International Journal of Engineering & Technology, 7 (3.6) (2018) 48-50

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Design and Synthesis of Restoring Technique Based Dual Mode Floating Point Divider for Fast Computing Applications

Shaikh Salman Faraz[1], Yogesh Suryawanshi[2], Sandeep Kakde[3], Ankita Tijare[4], Rajesh Thakare[5]

[1]*Department Of Electronics Engineering, Y C College Of Engineering, Nagpur, India.*
[2]*Department Of Electronics Engineering, Y C College Of Engineering, Nagpur, India.*
E-Mail:Yogesh_Surya8@Rediffmail.Com
[3]*Department Of Electronics Engineering, Y C College Of Engineering, Nagpur, India.*
E-Mail:Sandip.Kakde@Gmail.Com
[4]*Department Of Electronics Engineering, Y C College Of Engineering, Nagpur, India.*
E-Mail:Ankita.Tijare@Gmail.Com
[5]*Department Of Electronics Engineering, Y C College Of Engineering, Nagpur, India.*
E-Mail:Rdt2909@Gmail.Com
**Corresponding Author E-Mail:Salmanfaraz3@Gmail.Com*

Abstract

Floating point division plays a vital role in quick processing applications. A division is one amongst the complicated modules needed in processors. Area, delay and power consumption are the main factors that play a significant role once planning a floating point dual-precision divider. Compared to different floating-point arithmetic, the design of division is way a lot of sophisticated and needs longer time. Floating point division is that the main arithmetic unit that is employed within the design of the many processors in the field of DSP, math processors and plenty of different applications. This paper relies on the dual-mode practicality of floating point division. The proposed designed architecture supports the single precision (32-bit) as well as double precision (64-bit) IEEE 754 floating point format. It uses restoring division technique for the fraction part division. This design consists of varied sub-modules like shifters, exceptional handlers, Normalizers and many more.

Keywords: *Floating point division, shifter, exceptional Handler, normalizer, LUT, FPGA.*

1. Introduction

Floating point arithmetic architectures underwent vital improvement by research work in the recent many decades. Floating point arithmetic is an elementary module for several usual scientific and engineering domain applications. The floating point arithmetic unit is designed using conventional illustration as prescribed by IEEE 754 format. Floating point computation is commonly found in systems which demand mathematical operations on a high range of numbers. A floating point means that there is variable decimal point position due to the exponent factor. However, floating point representations are not only less precise but also have less operational speed than fixed-point representations. Floating point divider is employed in ALU units of various processors. It is crucial to implement high-performance floating-point complex division usage in an application with a high-speed real-time requirement. A divider is one amongst the important hardware sub-modules in most of the processes like DSP application based embedded processors, encoding and decoding techniques in cryptography and in numerous logical computations.

This paper is well-arranged as follows. Section I is the short overview of floating point arithmetic. Section II highlights the previous work on Floating point division architectures and their issues. Section III focuses on the mathematical analysis of the restoring technique and the implementation of the division algorithm using the specified flowchart. Section IV shows the results of simulation on the XILINX ISE Design Suite 13.1. Section V is the conclusion about the specifications of area, power and latency. Section VI provides references to the proposed work.

2. Previous Work

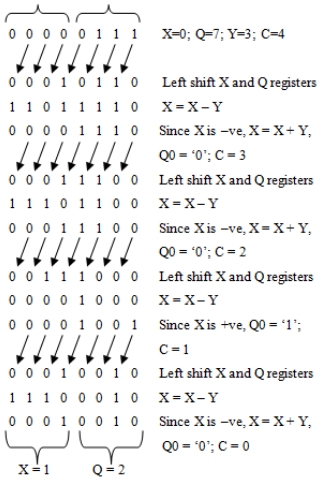
In paper [1] author planned a design for dual mode floating point division. This design is intended for dual-mode practicality, which either work out a double precision division or a pair of single precision division in parallel depending on the mode of operation. The series expansion multiplicative technique is employed for fraction division operation. A Modified Radix-4 Booth Multiplier having a dual-mode feature is employed for the dual-mode fraction part division. For the purpose of high-performance demand of division arithmetic computation, the main purpose of this research paper is for dual-precision (single or double precision) dual-mode architecture of floating point division. In paper [2] author used the Goldschmidt computational division technique for designing a single precision floating point division. A 32-bit floating point multiplier and subtractor are used for floating point division using the specified method. The most feature of this projected work is that the design for performing fraction part division by 32-bit floating point multiplier and is implemented by employing a 24-bit Vedic multiplication (Urdhva-Triyakbhyam-Sutra) technique. This multiplier has the higher speed of operation which ends up in an increase in the throughput

results of the floating point divider. The core aim is to create the planned floating point divider using Verilog HDL on FPGA. In this division technique, the dividend and divisor are computed employing a factor by which the divisor will approach one, and the dividend will be the quotient of the division. This Vedic multiplication principle is employed for calculating fraction part division. In paper [3] author designed single precision floating point divider using Vedic division logic. In this paper, the various algorithms are compared including Restoring algorithm, Non-restoring technique, SRT division algorithm, Nikhlam Sutra Vedic method.

3. Design Methodology

The proposed work uses the bit restoring method for the mantissa division. The inputs are divided depending on the mode input bit. The proposed architecture is capable of doing single precision floating point division as well as the double precision floating division. So, the bit restoring algorithm use two registers, one register is used to have a dividend which is input and other register is zero at the initial stage of the process. At the start, the count value is kept at zero. Final count value depends on the length of the mantissa division. Following example is of 4 bits division having divided 7 (0111) and divisor 3(0011). Let divided be in the Q register and divisor be in the Y register and register X kept as zero. As shown in the following flowchart, from the right side the four bits are X register and from the left side, the four bits are Q register. So, both X and Q register are of 4 bits length. Therefore, the mantissa division will be of 4 bits length. C register will be the counter. The steps of bit restoring division algorithm are as follows:

1. Store the dividend and divisor in registers Q and Y respectively.
2. Initialize X register as zero.
3. Shift the MSB of Q register to LSB of X register and set the LSB of Q register to logic '0'.
4. Perform X = X - Y and check MSB of X register.
5. If MSB of X register is '1' then restore X register means X = X + Y and repeat step 3.
6. If MSB of X register is '0' then set the LSB of Q register to logic '1' and repeat step 3.
7. At the end of n iterations (n = length of dividend or divisor), the contents of Q register will be quotient.
8. Note that the divisor and dividend must have equal length.



Register X is a remainder and register Q is quotient.

Fig. 1: Flow chart of an Algorithm

Single Precision (Parallel) Format of inputs and Output:

Sign (1bit)	Exponent (8bit)	Mantissa (23bit)	Sign (1bit)	Exponent (8bit)	Mantissa (23bit)

Double Precision Format of inputs and Output:

Sign(1bit)	Exponent(11bit)	Mantissa(52bit)

The architecture illustration of floating point divider is shown in fig. 2. The data extraction unit separates out the sign bit, exponent bits and mantissa bits from the inputs. The data extraction also determines the mode of division depending on the input mode bit. If the mode is logic '0' then the single precision division will perform. If the mode is logic '1' then division performed will be in double precision mode. Exception handling is the process to check whether input(s) are zero, infinity or invalid. Subnormal handler is used to concatenate logic 0 or logic 1 to the mantissa of both the inputs depending on the subnormal condition. The leading one detector detects the leading one of dividend as well as the divisor. It also shifts both to the right in order to normalize divisor and dividend. Then, the normalized dividend and divisor will be applied to mantissa division unit to perform division. The normalized result of mantissa division will be mantissa of the final output. Exponent normalizer is used to adding the bias value to exponent bits according to the mode. The EX-OR logic gate is used for sign computation of the final output.

Table I: Truth Table of Exceptional Handling Cases

IN1 (Numerator)	IN2 (Denominator)	Output
NaN	NaN	NaN
Infinity	Infinity	Infinity
Infinity	-	Infinity
-	Zero	NaN
Zero	-	Zero

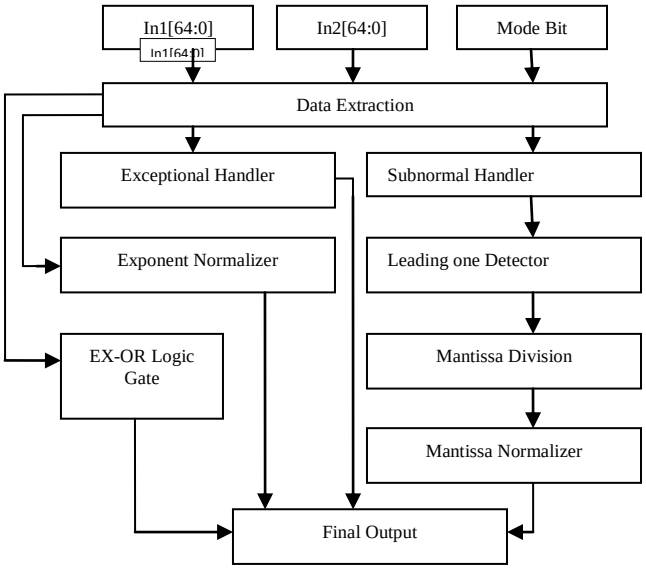


Fig. 2: Block Diagram of Proposed Work

4. Results and discussion

The architecture implementation of floating point divider is carried out using Verilog HDL and synthesized using XILINX ISE Design Suite 13.1.The synthesis tool used is XST and project is simulator by ISIM simulator tool. The FPGA device utilization is shown in the Table I. The targeted device for the project is XILINX Virtex-V family is XC5VLX50 of package FF324. Table I shows the FPGA device utilization summary.

Table II: FPGA Summary

Floating Point Division Architecture FPGA Family: Xilinx Virtex-V Device: XC5VLX50			
Parameters	Used	Available	Utilization
No. of Slice Registers	1852	28800	6%
No. of LUTs	21930	28800	76%
FFs Pairs	1574	22208	7%
Bonded IOs	197	220	89%
BUFG/BUFGCTRLs	1	32	3%
Latency Time and Power Simulations			
Delay	246.313 ns		
Power	0.563W		

RTL Schematics of 64-Bit Floating Point Division is as shown in figure 3.

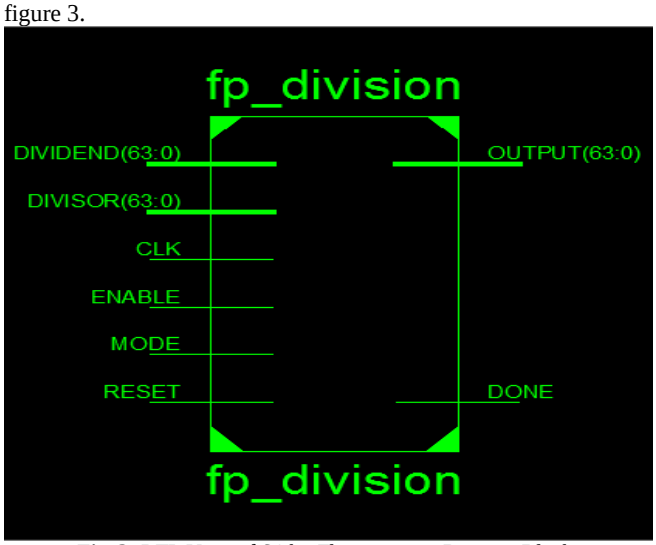


Fig. 3: RTL View of 64-bit Floating point Division Block

1] Single Precision Mode:

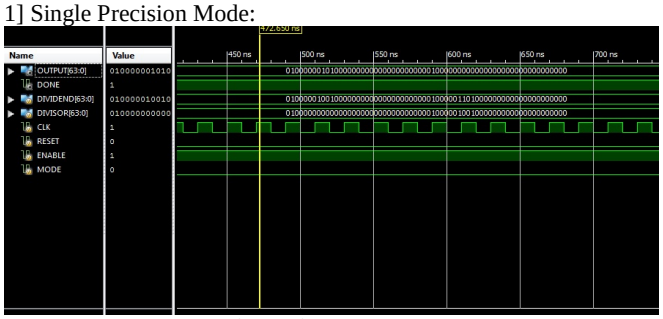


Fig. 4: Simulation Waveform of Single Precision Mode (two parallel divisions)

2] Double Precision Mode:

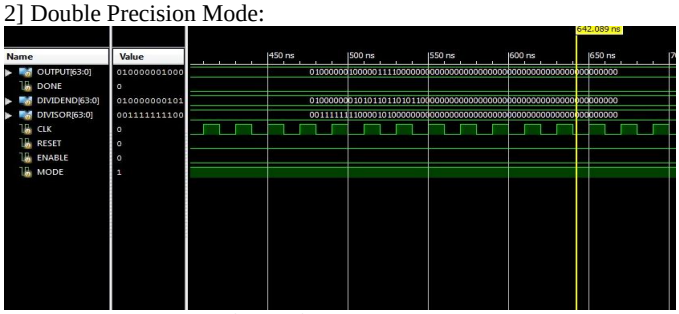


Fig.5: Simulation Waveform of Double Precision Mode (only one division)

For the single precision division, the mode is 0. The applied inputs DIVIDEND and DIVISOR are 64'b0100000100100000000000000000000010000011010000000000000000000000 (left side 32 bits are 10 in decimal and right side 32 bits are 20 in decimal) and 64'b0100000000000000000000000000000010000010010000000000000000000000 (left side 32 bits are 2 in decimal and right side 32 bits are 10 in decimal) respectively. After the simulation, the OUTPUT generated is 64'b0100000000010100000000000000000000000000000000000000000000000000 (left side 32 bits are 5 in decimal and right side 32 bits are 2 in decimal). Similarly, if the mode is 1 then double precision will perform. DIVIDEND and DIVISOR are 64'b0100000001010110110101100000000000000000000000000000000000000000 (91.34375 in decimal) and 64'b0011111111000010100000000000000000000000000000000000000000000000 (0.14453125 in decimal) respectively. Simulated result OUTPUT is 64'b0100000010000011110000000000000000000000000000000000000000000000 (632 in decimal).

5. Conclusion

Any processor or electronic system or ALU, which requires fraction computational capability, implements floating point representation. The division is one of the most vital functions of the arithmetic unit. The architecture is planned in such that it enhances the operation of the Arithmetic Logic Unit by parallel operation. The proposed design uses bit restoring technique for mantissa division. The project work supports single as well as double precision which increases arithmetic functionality. The maximum time required for the output after getting the clock signal is 246.313 ns. The power consumed by the proposed floating point divider is 0.563 W.

References

[1] Jaiswal MK & Hayden K, "Area-Efficient Architecture for Dual-Mode Double Precision Floating Point Division*", IEEE transaction on Circuits and Systems–i: Regular Papers,* (2017).

[2] Singh N, Sasamal TN & Anacan RM, "Design and Synthesis of Goldschmidt Algorithm based Floating Point Divider on FPGA*," IEEE International Conference on Communication and Signal Processing,* (2016).

[3] Ghatte N, Patil S & Bhoir D, "Single Precision Floating Point Division*", Fifth IRF international conference,* (2014).

[4] Huang S, Yu L, Han FJ & Luo Y, "A Pipelined Architecture for User-defined Floating Point Complex Division on FPGA*", IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE),* (2017).

[5] Oberman SF & Flynn M, "Division algorithms and implementation*", IEEE Trans. Comput.,* (1997).

[6] Jeong JC, Park WC, Jeong W, Han TD & Lee MK, "A cost-effective pipelined divider with a small lookup table*," IEEE Trans. Comput.,* (2004).

[7] Jaiswal MK & Cheung RCC, "High-performance reconfigurable architecture for double precision floating point division", *Proc. 8th Int. Symp. Appl. Reconfigurable Comput. (ARC), Hong Kong, China,* (2012), pp. 302–313.