International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

A Review: Design and Simulation of Binary Floating Point Multiplier Using VHDL

Ujjwala V. Chaudhari[1], Prof A. P. Dhande[2]

[1]Student, Electronics and Telecommunication Engg,Sant Gadge Baba University Amravati, India

[2]Assistant Professor, Electronics and Telecommunication Engg, Sant Gadge Baba University Amravati, India

Abstract: ***Most of the DSP applications need floating point numbers multiplication. The possible ways to represent real numbers in binary format floating point numbers are; the IEEE 754 standard represents two floating point formats, Binary interchange format and Decimal interchange format. To improve speed multiplication of mantissa is done using specific multiplier replacing Carry Save Multiplier. To give more precision, rounding is not implemented for mantissa multiplication. The binary floating point multiplier is plane to do implemented using VHDL and it is simulated and synthesized by using ModelSim and Xilinx ISE software respectively. The result so got will be compare with the previous work done. Floating point multiplication is important in many commercial applications including financial analysis, banking, tax calculation, currency conversion, insurance, and accounting.***

Keywords: floating point, ModelSim, Xilinx ISE, Binary interchange format, Decimal interchange format.

1. Introduction

Floating point numbers are one possible way of representing real numbers in binary format, the IEEE 754 standard presents two different floating point formats, Binary interchange format and Decimal interchange format. Multiplying floating point numbers is a critical requirement for DSP applications involving large dynamic range. This paper focuses only on single precision normalized binary interchange format. It consists of a one bit sign (S), an eight bit exponent (E), and a twenty three bit fraction (M or Mantissa).

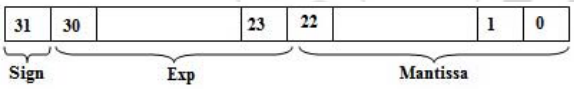


Figure 1: IEEE single precision floating point format

Z = (-1S) * 2 (E - Bias) * (1.M)
Bias = 127.

An extra bit is added to the fraction to form what is called the significant. If the exponent is greater than 0 and smaller than 255, and there is 1 in the MSB of the significant then the number is said to be a normalized number. Multiplying two numbers in floating point format is done by adding the exponent of the two numbers then subtracting the bias from their result ,and multiplying the significant of the two numbers, and calculating the sign by XORing the sign of the two numbers. The multiplier was verified against Xilinx floating point multiplier. In this paper representation of floating point multiplier in such a way that rounding support isn't implemented, thus accommodating more precision if the multiplier is connected directly to an adder in a MAC unit. Exponents addition, Significant multiplication, and Results sign calculation are independent and are done in parallel Xilinx ISE Design Suite 13.3 tool & VHDL programming is used. ISIM tool is used for Simulation process .Xilinx core generator tool is used to generate Xilinx floating point multiplier core The whole multiplier (top unit) was simulated against the Xilinx floating point multiplier core generated by Xilinx core generator.

A Binary multiplier is an integral part of the arithmetic logic unit (ALU) subsystem found in many processors. Integer multiplication can be inefficient and costly, in time and hardware, depending on the representation of signed numbers. Both's algorithm and others like Wallace-Tree suggest techniques for multiplying signed numbers that works equally well for both negative and positive multipliers. In this project, we have used VHDL as a HDL and Mentor Graphics Tools (MODEL-SIM & Leonardo Spectrum) for describing and verifying a hardware design based on Both's and some other efficient algorithms. Timing and correctness properties were verified. Instead of writing Test- Benches & Test-Cases we used Wave-Form Analyzer which can give a better understanding of Signals & variables and also proved a good choice for simulation of design. Hardware Implementations and synthesizability has been checked by Leonardo Spectrum and Precision Synthesis.

2. Literature Review

From the review of related work and published literature , it is observed that many researchers have design floating point multiplier by applying different techniques. Researchers have undertaken different systems , processes or phenomena with regards to design floating point multiplier and attempted to find the unknown parameters.

From the study of A High Speed Binary Floating Point Multiplier Using Dadda algorithm by B. Jeevan, it is observed that To improve speed multiplication of mantissa is done using Dadda multiplier replacing Carry Save Multiplier. The design achieves high speed with maximum frequency of 526 MHz compared to existing floating point multipliers. The floating point multiplier is developed to handle the underflow and overflow cases. To give more precision, rounding is not implemented for mantissa multiplication. The multiplier is implemented using Verilog

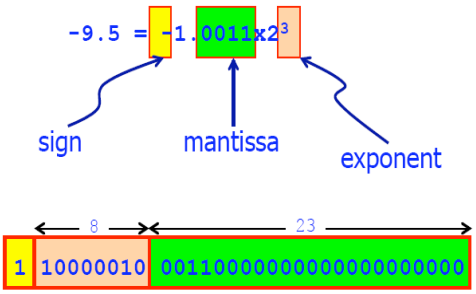
HDL and it is targeted for Xilinx. The multiplier is compared with Xilinx floating point multiplier core.

3. Proposed Work

Although computer arithmetic is sometimes viewed as a specialized part of CPU design, still the discrete component designing is also a very important aspect. A tremendous variety of algorithms have been proposed for use in floating-point systems. Actual implementations are usually based on refinements and variations of the few basic algorithms presented here.

Our discussion of floating point will focus almost exclusively on the IEEE floating-point standard (IEEE 754) because of its rapidly increasing acceptance. Although floating-point arithmetic involves manipulating exponents and shifting fractions, the bulk of the time in floating-point operations is spent operating on fractions using integer algorithms. Thus, after our discussion of floating point, we will take a more detailed look at efficient algorithms and architectures.

3.1 An Overview Of the IEEE FP Format



- The number, in binary, must be normalized: the integer part must always be equal to 1
- The exponent, an integer value, is not represented in 2-complement, but in a biased representation: a bias of 127 is added to the exponent.
- As the value 0 can not be normalized, a special representation is reserved for: all bits to zero
- In general, the values 00000000 and 11111111 from the exponent field are reserved for special cases and are not biased values:

3.2 Floating point multiplication of two numbers is made in four steps:

Step 1. Exponents of the two numbers are added directly,extra bias is subtracted from the exponent result.

Step 2. Significands multiplication of the two numbers using Dadda algorithm.

Step 3. To find the sign of result, XOR operation is done among sign bit of two numbers.

Step 4.Finally the result is normalized such that there should be 1 in the MSB of the result (leading one).

3.3 Proposed multiplier

3.3.1 Dadda Multiplier:
Dadda proposed a sequence of matrix heights that are predetermined to give the minimum number of reduction stages. To reduce the N by N partial product matrix, dada multiplier develops a sequence of matrix heights that are found by working back from the final two-row matrix. In order to realize the minimum number of reduction stages, the height of each intermediate matrix is limited to the least integer that is no more than 1.5 times the height of its successor.The process of reduction for a dadda multiplier is developed using the following recursive algorithm:
1. Let d1=2 and dj+1 = [1.5*dj], where dj is the matrix height for the jth stage from the end. Find the smallest j such that at least one column of the original partial product matrix has more than dj bits.
2. In the jth stage from the end, employ (3, 2) and (2, 2) counter to obtain a reduced matrix with no more than dj bits in any column.
3. Let j = j-1 and repeat step 2 until a matrix with only two rows is generated. This method of reduction, because it attempts to compress each column, is called a column compression technique. Another advantage of utilizing Dadda multipliers is that it utilizes the minimum number of (3, 2) counters. {Therefore,the number of intermediate stages is set in terms of lower bounds: 2, 3, 4, 6, 9 . . .For Dadda multipliers there are N2 bits in the original partial product matrix and 4.N-3 bits in the final two row matrix. Since each (3, 2) counter takes three inputs and produces two outputs, the number of bits in the matrix is
reduced by one with each applied (3, 2) counter therefore}, the total number of (3,2) counters is #(3, 2) = N2 – 4.N+3 the length of the carry propagation adder is CPA length = 2.N–2.

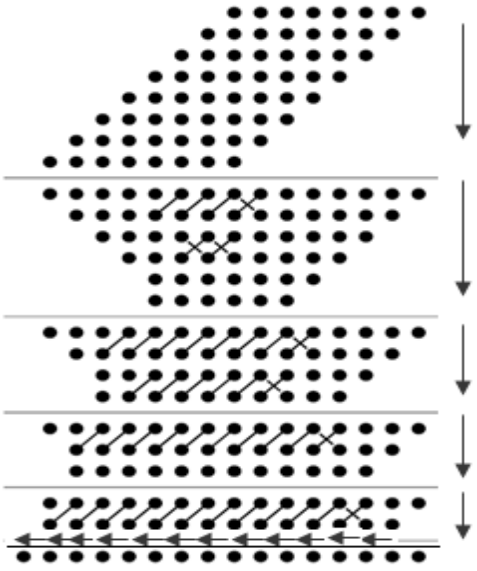


Figure: Dot diagram for 8 by 8 Dadda Multiplier

The number of (2, 2) counters used in Dadda's reduction method equals N-1.The calculation diagram for an 8X8 Dadda multiplier is shown in figure.Dot diagrams are useful tool for predicting the placement of (3, 2) and (2, 2) counter in parallel multipliers. Each IR bit is represented by a dot.The output of each (3, 2) and (2, 2) counter are represented as two dots connected by a plain diagonal line.

The outputs of each (2, 2) counter are represented as two dots connected by a crossed diagonal line.The 8 by 8 multiplier takes 4 reduction stages, with matrix height 6, 4, 3 and 2. The reduction uses 35 (3, 2) counters, 7(2, 2) counters, reduction uses 35 (3, 2) counters, 7 (2, 2)counters, and a 14-bit carry propagate adder. The total delay for the generation of the final product is the sum of one ANDgate delay, one (3, 2) counter delay for each of the four reduction stages, and the delay through the final 14-bit carry propagate adder arrive later, which effectively reduces the worst case delay of carry propagate adder. The decimal point is between bits 45 and 46 in the significand IR. Critical path is used to determine the time. taken by the Dadda multiplier. The critical path starts at the AND gate of the first partial products passes through the full adder of the each stage, then passes through all the vector merging adders.The stages are less in this multiplier compared to the carry save multiplier and therefore it has high speed than that.

4. Objectives

1) To study Dadda algorithm
2) To study the hardware language VHDL
3) 3.To design and implement the various technique about Floating Point Multiplier
4) Verify the functionality of Floating Point Multiplier
5) Analyze the design for FPGA design utilization summary, propogation delay and maximum operating frequency of design.

5. FPGA

FPGA or Field Programmable Gate Arrays can be programmed or configured by the user or designer after manufacturing and during implementation. Hence they are otherwise known as On-Site programmable. Unlike a Programmable Array Logic (PAL) or other programmable device, their structure is similar to that of a gate-array or an ASIC. Thus, they are used to rapidly prototype ASICs, or as a substitute for places where an ASIC will eventually be used [17]. This is done when it is important to get the design to the market first. Later on, when the ASIC is produced in bulk to reduce the NRE cost, it can replace the FPGA. The programming of the FPGA is done using a logic circuit diagram or a source code using a Hardware Description Language (HDL) to specify how the chip should work. FPGAs have programmable logic components called ‚logic blocks', and a hierarchy or reconfigurable interconnects which facilitate the ‚wiring' of the blocks together. The programmable logic blocks are called configurable logic blocks and reconfigurable interconnects are called switch boxes. Logic blocks (CLBs) can be programmed to perform complex combinational functions, or simple logic gates like AND and XOR. In most FPGAs the logic blocks also include memory elements, which can be as simple as a flip-flop or as complex as complete blocks of memory.

6. VHDL

The VHSIC (very high speed integrated circuits) Hardware Description Language (VHDL) was first proposed in 1981. The development of VHDL was originated by IBM, Texas Instruments, and Inter-metrics in 1983. The result, contributed by many participating EDA (Electronics Design Automation) groups, was adopted as the IEEE 1076 standard in December 1987. VHDL is intended to provide a tool that can be used by the digital systems community to distribute their designs in a standard format. Using VHDL, they are able to talk to each other about their complex digital circuits in a common language without difficulties of revealing technical details. As a standard description of digital systems, VHDL is used as input and output to various simulation, synthesis, and layout tools. The language provides the ability to describe systems, networks, and components at a very high behavioral level as well as very low gate level. It also represents a top-down methodology and environment. Simulations can be carried out at any level from a generally functional analysis to a very detailed gate-level wave form analysis.

References

[1] Remadevi R / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 3, Issue 2, March -April 2013, pp.283-286 283 “Design and Simulation of Floating Point Multiplier Based on VHDL”
[2] International Journal of Engineering Research and Development e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com Volume 10, Issue 3 (March 2014), PP.73-78 73 “Design of Floating Point Multiplier Using Vhdl” P.Gayatri(Department of Electronics & Communication Engineering, Lendi Institute of Engineering and Technology/JNTUK, India).
[3] L. Louca, T. A. Cook, and W. H. Johnson, “Implementation of IEEESingle Precision Floating Point Addition and Multiplication on FPGAs,”Proceedings of 83 the IEEE Symposium on FPGAs for CustomComputing Machines (FCCM’96), pp. 107–116, 1996.
[4] A. Jaenicke and W. Luk, "Parameterized Floating-Point Arithmetic on FPGAs", Proc. of IEEE ICASSP, 2001, vol. 2, pp.897-900.
[5] B. Lee and N. Burgess, “Parameterisable Floating-point Operations on FPGA,” Conference Record of the Thirty-Sixth Asilomar Conference onSignals, Systems, and Computers, 2002.
[6] Mohamed Al-Ashrafy, Ashraf Salem and Wagdy Anis” An Efficient Implementation of Floating PointMultiplier” Electronics, Communications and Photonics Conference (SIECPC), 2011 Saudi International.

Author Profile

Ujjwala V. Chaudhari[1] received her BE degree in Electronics and Telecommunication from Sant Gadge Baba Amravati University 2013. Currently pursuing ME degree in Electronics and Telecommunication from Sant Gadge Baba Amravati University,India

Prof A.P.hande[2] received his BE degree in Electronics and Telecommunication from Sant Gadge Baba Amravati University 2009 and ME in Electronics and Telecommunication from Sant Gadge Baba Amravati University in .Currently working as Assistant Professor in P.R.Patil COET, Amravati, India