ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

A High Speed Binary Floating Point Multiplier Using Dadda Algorithm

J.Swathi
M.Tech Student,
Department of ECE,
KITS for Women's, kodad, T.S, India.

Mr. B. Naresh Reddy
Associate Professor,
Department of ECE,
KITS for Women's, kodad, T.S, India.

ABSTRACT:

Abstract: In this paper area efficient Multiplier architecture is developed using Dadda Multiplier. The proposed Multiplier Algorithm takes reduced area than the previous one and the significant delay is also lower than the previous designs. The number of slices in the previous designs is 648 and in our proposed Dadda Multiplier architecture utilizes only 402 slices then area is reduced up to 30%. As shown in the design as well as the simulation results the proposed Multiplier architecture area as well as delay is better.

Keywords:

Double Precision, Dadda Multiplier, Floating Point, Area Efficient.

1. INTRODUCTION:

Digital arithmetic operations are very important in the design of digital processors and application-specific systems. Arithmetic circuits form an important class of circuits in digital systems. With the remarkable progress in the very large scale integration (VLSI) circuit technology, many complex circuits, unthinkable yesterday have become easily realizable today. Algorithms that seemed impossible to implement now have attractive implementation possibilities for the future. This means that not only the conventional computer arithmetic methods, but also the unconventional ones are worth investigation in new designs. The notion of real numbers in mathematics is convenient for hand computations and formula manipulations.

However, real numbers are not well-suited for general purpose computation, because their numeric representation as a string of digits expressed in, say, base 10 can be very long or even infinitely long. Examples include π, e, and 1/3. In practice, computers store numbers with finite precision. Numbers and arithmetic used in scientific computation should meet a few general criteria:- Numbers should have modest storage requirements.

- Arithmetic operations should be efficient to carry.

• A level of standardization, or portability, is desirable–results obtained on one computer should closely match the results of the same computation on other computers Internationallystandardized methods for representing numbers on computers have been established by the IEEE-754 standard to satisfy these basic goals [1]. An arithmetic unit based on IEEE standard for floating point numbers has been implemented on FPGA Board. The arithmetic unit implemented has a 64-bit processing unit which allows various arithmetic operations such as, Addition, Subtraction, Multiplication, Division and Square Root on floating point numbers. Each operation can be selected by a particular operation code. Synthesis of the unit for the FPGA board has been done using XILINX-ISE.The IEEE standards mandate exact representations for binary single and double precision floating-point formats [4], as well as more flexible guidelines for single-extended and double-extended formats. Quadruple precision is not yet an official standard, although at present, an IEEE working group is standardizing it [12]. The IEEE standards have been extraordinarily successful in ensuring a level of portability for computer arithmetic across a vast array of implementations and disparate architectures. Since these standards are the basis for virtually all floating-point computation, it is important to understand their details.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

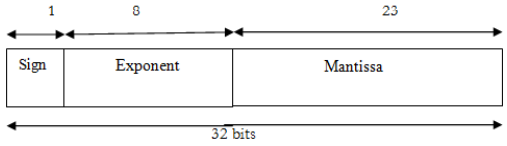


Fig 1: Single Precision Floating-Point IEEE Formats

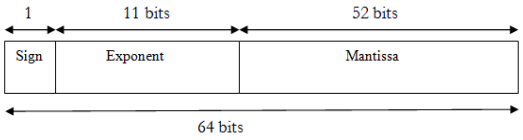


Fig 2: Double Precision Floating-Point IEEE Formats

Fig.1 and Fig.2 illustrates the IEEE standard binary single precision floating-point formats, along with the IEEE standard for double precision floating-point format. Single precision has 1 sign bit, 8 exponent bits, and 23 mantissa bits. Double precision has 1 sign bit, 11 exponent bits, and 52 mantissa bits. The IEEE format requires normalization, and since it uses radix 2, it is known a prior that the first bit of the mantissa is a 1, which means that it can be implied. This implied bit gives IEEE formats an extra bit of mantissa. For example, IEEE single precision has effectively 24 bits of mantissa, rather than the 23 which are expressed in the external representation as shown in Fig 1. Floating Point Numbers The term floating point is derived from the fact that there is no fixed number of digits before and after the decimal point, that is, the decimal point can float.

There are also representations in which the number of digits before and after the decimal point is set, called fixed-point representations. In general, floating point representations are slower and less accurate than fixed-point representations, but they can handle a larger range of numbers. Floating Point Numbers are numbers that can contain a fractional part. For e.g. following numbers are the floating point numbers: 3.0, - 111.5, ½, 3E-5 etc. Floating-point arithmetic is considered an esoteric subject by many people. This is rather surprising because floating-point is ubiquitous in computer systems. Almost every language has a floating-point data type; computers from PC's to supercomputers have floating-point accelerators; most compilers will be called upon to compile floating-point algorithms from time to time; and virtually every operating system must respond to floating-point exceptions such as overflow. A number representation (called a numeral system in mathematics) specifies some way of storing a number that may be encoded as a string of digits.

In computing, floating point describes a system for numerical representation in which a string of digits (or bits) represents a rational number. The term floating point refers to the fact that the radix point (decimal point, or, more commonly in computers, binary point) can "float"; that is, it can be placed anywhere relative to the significant digits of the number. This position is indicated separately in the internal representation, and floating-point representation can thus be thought of as a computer realization of scientific notation. Over the years, several different floating-point representations have been used in computers; however, for the last ten years the most commonly encountered representation is that defined by the IEEE 754 Standard. The advantage of floating-point representation over fixedpoint (and integer) representation is that it can support a much wider range of values. For example, a fixed point representation that has seven decimal digits, with the decimal point assumed to be positioned after the fifth digit, can represent the numbers 12345.67, 8765.43, 123.00, and so on, whereas a floating-point representation (such as the IEEE 754 decimal32 format) with seven decimal digits could in addition represent 1.234567, 123456.7, 0.00001234567, 1234567000000000, and so on. The floating-point format needs slightly more storage (to encode the position of the radix point), so when stored in the same space, floating-point numbers achieve.

II. FLOATING POINT MULTIPLIER ALGORITHM:

The normalized floating point numbers have the form of $Z= (-1\ S) * 2\ (E - Bias) * (1.M)$. 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.

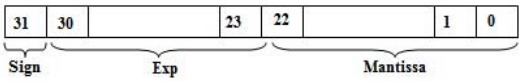
3. IMPLEMENTATION:

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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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



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 byworking back from the final two-row matrix.

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 byworking back from the final two-row matrix.

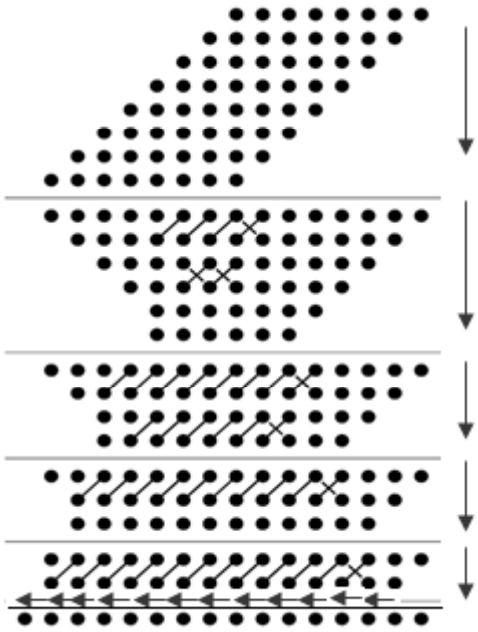
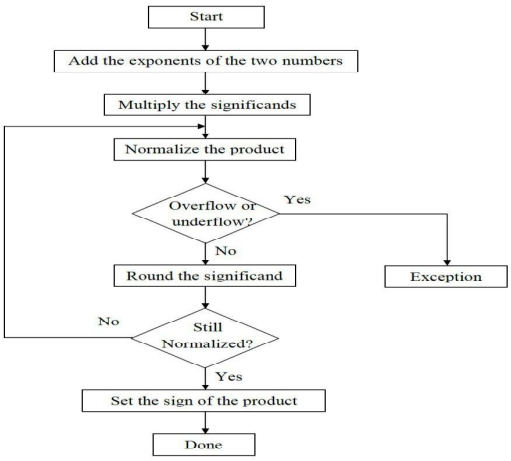


Figure 2 Dot diagram for 8 by 8 Dadda Multiplier

4. FLOATING POINT MULTIPLIER ALGORITHM:

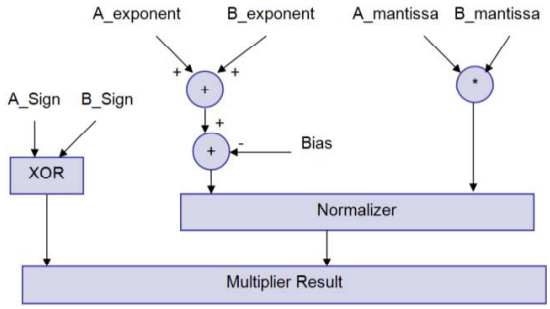


As shown in Figure 3 the Floating Point Algorithm.The normalized floating point numbers have the form of Z= (-1S) * 2 (E - Bias) * (1.M). The following algorithm is used to multiply two floating point numbers.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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.

PROPOSED METHODOLOGY:



Floating point multiplier block diagram

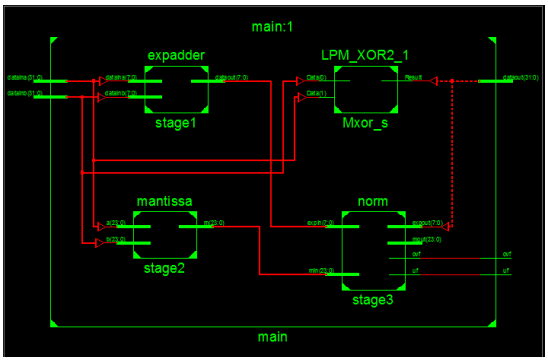
Exponent The exponent field represents the exponent as a biased number. It contains the actual exponent plus 127 for single precision, or the actual exponent plus 1023 in double precision.This converts all single precision exponents from -127 to 127 into unsigned numbers from 0 to 254, and all double 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. 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. Concluding Remarks :

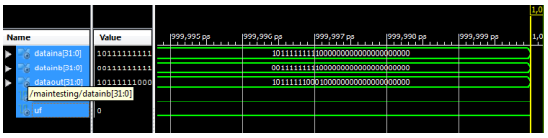
The Double Precision proposed Dadda Multiplier architecture is implemented on FPGA vertex board and the device utilization summary is shown in the previous section.

The architecture found area efficient as it utilizes only 402 slices against the 648 slices on the previous Multiplier design. The proposed Multiplier architecture is capable of calculating 64 bit numbers. In the future designs the adder architectures will help to reduce the device utilization to large extent because if the components of Multiplier architecture is efficient than the whole architecture will be definitely better in terms of delay as well as area.

6. Experimental Results:



RTL view of Dadda Multiplier



Simulation waveform of Dadda multiplier

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. 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 aheadadder. The two inputs are a & b which produces the output (dataout) given below.

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

ACKNOWLEDGMENTS:

I am J.Swathi and would like to thank the publishers, researchers for making their resources material available.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

I am greatly thankful to Assistant Prof: Miss.Ch.Nirmala for their guidance. We also thank the college authorities, PG coordinator and Principal for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

REFERENCES:

1] M. C. C¸ avu¸so˘glu. Telesurgery and Surgical Simulation: Design, Modeling, and Evaluation of Haptic Interfaces to Real and Virtual Surgical Environments. PhD thesis, University ofCalifornia, Berkeley, August 2000.

[2] M. C. C¸ avu¸so˘glu, F. Tendick, M. Cohn, and S. S. Sastry. A laparoscopic telesurgical workstation. IEEE Transactions on Robotics and Automation, 15(4):728–739, August 1999.

[3] E. Graves. Vital and Health Statistics. Data f rom the National Health Survey No. 122. U.S. Department of-Health and Human Services, Hyattsville, MD, 1993.

[4] J. W. Hill, P. S. Green, J. F. Jensen, Y. Gorfu, and A. S. Shah. Telepresence surgery demonstration system. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 2302–2307, 1994.

[5] A. J. Madhani. Design of Teleoperated Surgical Instruments for Minimally Invasive Surgery. PhD thesis, Massachusetts Institute ofTechnology, 1998.

[6] A. J. Madhani, G. Niemeyer, and J. K. Salisbury. The black falcon: a teleoperated surgical instrument for minimally invasive surgery. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98), volume 2, pages 936–944, 1998.

[7] J.W.Hill, P. S. Green, J. F. Jensen,Y. Gorfu, and A. S. Shah, "Telepresencesurgery demonstration system," in Proc. IEEE Int. Conf. Robot. Autom.,San Diego, CA, May 1994, vol. 3, pp. 2302–2307.

[8] P. Dario, E. Guglielmelli, B. Allotta, and M. C. Carrozza, "Robotics formedical applications," IEEE Robot. Autom.Mag., vol. 3, no. 3, pp. 44–56,Sep. 1996.

Author's Details:

Ms. J.Swathi. MTech student, in M.Tech Student, Dept of ECE in KITS for women's,kodad, T.S, India.

Mr. B. Naresh Reddy working as a Assistant at ECE in KITS for women's,kodad, T.S, IndiaJNTUH Hyderabad. he has 6 years of UG/PG Teaching Experience

.