ISSN: 2456-043X

International Journal of Engineering Science and Generic Research (IJESAR)
Available Online at www.ijesar.in
Volume 2; Issue 5; September-October; 2016; Page No. 56-59

# Design and implementation of Floating point multiplication with Karatsuba-Urdhva multiplier and kogge stone adder

**DONURU MADHAVI [1], V.VENKANNA [2]**

[1] PG Scholar, Department of ECE, Brilliant Institute of Engineering and Technology, Hyderabad, India

[2] Assistant Professor, Department of ECE, Brilliant Institute of Engineering and Technology, Hyderabad, India

donuru.madhavi@gmail.com

**Abstract**

Floating point multiplication is a crucial operation in high power computing applications such as image processing, signal processing etc. And also multiplication is the most time and power consuming operation. This paper proposes an efficient method for IEEE 754 floating point multiplication which gives a better implementation in terms of delay and power. A combination of Karatsuba algorithm and Urdhva-Tiryagbhyam algorithm (Vedic Mathematics) is used to implement unsigned binary multiplier for mantissa multiplication. The multiplier is implemented using Verilog HDL, targeted on Spartan-3E and Virtex-4 FPGA.

**Keywords:** FPGA, Floating point multiplier, Vedic mathematics, Urdhva-Tiryagbhyam, Karatsuba

## 1. Introduction

Floating point multiplication units are an essential IP for modern multimedia and high performance computing such as graphics acceleration, signal processing, image processing etc. There are lot of effort is made over the past few decades to improve performance of floating point computations. Floating point units are not only complex, but also require more area and hence more power consuming as compared to fixed point multipliers. And the complexity of the floating point unit increases as accuracy becomes a major issue. IEEE 754 support different floating point formats such as Single Precision format, Double Precision format, Quadruple Precision format etc. But as the precision increases, multiplier area, delay and power increases drastically. In the proposed paper, we present a new multiplication method which uses a combination of Karatsuba and Urdhva-Tiryagbhyam (Vedic Mathematics) algorithm for multiplication. This combination not only reduces delay, but also reduces the percentage increase in hardware as compared to conventional methods. IEEE 754 format specifies two different formats namely single precision and double precision format. Fig. 3.1 shows the different IEEE 754 floating point formats used commonly. The Single precision format is of 32-bit wide and Double precision format is of 64-bit wide. The Most Significand Bit is the sign bit. The exponent is a signed integer. It is often represented as an unsigned value by adding a bias. In Single precision format, the exponent is of 8-bit wide and the bias is 127, i.e. the exponent has a range of(-127 to 128). In Double precision format, the exponent is of 11-bit wide and the bias is 1023, i.e. the exponent has a range of (-1023 to 1024). The mantissa or significand of Single Precision format is of 23-bit and of double precision format is of 52 bit wide. The maximum value that can be represented using floating point format is

Largest significant * base $^{\text{largest exponent}}$

And the minimum value that can be represented is

Smallest significant * base $^{\text{smallest exponent}}$

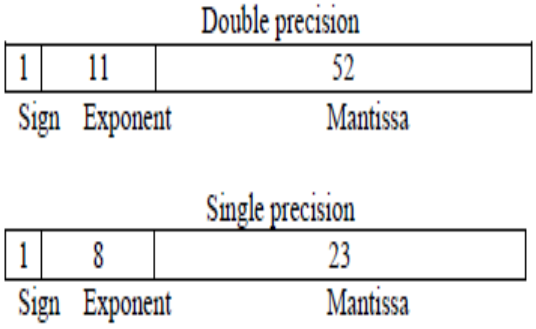



**Fig. 1: Floating point formats in the proposed model**



*Corresponding author: DONURU MADHAVI*

## II.FLOATING POINT MULTIPLIER DESIGN

A floating point number has four parts: sign, exponent, significand or mantissa and the exponent base. A floating point number is represented in IEEE-754 format as $\pm s * b^{e}$ or $\pm$ significand * base $^{exponent}$. The exponent base for binary format is 2. To perform multiplication of two floating point numbers $\pm s1*b^{e1}$ and $\pm s2 * b^{e2}$, the significant or mantissa parts are multiplied to get the product mantissa and exponents are added to get the product exponent. i.e.; the product is as $\pm (s1*s2) * b^{(e1+e2)}$. The hardware block diagram of floating point multiplier is shown in fig. 3.2. The important blocks in the implementation of proposed floating point multiplier is described below.

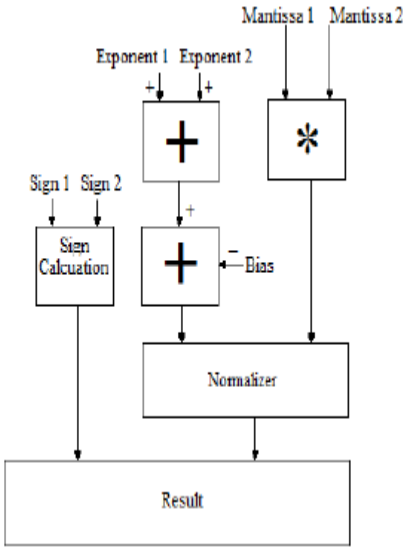



**Fig. 2: Floating point multiplier**

### *Sign Calculation*

The MSB of floating point number represents the sign bit. The sign of the product will be positive if both the numbers are of same sign and will be negative if numbers are of opposite sign. So, to obtain the sign of the pr duct, we can use simple XOR gate as the sign calculator.

### *B. Addition of Exponents*

To get the product exponent, the input exponents are added together. Since we use a bias in the floating point format exponent, we need to subtract the bias from the sum of exponents to get the actual exponent. The value of bias is $127_{10}$ ($01111111_2$) for single precision format and $1023_{10}$($0111111111_2$) for double precision format. In proposed custom precision format also, a bias of 127 is used.

The computational time of mantissa multiplication operation is much more that the exponent addition. So a simple ripple carry adder and ripple carry borrow subtractor is optimal for exponent addition.

### A.Karatsuba-Urdhva Tiryagbhyam binary multiplier

In floating point multiplication, most important and complex part is the mantissa multiplication. Multiplication operation requires more time compared to addition. And as the number of bits increase, it consumes more area and time. In double precision format, we need a 53x53 bit multiplier and in single precision format we need 24x24 bit multiplier. It requires much time to perform these operations and it is the major contributor to the delay of the floating point multiplier. To make the multiplication operation more area efficient and faster, the proposed model uses a combination of Karatsuba algorithm and Urdhva Tiryagbhyam algorithm.

A Karatsuba algorithm uses a divide and conquers approach where it breaks down the inputs into Most Significant half and Least Significant half and this process continues until the operands are of 8-bits wide. Karatsuba algorithm is best suited for operands of higher bit length. But at lower bit lengths, it is not as efficient as it is at higher bit lengths. To eliminate this problem, Urdhva Tiryagbhyam algorithm is used at the lower stages. The model of Urdhva-Tiryagbhyam algorithm is shown in Fig. 3.3.

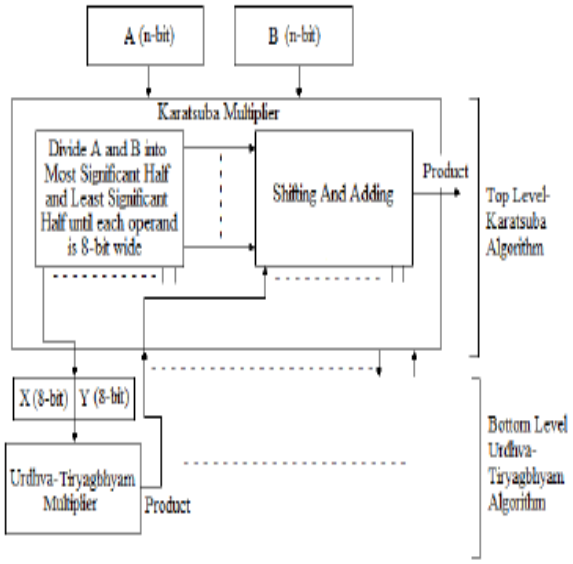



**Fig. 3: Karatsuba-Urdhva multiplier model**

Urdhva Tiryagbhyam algorithm is the best algorithm for binary multiplication in terms of area and delay. But as the number of bits increases, delay also increases as the partial products are added in a ripple manner. For example, for 4-bit multiplication, it requires 6 adders connected in a ripple manner. And 8-bit multiplication requires 14 adders and so on.

Compensating the delay will cause increase in area. So Urdhva Tiryagbhyam algorithm is not that optimal if the number of bits is much more. If we use Karatsuba algorithm at higher stages and Urdhva Tiryagbhyam algorithm at lower stages, it can somewhat compensate the limitations in both the algorithms and hence the multiplier becomes more efficient. The circuit is further optimized by using carry select and carries save adders instead of ripple carry adders. This reduces the delay to a great extent with minimal increase in hardware. These two algorithms are explained in detail in the below sections.

**B. Kogge stone adder**

KSA is another of prefix trees that use the fewest logic levels. The 16 bit kogge stone adder uses BC's and GC's and it won't use full adders. This adder totally operates on generate and propagate blocks

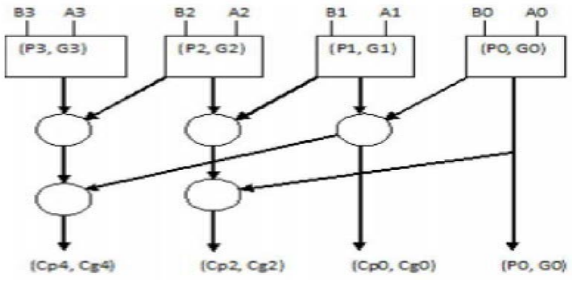



Fig. 4 Kogge stone adder

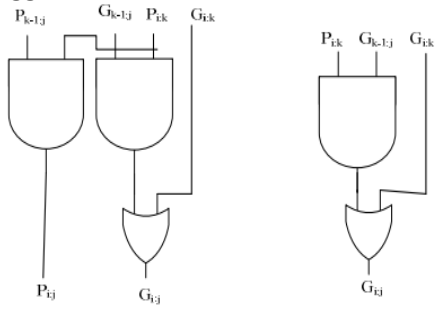


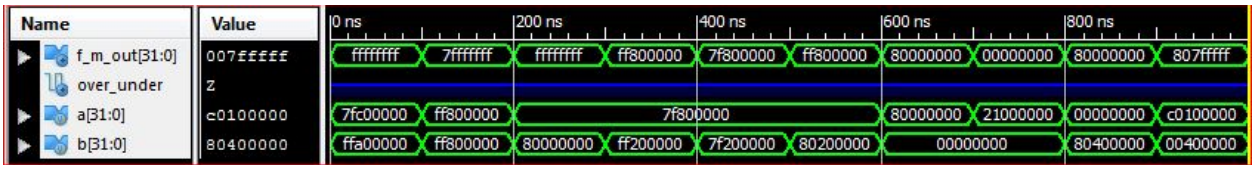

**Fig 5 Black and Gray Cell logic**

**C. Representation of exceptions**

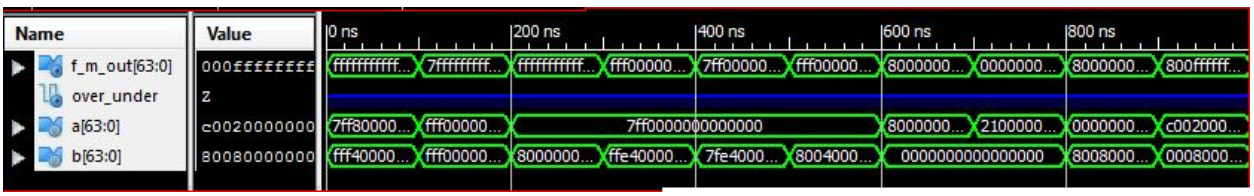
Some of the numbers cannot be represented with a normalized significand. To represent those numbers a special code is assigned to it. In the proposed model, we use four output signals namely Zero, Infinity, NaN (Not-a—number) and Denormal to represent these exceptions. If the product has exponent + bias = 0 and significand = 0, then the result is taken as Zero (0). If the product has exponent + bias = 255 and significand = 0, then the result is taken as Infinity. If the product has exponent + bias = 255 and significand not equal to 0, then the result is taken as NaN. Denormalized values or Denormal are numbers without a hidden 1 and with the smallest possible exponent. Denormal are used to represent certain small numbers that cannot be represented as normalized numbers. If the product has exponent + bias = 0 and significand not equal to 0, then the result is represented as Denormal. Denormal is represented as $0.s \times 2^{-126}$, where s is the significand.

**III.SIMULATION RESULTS**

**Single Precision with KSA**




**Double Precision with KSA**



## IV.CONCLUSION

This paper shows how to effectively reduce the percentage increase in delay and area of a floating point multiplier by using a very efficient combination of Karatsuba and Urdhva Tiryagbhyam algorithms. The model can be further optimized in terms of delay by using pipelining methods and precision of the result can be increased by adding efficient truncation and rounding methods.

## V.REFERENCES


1. IEEE 754-2008, IEEE Standard for Floating-Point Arithmetic, 2008.
2. Computer Arithmetic, Behrooz Parhami, Oxford University Press, 2000.
3. B. Jeevan , S. Narender , C.V. Krishna Reddy, K. Sivani, "A High Speed Binary Floating Point Multiplier Using Dadda Algorithm", International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing, pp. 455-460, 2013
4. "Vedic mathematics", Swami Sri Bharati Krsna Thirthaji Maharaja, Motilal Banarasidass Ideological publishers and Book sellers, 1965
5. R. Sridevi, Anirudh Palakurthi, Akhila Sadhula, Hafsa Mahreen, "Design of a High Speed Multiplier (Ancient Vedic Mathematics Approach)", International Journal of Engineering Research (ISSN : 2319-6890), Volume No.2, Issue No.3, pp : 183-186, July 2013
6. Nivedita A. Pande, Vaishali Niranjane, Anagha V. Choudhari, "Vedic Mathematics for Fast Multiplication in DSP", International Journal of Engineering and Innovative Technology (IJEIT), Volume 2, Issue 8, pp. 245-247, February 2013
7. R.K. Bathija, R.S. Meena, S. Sarkar, Rajesh Sahu, "Low Power High Speed 16x16 bit Multiplier using Vedic Mathematics", International Journal of Computer Applications (0975 – 8887), Volume 59– No.6, pp. 41-44, December 2012
8. Poornima M, Shivaraj Kumar Patil, Shivukumar , Shridhar K P , Sanjay H, "Implementation of Multiplier using Vedic Algorithm", International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-2, Issue-6, pp. 219-223, May 2013
9. Premananda B.S., Samarth S. Pai, Shashank B., Shashank S. Bhat, "Design and Implementation of 8-Bit Vedic Multiplier", International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 2, Issue 12, pp. 5877-5882, December 2013
10. Harpreet Singh Dhillon, Abhijit Mitra, "A Reduced-Bit Multiplication Algorithm for Digital Arithmetic", World Academy of Science, Engineering and Technology, Vol 19, pp. 719-724, 2008


## AUTHOR'S BIOGRAPHY:

**DONURU MADHAVI** has completed his B.Tech in Electronics and Communication Engineering from Noble institute of engineering and technology for Women affiliated to J.N.T.U.H in 2009-2013, established in 2008. She is pursuing her M.Tech in VLSI System Design from Brilliant Institute of Engineering and Technology, J.N.T.U.H Affiliated College.

**V.Venkanna** is an Assistant Professor at BRILLIANT college of Engineering and Technology, Hyderabad in ECE Department. He received his B.Tech degree in Electronics and Communication Engineering from MADIRA Institute of Technology and Sciences, Hyderabad and M.Tech degree in VLSI System Design from NETAJI Institute of Engineering & Technology, Hyderabad.