IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 5, Issue 3, Ver. I (May - Jun. 2015), PP 58-62
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Design and Simulation of Double Precision Floating Point Division Using VHDL

Bharti W.Sakande[1], Dinesh Rotake[2]
[1]*Department of Electronics & Telecommunication, GHRIETW, RTM Nagpur university, India*
[2]*Department of Electronics & Telecommunication, GHRIETW, RTM Nagpur university, India*

Abstract: *This paper proposed the methodology for double precision floating point division using radix-8.This is the improved method of producing high speed. The architecture is based on look-up table and comparator. Double precision floating point division offer qualities like high speed on the expense of larger area and circuit complexity. Also floating point value can be represented by using IEEE-754 standard for division. This design will be simulated in Xilinx and it can be highly portable if it is design on a field programmable gate array (FPGA).*

Keywords: *Floating point unit, FPGA, IEEE-754, Radix-8, VHDL, Xilinx.*

I. Introduction

In the field of computation floating point method of division is highly important for high speed. The application of floating point division is found in Radar, Telecommunication, Financial Analytics, DSP and many other fields. Division is always consider to be bulky and one of the most difficult operation in arithmetic and hence all the implementation of division algorithm in VLSI architecture have higher order of time and space complexities.

In new proposed work I have implemented optimized division architecture using radix-8 or base-8.Radix-8 means division is to be performed in octal number system. Hence iteration period of division is to be minimized. As the radix-8 will be consider 3 bit retired in each iteration hence mantissa is to be calculated in fewer iteration.Radix-8 divider is about 33% faster and energy dissipated to perform a division is about the same with respect to radix-4.

In this system number of cycle to complete the operation is to be less required on the expense of more circuit complexity and power consumption.

II. Literature Review

Lot of work had been done by researcher in the area of design and simulation of double precision floating point division to increase the performance and speed. Some worked to minimized power consumption and area of unit also worked on obtaining high throughput and latency. From these some of work us listed below which referred for our project.

K. Scott Hemmert and Keith D. Underwood in their paper "Floating-point dividers in FPGA technology" had used a several FPGA specific optimizations. They have divided the divider into three parts: front end, core, and back end. In this paper the core is a fixed-point divide where the divisor can be assumed to have a 1 in the left most bit. This results in fully pipelined to 187-MHz iterative cores. They explored five different types of divide cores. Three cores are fully pipelined and two are iterative. These are Fully Pipelined, No restoring, Radix-2, Fully Pipelined, SRT, Radix-2.They have designed only for largest, fastest, highest throughput design that is only needed for divide sensitive applications to smaller iterative units with a lower throughput that is only suitable if divides are relatively rare.

Riya Saini, Galani Tina G. and R.D. Daruwala proposed an architecture that introduce a pipeline floating point arithmetic logic unit (ALU).Pipeline is used to give high performance and throughput to arithmetic operation. The main advantage of this design arithmetic unit based on IEEE standard for floating point numbers has been implemented on FPGA Board. The arithmetic unit implemented has a 64- bit pipeline processing unit which allows various arithmetic operations such as, Addition, Subtraction, Multiplication and Division on floating point numbers. But the serious disadvantage is the area requirement because of the presence of pipelined architecture. The clock frequency applied is 4.55 MHz and total power required are 0.38W.In future we can try to implement reduction in frequency and time required by divider.

Nicolas Brisebarre, Jean-Michel Muller, Member, IEEE, and Saurabh Kumar Raina proposed an architecture that used techniques for accelerating the floating-point computation of x=y when y is recognized earlier than x. The planned algorithms are slanting toward architectures with available fused-mac operations. These techniques can be used by compilers to accelerate some numerical programs without loss of accuracy.

Algorithm 1 always works and does not require much precomputation (so it can be used even if y is known a few tens of cycles only before x). Algorithm 2 is more rapidly and so far it requires much precipitation (for computing zh and z‘and making sure that the algorithm works), so it is more suited for division by a constant. Algorithm 4 always works and requires two operations only formerly x is known, but it require the accessibility of a slightly larger accuracy.

Shamna.K1 and S.R Ramesh proposed an architecture iterative and pipelined design of double precision floating point divider unit. The design shown here can produce performances that are comparable to, and in some case higher than, non-iterative designs based on number representations of higher radices. divider requires less area in case of iterative design. Since the pipelining of our iterative designs is intended to accelerate compute-intensive applications on FPGA chips, full unrolling of these iterative designs is highly desirable since it can produce maximum performance. But it cause significant area overhead. Hence partial unrolling of the design of the devider is done without any affect on the performance of the divider.In future The latency of the divider can be reduced by using a secondary clock for mantissa division alone. The frequency of the secondary clock is twice larger than the primary clock.

Xin Fang and Miriam Leeser proposed an architecture that has employed the fast divider design for double precision floating point that Capable to make use of FPGA resources as well as embedded multipliers. The design is based on table and match up to it to iterative and digit Recurrence implementations. In which division implementation targets performance with balanced latency and high clock frequency. The latency of first implementation is 14 clock cycles on both vendors’ hardware. The maximum frequency on a Stratix V device(5SGXB6) from Altera is 121MHz. Implemented on a Xilinx Virtex 6 device (XC6VLX75T), the maximum frequency is 148MHz.In the future, we will focus on improving the frequency by focusing on the critical path and annoying another steps of pipelining.

In this paper we will try to simulate a high speed double precision floating point divider. To achieve our proposed design, we used Radix-8 algorithm.

III. Approach

The approach of Double precision floating point division is shown in Fig.1 First consider Numerator and Denominator values then group it in to octal number.

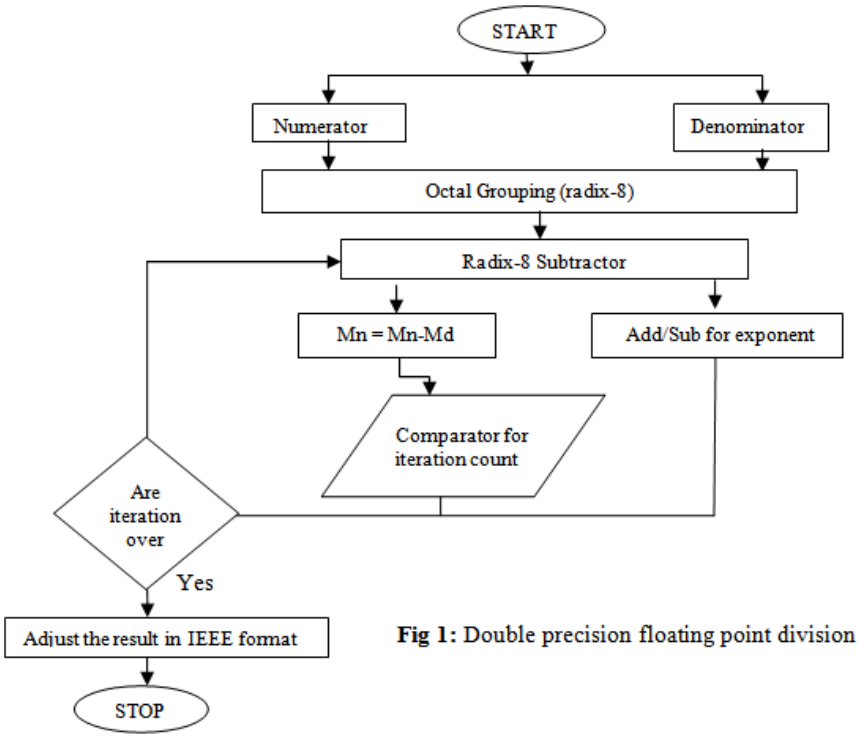

Fig 1: Double precision floating point division

Subtractor, subtract mantissa and add/sub the exponent. Comparator is used to count the iteration; either iteration is completed then adjusts the quotient and remainder in IEEE format or move on the subtractor to repeat the process till iteration will be over.

For handling the floating point numbers there are many complex issues to resolve with the in computers to ensure that the results are consistent when same program is run on different machines. This

problem was solved by introducing the IEEE standard for binary floating point arithmetic. The most compact representation of a floating point number define by IEEE 754 is the 64-bit floating point number format. Other than this IEEE 754 binary floating point formats such as half precision, single precision, double extended precision and quad precision are also available. From these we worked on double precision floating point binary number format. The format of IEEE 754 double precision floating point number is given in Fig.2

1: Sign bit (bit 63) (0 for positive and 1 for negative number)
2: Exponent which is of 11 bit (from bit 52 to 62)
3: Mantissa or Fraction which is of 52 bit (from bit 0 to 51)

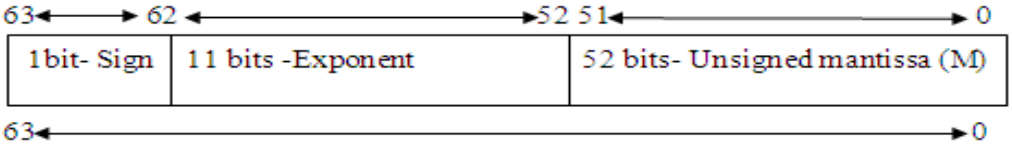

Fig.2: IEEE Double Precision Data Format

IV. Proposed Methodology

The proposed method of double precision floating point division overcomes the problem of speed by using radix-8 methodology. The value of radix-8 vary from -7 to +7 hence total 64 bit combination of numerator and denominator are consider in look up table for division.

The flow chart of Radix-8 division in which mantissa part will be consider for division, exponent is used for subtraction and sign bit is used to decide the magnitude of the sign whether it is positive or negative.

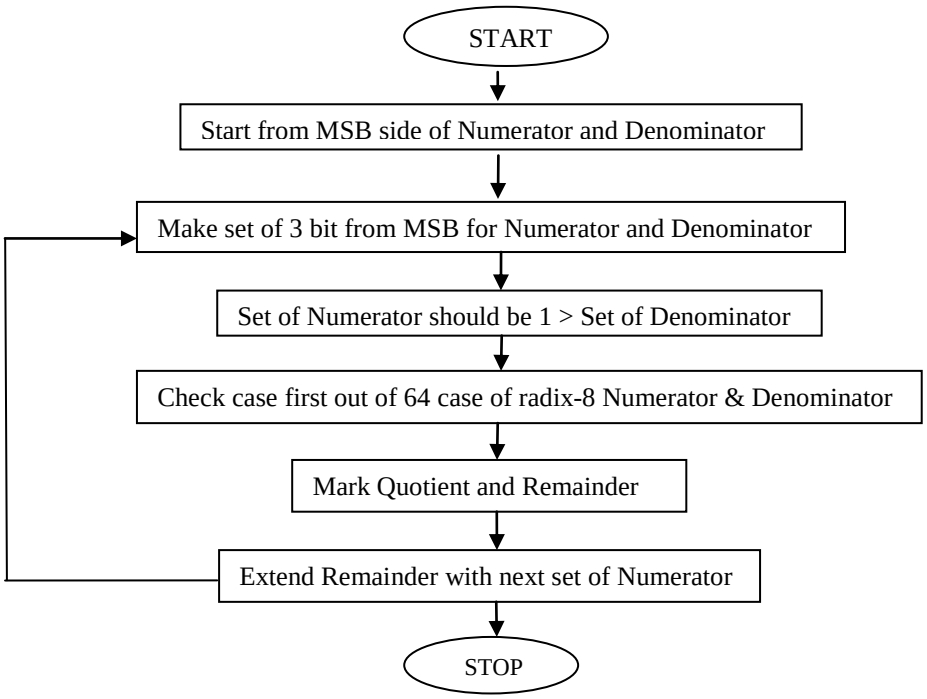

Fig.3: Radix-8 division Flowchart

Example of suggested division method by using of radix-8 will be considered for our simplicity. This division is performed in octal number system as it Radix-8 or base-8.

V. Design Of Module

Before we switch on 64-bit floating point division first we will try to simulate 32 bit division This section contains a experimental result of 32-bit floating point division is to be performed that is based on Radix-8 methodology.In which sign bit, 8 bit exponent and 23 bit normalized mantissa with 127 bias values will be considered. IEEE 754 standard are used with exception like an exponent of all zeroes will always mean zero and an exponent of all ones will always mean infinity. Also non normalized number is not implemented. Radix-8

methodology is to be used for faster speed. The total CPU completion time will be 0.41secs.and memory usages are 324788kb. Various basic modules like XOR gate, subtractor was shown in VHDL. Complete design was synthesized using Xilinx ISE 12.4. Fig. 4 shows RTL Schematic of radix-8 based floating point division.

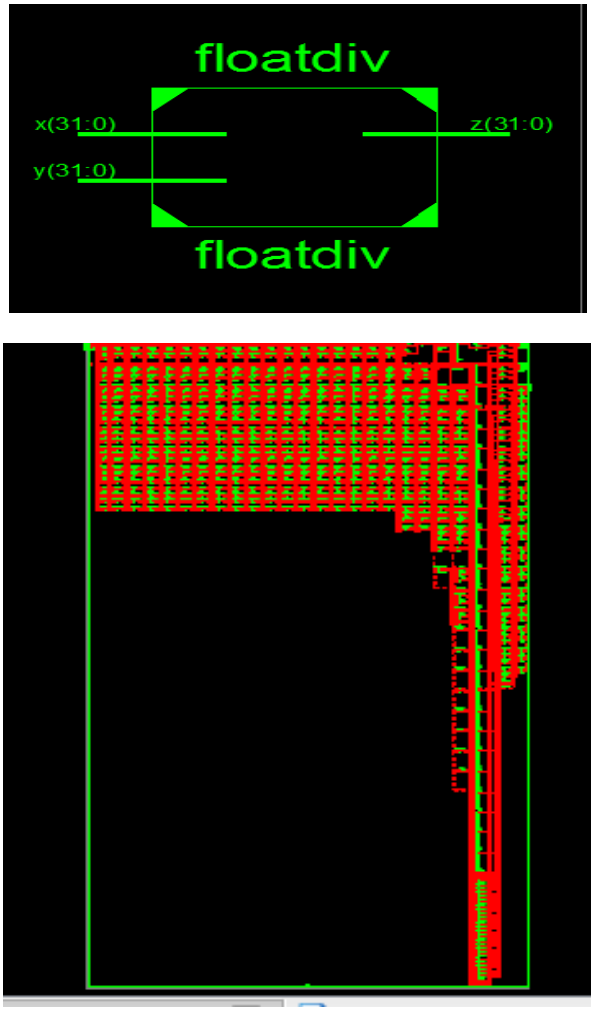

Figure 4: RTL Schematic of 32 bit Radix-8 base floating point division

The above mentioned modules are simulated and simulation result are shown in fig. 5 in which two random 32 bit floating point input number are consider and we get a 32 bit output in single clock cycle.

For example take as input as

X= 32'b 01000011000000000000000000000000

Y=32'b 01000000010000000000000000000000

Then output we get,

Z=32'b 01000010010011001100110011001101

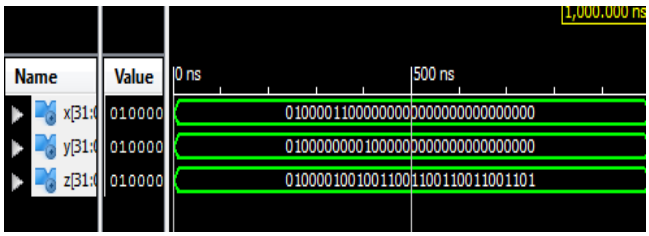

Figure 5: Simulation result of Radix-8 based floating point division

Table 1. Speed and memory consumption between previous design and Radix-8 based 32 bit floating point division

Resource design	Previous design	Radix-8 based
Speed	77.64Secs	44.36Secs
Memory usage	239872Kb	324788Kb

VI. Future Work

This complete system can be designed on Field programmable gate array (FPGA). The reference clock will be the clock present on the FPGA board. This adds great portability and the complete device will be less weighted. We can extend our division for 128 bit that is quad precision in future.

VII. Conclusion

In this paper, we have presented Double precision floating point division by using Radix-8 methodology, which will provide faster speed. Also we have design a 32-bit division and its simulation result. This complete design methodology provides 33% faster speed as compare to Radix-4 division with nearly equal power consumption.

References

[1]. K.Scott Hemmert and Keith D.Underwood "Floating-Point Divider Design for FPGAs" IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol.15, No.1, January 2007.
[2]. ANSI/IEEE Standard 754-1985, IEEE Standard for Binary Floating-Point Arithmatic 1985.
[3]. Jitendra Soni, Ravi Mohan. ECE, SRIT, Jabalpur, India" Floating Point Single Precision Division in VHDL Environment" International Journal of Emerging Trends in Electronics and Computer Science (IJETECS) Volume 2,Issue 10,Oct.2013.
[4]. Riya Saini, Galani Tina G. and R.D. Daruwala" Efficient Implementation of Pipelined Double Precision Floating Point Unit on FPGA" International Journal of Emerging Trends in Electrical and Electronics (IJETEE – ISSN: 2320-9569) Vol. 5, Issue. 1, July-2013.
[5]. Nicolas Brisebarre, Jean-Michel Muller, Member, IEEE, and Saurabh Kumar Raina "Accelerating Correctly Rounded Floating-Point Division when the Divisor Is Known in Advance" IEEE Transactions on Computers, Vol. 53, No. 8, August 2004.
[6]. Shamna.K1 and S.R Ramesh, M-Tech VLSI Design, 2Assistant. Professor Dept. of Electronics and Communication Engineering Amrita Vishwa Vidyapeetham, Coimbatore, India" Design and Implementation of an Optimized Double Precision Floating Point Divider on FPGA" International Journal of Advanced Science and Technology, Vol. 18, May, 2010
[7]. Xin Fang and Miriam Leeser Dept of Electrical and Computer Engg Northeastern University Boston, Massachusetts 02115 "Vendor Agnostic, High Performance, Double Precision Floating Point Division for FPGAs" 978-1-4799-1365-7/13/$31.00 ©2013