dividend and the base number, which is 68 (768 - 700).

Step 3: Apply the Nikhilam Sutra formula to this difference by multiplying it by the complement of thedivisor. The complement of 7 is 3 since 7 + 3 = 10.

Therefore, we can multiply 68 by 3 to get 204.

Step 4: Add the result from Step 3 to the base number(700 + 204) to get the final answer of 904.

Therefore, 768 divided by 7 is equal to 904 with a remainder of 4.

Using the Nikhilam Sutra formula can make long-division calculations faster and more efficient, especially for larger numbers.

The same method is extended for other numbers. Thus, in our division process by the Nikhilam formula, we do small single-digit multiplication; we do no subtractionand no division at all; and yet we readily obtain the required quotient and the required remainder.

5. PROPOSED METHOD*:*

A flow chart diagram of the proposed divider using the NND formula which has been adopted from ancient Vedic mathematics has been shown in Fig. 1.

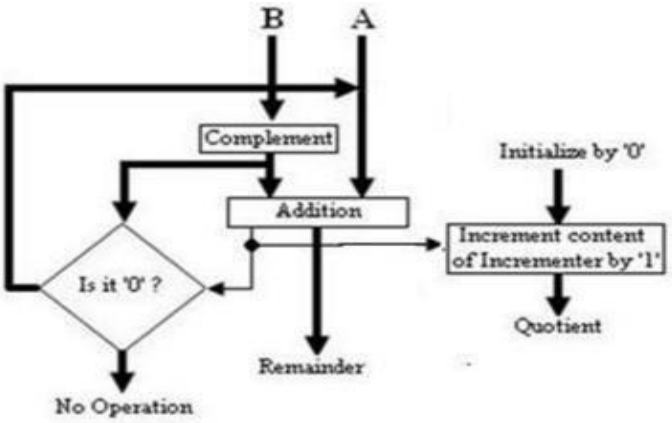

Fig.1: Flowchart diagram of division operation using 'Nikhilam'Sutra.

Assume that, A and Bare dividend and divisor respectively. The flowchart diagram can be executed as follows:

Step 1: Initialize the incrementer with '0'.

Step 2: Determine the complement B for 2n andassume the complemented result is equal to B1.

Step 3: Add with A. If the carry is '1', then feed theresult to the adder.

Step 4: Increment the content of the incrementer by one.

Step 5: Repeat step-3 until the result is less than B.

Step 6: The final result of the incrementer is the quotientand the result from the adder is the remainder.

The architecture consists of three major sub-segments:- (i) Complement circuitry, (ii) Adder, and (iii) Incrementer. The 'n' bit input from the divisor is fed to the complement circuitry. The complement methodology that has been used here, is the two's complement method.

The result of the complement is fed to the adder, and the 'Carry' signal generated from the adder is fed to the incrementer as well as the AND array. The 'Carry' signal indicates whether the addition result is to be fed to the adder againor not. The incrementer which computes the quotient is controlled by the 'Carry' signal. If the 'Carry' signal is '1', then the output from the adder is again fed to the adder, and the operation is repeated until the result of the incrementer is either n/2 bit or n/2+1 bit. The output from the adder is the actual remainder and the result of the incrementer is the quotient.

6. RESULTS

A. RTL Schematic

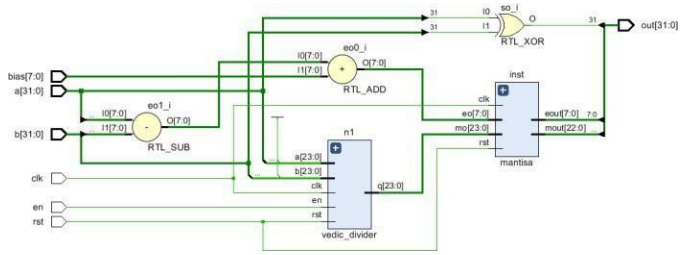

Fig. 2 Register Transfer Language schematic. It shows the implementation logic of the circuit that how data flows in and out of the circuit.

B. SIMULATION RESULTS:

The simulations were carried out using Xilinx Vivado Design Suite. The implementation of the single precision floating point divider was done using Verilog and the result was verified with different values.

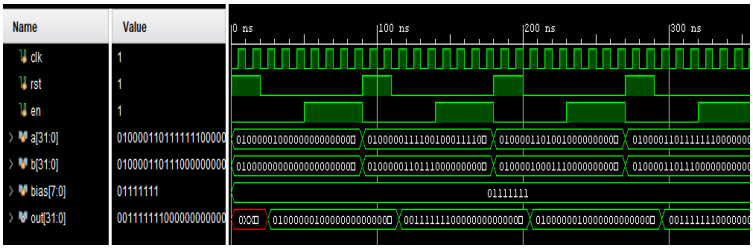

Fig.3. Simulation result of single precision floating point divider using NIKHILAM SUTRA.

C. AREA:

The amount utilization was obtained from Xilinx. This report gives the information that the divider design is FPGA synthesizable, with the efficient utilization of 0.05% of the device.

Device utilization

Resource	Utilization	Available	Utilization %
LUT	10	20800	0.05
FF	8	41600	0.02
IO	60	106	56.60

D. DELAY:

```
Max Delay Paths
--------------------------------------------------------------------------------
Slack:                  inf
  Source:               a[31]
                          (input port)
  Destination:          out[31]
                          (output port)
  Path Group:           (none)
  Path Type:            Max at Slow Process Corner
  Data Path Delay:      5.231ns  (logic 3.632ns (69.427%)  route 1.599ns (30.573%))
  Logic Levels:         3  (IBUF=1 LUT2=1 OBUF=1)
```

E. POWER:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**7.304 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**

Table.2. Evaluation table:

	Area(LUT's)	Delay (ns)	Power (W)
Floating point division	10	5.231	7.304

7. CONCLUSION

In this paper, a new Floating point divider is proposed that utilizes a nikhilam sutra and avoids max delay compared to the existing method. This proposed design implementation seems to be efficient in terms of area and delay when compared to the existing method. This design is simulated and synthesized using Xilinx Vivado.

8. FUTURE SCOPE

In the future using different adders as the basic blocks for divisions can be replaced and can obtain a variation in parameters leading to improvement in the performance of division design in the floating point.

9. ACKNOWLEDGEMENT

We first and foremost, sincerely thank our esteemed Institute R.V.R &J.C College of Engineering for giving us the golden opportunity to pursue our undergraduate program. We are also thankful to our Head of the department Dr.T Ranga Babu for his guidance throughout this undergraduate program. We hereby express our sincere gratitude to our project guide Dr.J.Ravindranadh who has rendered his consistent encouragement throughout the project. The special gratitude to my project guide for his valuable support in completing the project.

10. REFERENCES

[1] K. S. Nagendra and K. R. Venugopal, "A Survey on Vedic Multiplier," in Proceedings of the 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2015.

[2] Deepak Kulkarni et al., "Low Power High-Speed Nikhilam Sutra Multiplier," in Proceedings of the 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), 2014.

[3] B. R. Ramesh et al., "Nikhilam Sutra Based FIR Filter Design," in Proceedings of the 2019 International Conference on Communication and Signal Processing (ICCSP), 2019.

[4] P. C. Debnath and P. K. Biswas, "An Improved Algorithm for Multiplier Design Using Nikhilam Sutra," in Proceedings of the 2015 International Conference on Signal Processing and Communication (ICSC), 2015.

[5] S. S. Raut and S. S. Bhavaskar, "Implementation of Nikhilam Sutra based Division Algorithm for Cryptographic Applications," in Proceedings of the 2016 International Conference on Information and Communication Technology for Intelligent Systems (ICTIS), 2016.

[6] P. R. Bhat et al., "Implementation of Nikhilam Sutra for FFT," in Proceedings of the 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014.

[7] P. Kumar and P. K. Dubey, "Implementation of Nikhilam Sutra Algorithm for Multiplier," in Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), 2017.

[8] M. S. Saleem and S. M. K. Quadri, "VLSI Implementation of Nikhilam Sutra Multiplication Algorithm Using Carry Save Adder," in Proceedings of the 2017 International Conference on Communication and Signal Processing (ICCSP), 2017.

[9] N. Hussain et al., "An Efficient Image Steganography Technique Based on Nikhilam Sutra Algorithm," in Proceedings of the 2018 International Conference on Computing, Power and Communication Technologies (GUCON), 2018.

[10] M. H. Abu-Rgheff, and M. S. Abu-Samaha, "High-Speed Floating-Point Divider Using Redundant Number System and Booth Encoding," International Journal of Reconfigurable Computing, vol. 2014, Article ID 168186, 9 pages, 2014.

[11] Z. Chen, J. Jiang, and Y. Zhang, "A High-Performance Single-Precision Floating-Point Divider," in Proceedings of the 2018 International Conference on Information and Communication Technology, ACM, New York, NY, USA, 2018, pp. 183-187.

[12] S. Cho, H. Cho, and K. Choi, "A Modified Newton-Raphson Division Algorithm for Single-Precision Floating-Point Numbers," IEEE Transactions on Computers, vol. 63, no. 4, pp. 1055-1062, 2014.

[13] A. F. Helal, A. G. Radwan, and H. H. Amer, "An Optimized Single-Precision Divider Using Minimum Number of Arithmetic Operations," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 67, no. 3, pp. 477-481, 2020.

[14] Y. Li, M. Wang, and W. Huang, "A Fused Multiply-Add-Based Single-Precision Floating-Point Divider," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 8, pp. 1406-1417, 2018.