International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 3, Issue 6, June 2014 ISSN 2319 - 4847

Design and Implementation of Floating Point Multiplier Using Wallace and Dadda Algorithm

Chaitali V. Matey [1], Dr. S. D. Chede[2] Prof. S. M. Sakhare[3]

[1] Student M. Tech, Department of Electronics, Suresh Deshmukh College of Engineering Selukate Wardha
[2]Principal, Om College of Engineering Wardha
[3] Assistant Professor, Department of ETRX Engg., Suresh Deshmukh College of Engineering Selukate Wardha

Abstract

In computing, floating point describes a method of representing an approximation of a real number in a way that can support a wide range of values. Low power consumption and smaller area are some of the most important criteria for the fabrication of DSP systems and high performance systems. Optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. This paper presents a floating point multiplier using Wallace and Dadda Algorithm of an IEEE 754 single precision floating point multiplier targeted for Xilinx. Improvement in speed multiplication of Dadda and Wallace multiplier is done thereby replacing Look ahead Carry adder. The methodology uses Exponent Calculator, Mantissa multiplier, Sign Calculator, and Normalization unit.

Keywords: Dadda Algorithm, Wallace Algorithm, Floating point, multiplication, Spartan 6, ISE 13.1, VHDL language.

1. INTRODUCTION

Single-precision binary floating-point is used due to its wider range over fixed point (of the same bit-width), even if at the cost of precision. Our discussion of floating point will focus almost exclusively on the IEEE floating-point standard (IEEE 754) because of its rapidly increasing acceptance. Multiplying floating point numbers is a critical requirement for DSP applications. The possible ways to represent real numbers in binary format floating point numbers are; the IEEE 754 standard [1] represents two floating point formats, Binary interchange format and Decimal interchange format. This paper focuses only on single precision normalized binary interchange format. Representation of single precision normalized binary interchange format is shown in Fig.1. It consists of a one bit sign (S), an eight bit exponent (E), and a twenty three bit fraction (M or Mantissa).

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

Where M = n22 2-1 + n21 2-2 + n20 2-3+…+ n1 2-22+ n0 2-23;

Bias = 127.

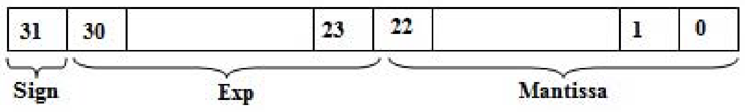


Figure 1 IEEE Single Precision Floating Point Format

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

Step1. 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 & Wallace algorithm.

Step 3. Calculating the sign by XORing the sign of the two numbers.

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

Mohamed Al-Ashrfy, Ashraf Salem and Wagdy Anis implementation which handles the overflow and underflow cases. Rounding is not implemented to give more precision when using the multiplier in a multiply and Accumulate (MAC) unit. And an implementation of a floating point multiplier that supports the IEEE 754-2008 binary interchange format. The multiplier doesn't implement rounding and just presents the significant multiplication result as is (48 bits).

In 2013, B. Jeevan, et al, shows a high speed binary floating point multiplier based on Dadda Algorithm. In this improvement in speed of multiplication of mantissa is done using Dadda multiplier thereby 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. The significant multiplication time is reduced by using Dadda Algorithm.

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.
Fig. shows 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 . . .

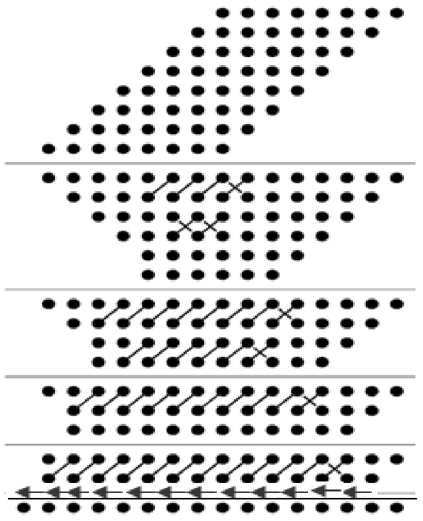
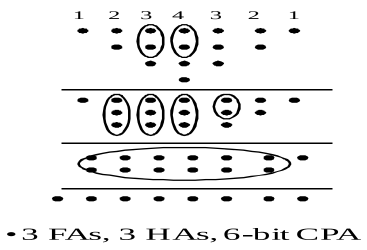
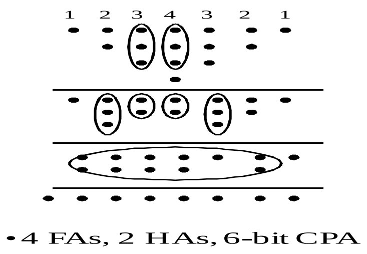


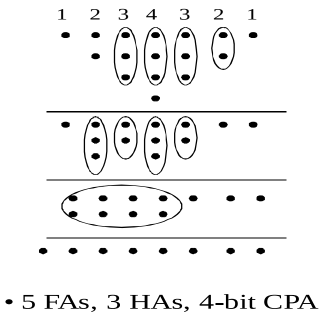
Figure 2 Dot diagram for 8 by 8 Dadda Multiplier

DADDA MULTIPLIER EXAMPLE





WALLCE MULTIPLIER EXAMPLE



2. Floating Point Multiplier Algorithm

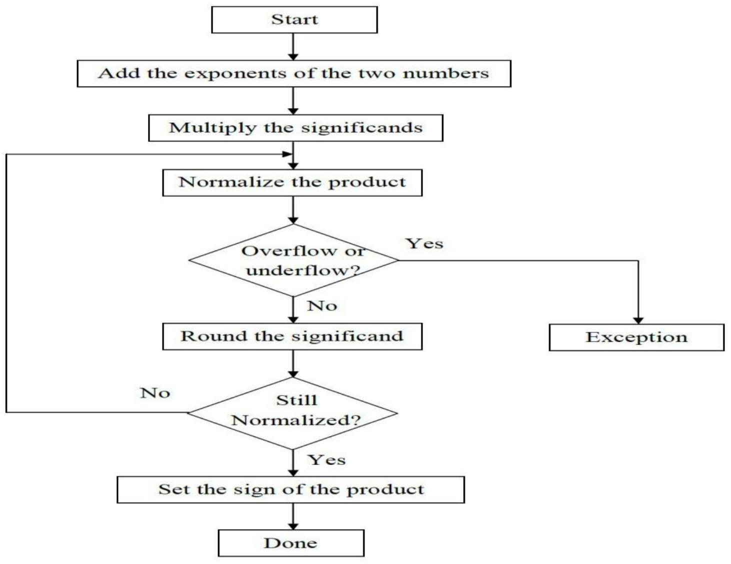


Figure 3 Floating Point Algorithm

As shown in Figure 3 the Floating Point Algorithm.The normalized floating point numbers have the form of Z= (-1S) * 2 (E - *Bias*) * (1.M). The following algorithm is used to multiply two floating point numbers.

1. Significand multiplication; i.e. (1.M1*1.M2).
2. Placing the decimal point in the result.
3. Exponent's addition; i.e. (E1 + E2 - Bias).
4. Getting the sign; i.e. s1 xor s2.
5. Normalizing the result; i.e. obtaining 1 at the MSB of the results' significand.
6. Rounding implementation.
7. Verifying for underflow/overflow occurrence.

3. Proposed Methodology

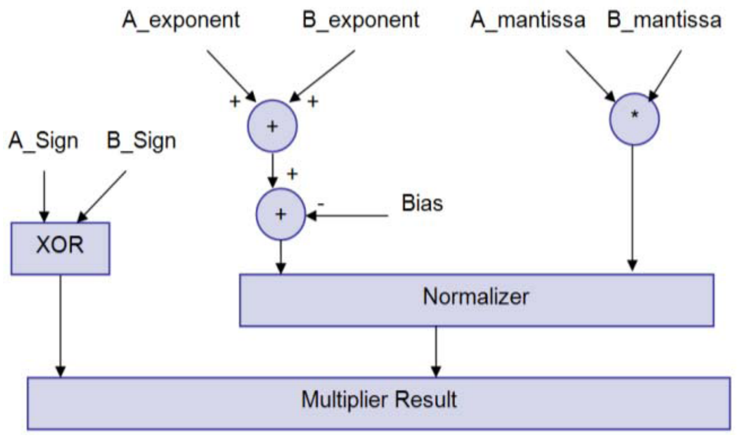


Figure 4 Floating point multiplier block diagram

Above Figure shows Floating point multiplier block diagram.

3.1 Exponent

The exponent field represents the exponent as a biased number. It contains the actual exponent plus 127 for single precision, or the actual exponent plus 1023 in double precision.This converts all single precision exponents from -127 to 127 into unsigned numbers from 0 to 254, and all double precision exponents from -1023 to 1023 into unsigned numbers from 0 to 2046. Two Examples shown below for single precision If the exponent is 4, the e-field will be 4+127=132 (100000112). If the e-field contains 8'b01011101(9310) the actual exponent is 93-127 = 34 Storing a biased exponent means we can compare IEEE values as if they were signed integers.

3.2 Mantissa

The field f contains a binary fraction. The actual mantissa of the floating-point value is (1 + f). In other words, there is an implicit 1 to the left of the binary point. For example, if f is 01101…, the mantissa would be 1.01101…There are many ways to write a number in scientific notation, but there is always a unique normalized representation, with exactly one non-zero digit to the left of the point. 0.232 * 103 = 23.2 * 101 = 2.32 * 102 = …A side effect is that we get a little more precision: there are 24 bits in the mantissa, but we only need to store 23 of them.

3.3 Sign

The sign bit is 0 for positive numbers and 1 for negative numbers. But unlike integers, IEEE values are stored in signed magnitude format.

3.4 Normalizing

The result of the significant multiplication (intermediate product) must be normalizing. Having a leading ‘1’ just immediate to the left of the decimal point is known as a normalized number.

4. METHODOLOGY USED

Design of Floating point multiplier is done by using VHDL in previous last years. All the available design uses carry save adder or ripple carry adder for design of floating point multiplier. Also different algorithms are available for the design. Carry look ahead adder is one of the fastest adder and having more advantages among all the available adders. So our aim is to design and implement floating point multiplier using Wallace and Dadda algorithm with carry look ahead adder.

5. RESULT AND CONCLUSIONS

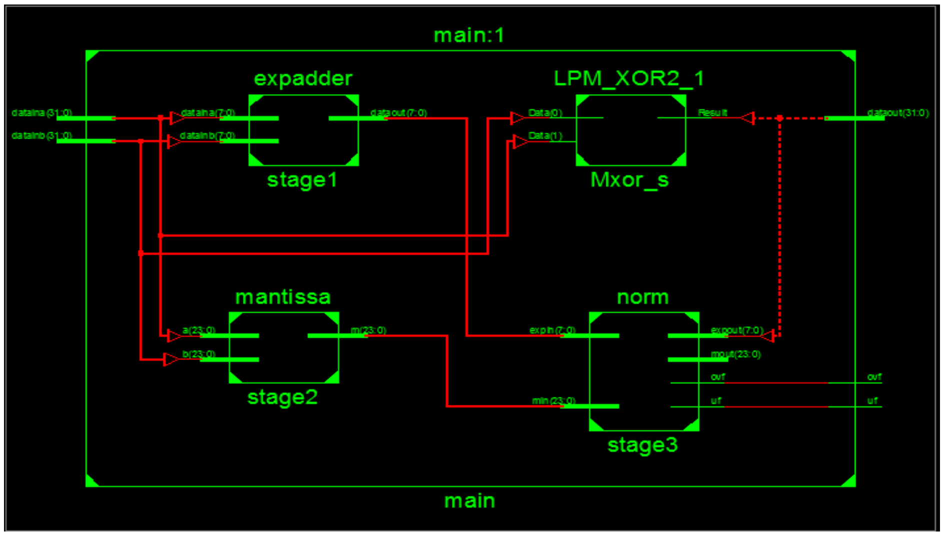


Figure 5 RTL view of Dadda Multiplier

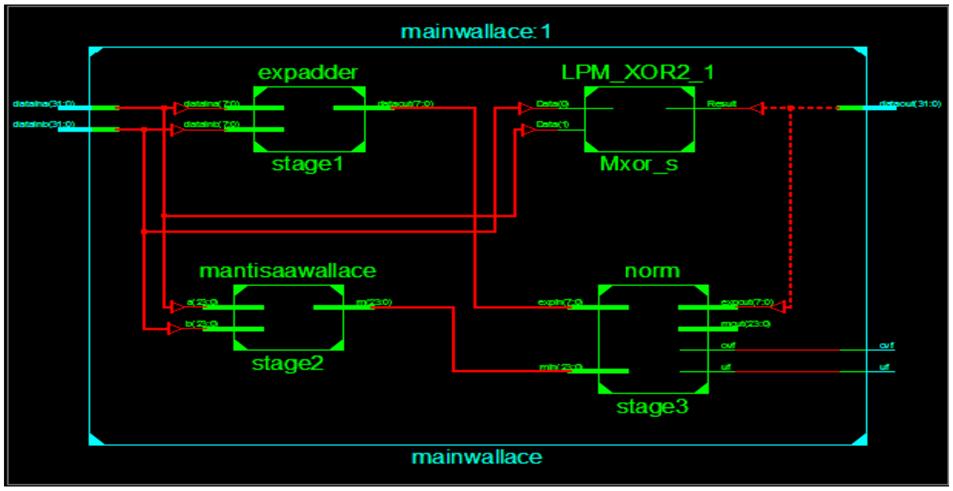


Figure 6 RTL view of Wallace Multiplier

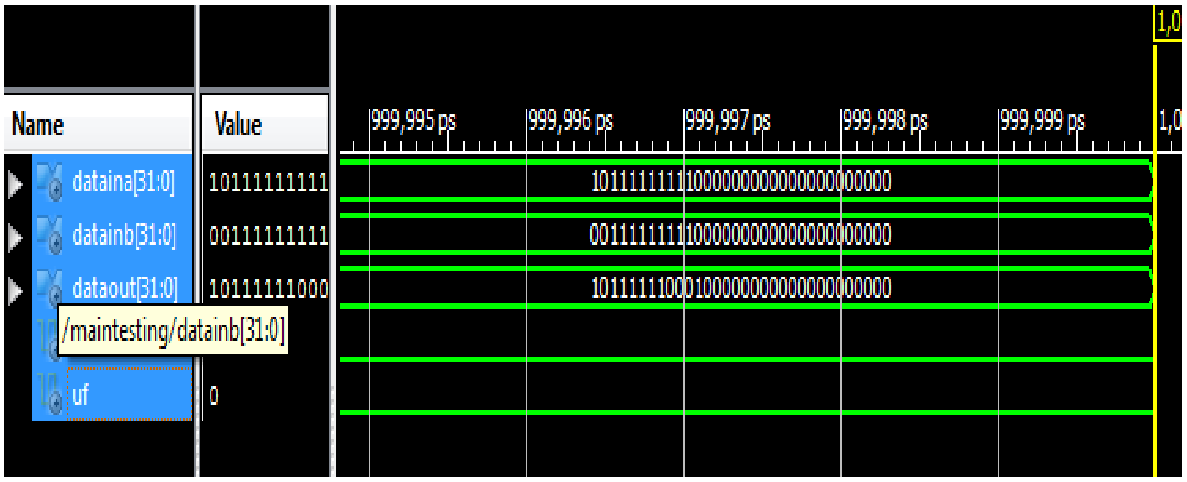


Figure 7 Simulation waveform of Dadda multiplier

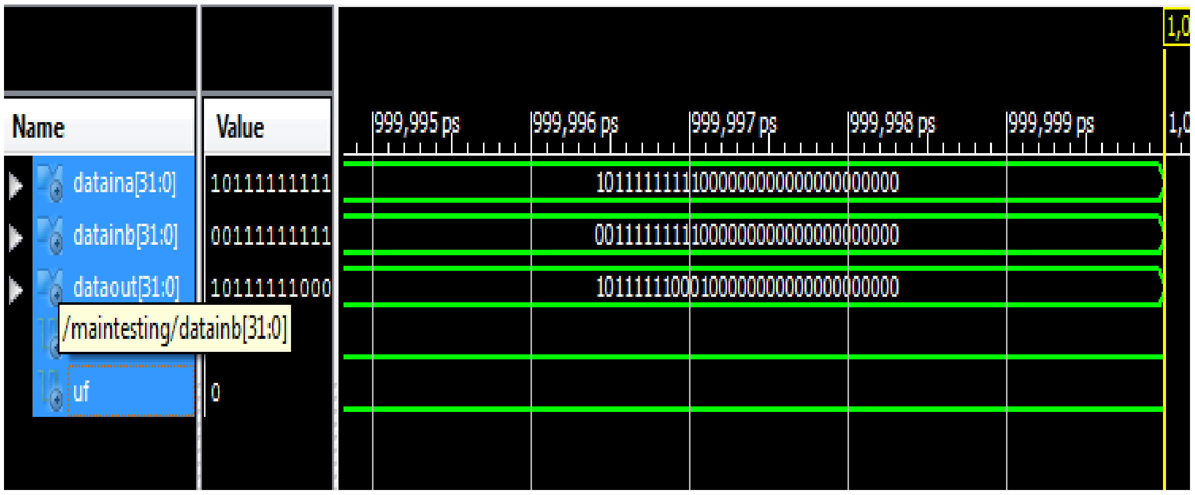


Figure 8 Simulation waveform of Wallace multiplier

Fig.6 and Fig.7 shows the RTL view of Dadda multiplier and Wallace multiplier. In both, There are used 4 stages i.e. Expadder, Mantissa, Sign and Normalizer. Thus, the exponent of single precision normalized binary interchange format is designed. Fig. 8 and Fig. 9 shows the Simulation waveform of Dadda and Wallace multiplier, wherein the multiply of two 32 bit data inputs (dataina & datainb) are performed using Dadda and Wallace multiplier with carry look ahead adder. The two inputs are a & b which produces the output (dataout) given below.

i.e. a = 10111111111000000000000000000000
b = 00111111111000000000000000000000
Result=10111111000000000000000000000000

Table 1: Area and delay comparison between the implemented floating point multiplier

Parameters	Our Implementations	
	Dadda Multiplier	Wallace Multiplier
No. of Slices	432	431
No. of Flip Flops	2	2
No. of i/p LUTs	755	754
IOBs	96	96
Delay	15.607nsec	17.495nsec

In this way, Design and implementation of floating point multiplier using Wallace and Dadda multiplier with carry look ahead adder is realized and Dadda multiplier have less delay than Wallace multiplier. The design has been implemented on a Spartan6 and achieved less delay of 15.607nsec.

References

[1] Mohamed Al-Ashrfy, Ashraf Salem and Wagdy Anis "An Efficient implementation of Floating Point Multiplier" IEEE Transaction on VLSI 978-1-4577-0069-9/11@2011 IEEE, Mentor Graphics.

[2] Jaenicke and W. Luk, "Parameterized Floating-Point Arithmetic on FPGAs", Proc. of IEEE ICASSP, 2001, vol. 2, pp.897-900.

[3] B. Jeevan, et al, "A High Speed Binary Floating Point Multiplier using Dadda Algorithm", in IEEE International Multi Conference on

[4] B.Fagin and C.Renard, "Field Programmable Gate Arrays and Floating Point Arithmetic," IEEE Transactions on VLSI, vol. 2, no.3, pp. 365- 367, 1994.

[5] N. Shirazi, A.Walters, and P. Athanas, "Quantitative Analysis of Floating Point Arithmetic on FPGA Based Custom Computing Machines," Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM'95), pp.155-162, 1995.

[6] Pradeep Sharma, et al, "Analyzing Single Precision Floating Point Multiplier on Virtex 2P Hardware Module", in International Journal of Engineering Research and Applications, vol 2, no. 5, 2012, pp2016 – 2020.

[7] Amaricai A., Vladutiu M., Udrescu M., Prodan L. and Boncalo O., "Floating Point Multiplication Rounding Schemes for Interval Arithmetic", pp. 19-24, 2008.

[8] Awan M.A., Siddiqui M.R., "Resolving IEEE Floating-Point Error using Precision-Based Rounding Algorithm", pp. 329-333, 2005.

[9] IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic, 2008.

[10] L. Louca, T. A. Cook, and W. H. Johnson, "Implementation of IEEE Single Precision Floating Point Addition and Multiplication on FPGAs," Proceedings of 83 the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM'96), pp. 107-116, 1996.

AUTHOR

Miss Chaitali V. Matey received the bachelor degree B.E in Electronics and Communication Engg. From R.T.M.N.U Nagpur university in the year 2012.Currently she is working as Research scholer for pursuing M.Tech in Electronics (Communication Engineering) at SDCE, Selukate, Wardha, Maharashtra, India.

Dr.Santosh D. Chede received the bachelor degree B.E in Industrial Electronics from Amravati university in the year 1990.Also he has received his master's degree M.E in electronics Engg. From Amravati university in the year 2000.In year 2010 he received his Doctor of Philosophy from VNIT,Nagpur.Currently he is working as Principal, Om college of Engg, Inzapur, Wardha, Maharashtra,India

Prof. Shailesh M. Sakhare received the bachelor of Engg. Degree B.E in Electronics Engg. From R.T.M Nagpur university in 2008.Also he received his master's degree master of Technology in Electronics Engg in 2012 from GHRCE, Nagpur,An Autonomous Institute . Currently he is working as Asst.Professor in Electronics Dept. at Suresh Deshmukh College of Engineering , Selukate,Wardha, Maharashtra,India.