64×64 Bit High Speed Floating Point Multiplier using Urdhva Triyagbhyam Algorithm

Dharshna L.[1] Deepthi V.[2]
[1,2]Department of Electronics & Communication Engineering
[1,2]Bannari Amman Institute of Technology, Sathyamangalam, Erode, Tamil Nadu, India

Abstract— In VLSI systems like microprocessors and application specific DSP architectures, the arithmetic operation which is extensively used is "Multiplication". This paper proposes an efficient method for IEEE 754 floating point multiplication which gives a better implementation in terms of delay and power. The overall performance of most of the systems is determined by the multipliers. The power efficient, faster and low area multiplier design decides the performance of the system. The speed of operation is increased compared with carry save Multiplier. Multiplication is the most time consuming operation. For this fast method of multiplication based on ancient Indian Vedic mathematics is used. Among various method of multiplication Urdhva Tiryagbhyam (Vedic mathematics) algorithm is used and multiplication is for 64 x 64 bits. It enables parallel generation of intermediate products, eliminates unwanted multiplication steps with zeros. This design using Urdhva Tiryagbhyam sutra exhibits less combinational delay, high speed and power utilization.

Keywords: Urdhva Triyagbhyam Sutra, Power Utilization, Combinational Delay

I. Introduction

Digital multipliers are the core components of all digital signal processors (DSPs) and the speed of the DSP is largely determined by the speed of its multipliers. DSP applications essentially require the multiplication of binary floating point numbers. The IEEE 754 standard provides the format for representation of Binary Floating Point numbers [1, 2].The binary Floating point numbers are represented in Single and Double precision formats. The Single precision consists of 32 bits and the Double precision consists of 64 bits. The formats are comprised of 3 fields: Sign, Exponent and Mantissa. In case of double precision, the Mantissa is represented in 52 bit, the Exponent is represented in 11 bit which is biased to 1023 and the MSB of double is reserved for sign bit. Multiplication of two floating point numbers represented in IEEE 754 format is done by multiplying the normalized 52 bit mantissa, adding the biased 11 bit exponent and resultant is converted in excess 1023 bit format, for the sign bit calculation the input sign bits are XOR Ed. We propose the Vedic multiplication algorithm [3] i.e., Urdhva Triyagbhyam algorithm for multiplication of 52 bit mantissa. Urdhva Triyagbhyam sutra means "vertically-crosswise" multiplication. In this technique the partial products and sums are generated parallelly. The details of Vedic multiplication with their advantages over the conventional multiplication method are discussed.

II. Literature survey

Performance Analysis of 64×64 bit Multiplier Designed Using Urdhva Tiryakbyham and Nikhilam Navatashcaramam Dashatah Sutras - Sai Venkatramana Prasada G S, Seshikala G, Niranjana Sampathila, 2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), IEEE Conference Paper, August 2018. This paper compares the performance of multipliers designed using Urdhva Tiryakbyham and Nikhilam Navatashcaramam Dashatah Sutras. The overall performance of the most systems is determined by the multipliers. The proposed designs were implemented and simulated for parameters such as slices, number of 4 input LUT's and delay. It explains multiplication procedure carried out by both Urdhva Tiryakbyham and Nikhilam Navatashcaramam Dashatah Sutras. The proposed designs were implemented using Verilog code and simulated by Xilinx 10.1 and Cadence simvision with 45nm technology. And is concluded that the multiplier designed using Urdhva Tiryakbyham Sutra exhibits less combinational delay and power utilization.

A Binary High Speed Floating Point Multiplier - Konduri Arun, Srivatsam K, 2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2), IEEE Conference Paper, October 2017. In this paper Recursive Dadda Algorithm is used for the Floating Point Multiplier which is for improving the speed. This paper explains general floating point multiplication algorithm, the existing Carry Save Multiplier algorithm and describes proposed Recursive Dadda Algorithm for mantissa multiplication. The format used in this paper for Floating Point Multiplier is IEEE 754 Single Precision Binary Floating Point representation. For multiplication Carry Save Multiplier is replaced by Dadda Multiplier for improving speed. The multiplier developed handles both overflow and underflow cases.

Design Of Vedic Multiplier Using Urdhva Tiryakbyham Sutra - Harsha R, Anilkumar S R, Chandan R, Manjula N, International Journal Of Advance Research, Ideas And Innovations In Technology, IEEE Journal Article, February 2019. In this paper the computation of partial products in parallel in the Urdhva Tiryakbyham Sutra algorithm increases the speed of the computation process and the processing time is also reduced in comparison with the use of inbuilt MATLAB functions. In the proposed multiplier design, the delay for the 4×4 Vedic multiplier is reduced and also the number of transistors is reduced by a large amount compared to previously proposed design. The results of both designs are compared with the following parameters namely, delay, transistors count, and the type of adders used. The limitations of the proposed design system are the power consumption is high and the design also becomes complex as the size of the numbers to be multiplied increases.

FPGA Implementation of 32 Bit Complex Floating Point Multiplier Using Vedic Real Multipliers With Minimum Path Delay –Deergha Rao K, Muralikrishna P V, Gangadhar C H. 5th IEEE Uttar Pradesh Section

International Conference on Electrical, Electronics and Computer Engineering, IEEE Conference Paper, November 2018. In this paper, two possible architectures are proposed for a Vedic real multiplier based on Urdhva Triyagbhyam (vertically and cross wise) sutra of Indian Vedic mathematics and expression for path delay of an N×N Vedic real multiplier with minimum path delay architecture is developed. Then, architectures of four Vedic real multipliers solution, three Vedic real multipliers solution of complex multipliers are presented. The architectures of Vedic real multiplier with minimum path delay is used in the implementation of complex multiplier. The architectures for the four multiplier solution and three multiplier solution of complex multiplier for 32 × 32 bit complex numbers multiplication are coded in VHDL and implemented through Xilinx ISE 13.4 navigator and Modelsim 5.6. Finally, the results are compared with that of the four and three real multipliers solutions using the conventional Booth and Array multipliers.

High Speed Signed Multiplier for Digital Signal Processing Applications – Saokar S.S and Siddamal S, 2012 International Conference on signal processing, computing and control (ISPCC), IEEE Conference paper, March 2012.In this paper a fast multiplier architecture is proposed for signed Q-format multiplications using Urdhva Triyagbhyam method of Vedic mathematics for 16 bit and 32 bit fractional fixed point multiplications. Further the speed compared with normal booth multiplier and Xilinx LogiCore parallel multiplier Intellectual property (IP) is presented. The Simulation is done using Xilinx ISE version 12.2 and the results clearly indicate that Urdhva Triyagbhyam can have a great impact on improving the speed of Digital Signal Processors.

III. Overview of Floating Point Multiplication

A. *Floating Point Multiplication Algorithm*

Different techniques and algorithms are used for designing multipliers. The main aim is to minimize the time delay, latency period, area, power consumption and to maximize the accuracy and speed of the multiplication process. The following steps are necessary to multiply two floating point numbers.

1) Multiplying the significant i.e., (1.M1 x 1.M2).
2) Placing the decimal point in the result.
3) Adding the exponents and subtract with the bias i.e., (E1 + E2 - Bias).
4) Obtaining the sign of final result. i.e., (S1 XOR S2).
5) Normalizing the result from the multiplication by obtaining 1 at the MSB of significant multiplication's result.
6) Rounding the normalized result to fit in the given floating point number format.
7) Checking for underflow and overflow condition.

B. *Main Blocks of Floating Point Multiplier*

The double precision format helps overcome the problems of single precision floating point. The main sub blocks of floating point numbers represented in IEEE 754 can be divided in four different units they are Sign calculation block, exponent calculation block, mantissa multiplier block and normalization unit as shown below.

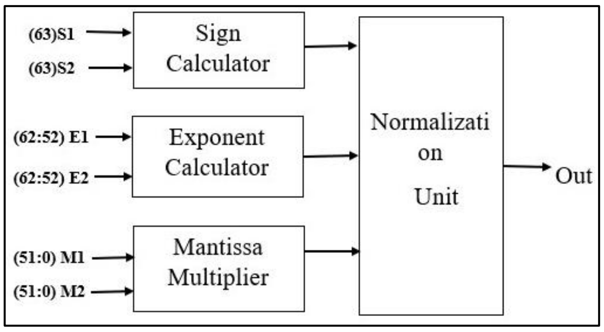


Fig. 1: Main Blocks of Floating Point Multiplier

1) *Sign Calculation Block*

This block is used to calculate the sign of result. In this block there are two inputs given to an XOR gate. MSB bit of the floating point format indicates the sign of the number. So two inputs for XOR gate are bit number 63 of both operands as the operand inputs are of [63:0] 64 bit each. If both inputs are the same (i.e., 11 or 00) then the output will be '0', which indicates positive sign and if both inputs are different (i.e., 10 or 01) then the output will be '1', which indicates negative sign. By the aid of a truth table we find that this can be obtained by XOR ing the sign of two inputs.

2) *Exponent Calculation Block*

This block is used to calculate the exponent value of the output. In this block, the inputs are first added together and then the output of the addition is subtracted with bias. For exponent calculation 11-bit adder is used, as the floating point format is double precision in which exponent has 11 bits. The inputs to the 11-bit adder are exponents of the two operands (i.e., [62:52] E1 and [62:52]E2). Subtraction is done by adding 2's complement of bias. 2's complement is calculated by adding binary 1 in 1's complement of given number. In this block bias is always constant and it is equal to 1023 i.e. (EA=EA – true + bias and EB=EB – true + bias). Hence ER=E1+E2-1023.

3) *Mantissa Multiplier Block*

The performance of mantissa calculation unit dominates the overall performance of the floating point multiplier. This unit requires integer's multiplier for multiplication of 52*52 bits. The Vedic Multiplication technique is chosen for the implementation of this unit. This technique gives result in terms of speed and power. For design of 54 bit multiplier four Vedic 27 bit multipliers are used.

4) *Normalization Unit*

Normalized number means it will have leading '1' just immediate to the left hand side of the decimal point in 106th bit of mantissa multiplication result. If the one is at 106th bit (i.e. to the left of a decimal point) then there is no need of the intermediate product and is known to be a normalized number. The product is shifted to the right and exponent is incremented by 1 if the leading one is at 107th bit. According to the position of decimal point exponent is adjusted. Leading '1' is skipped from the result and then the remaining bits, after the decimal point are truncated to 52 bit which is mantissa of the final result. If the decimal point is shifted to the left hand side then the exponent is decremented and if the decimal point is shifted to the right hand side then the exponent is incremented. Number of increase or decrease in the exponent value is equal to the number by which decimal point is shifted. Rounding of a number is also performed by the normalizer, because floating point arithmetic operations compute result that cannot be represented in a specified amount of precision.

C. Proposed 64×64 Bit Floating Point Multiplication Architecture

This project presents an efficient double precision floating point multiplier design. High speed can be achieved by using the Vedic multiplier. Floating point multiplication of numbers represented using either single precision or double precision format which involve calculation of sign bit, exponent and mantissa of the product and then normalizing and rounding off the results. Sign bit is calculated by XOR operation of the sign bits of two operands. The exponents of both the operands are added to obtain the product exponent. The resultant sum is subtracted from 127 in case of single precision and 1023 in case of double precision to obtain its biased exponent. The addition operation involves the use of 8-bit and 11-bit ripple carry adder for single and double precision respectively. In order to improve the performance high speed adders can be used. The mantissa is multiplied using any of the multiplication algorithms. The block diagram of floating point multiplier for 64x64 bit is shown below.

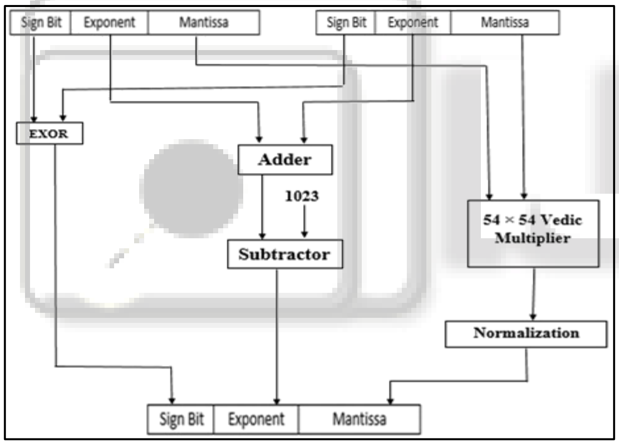


Fig. 2: Proposed Architecture

D. Urdhva Triyagbhyam Sutra

Urdhva Triyagbhyam sutra is applicable to all cases of multiplication. This sutra performs the multiplication using the principle vertically and crosswise multiplication. The generation of all partial products can be done with the concurrent addition of these partial products. The algorithm can be generalized for n x n bit number. This multiplier is independent of the clock frequency of the processor as the partial products and their sums are calculated in parallel.
Taking 2 bit case a1, a0 and b1, b0 then

1) Step 1: AND of a[0] with b[0]
2) Step 2: Cross AND of a[0] with b[1]
3) Step 3: Cross AND of a[1] with b[0]
4) Step 4: Cross AND of b[1] with a[0]
5) Step 5: ADD them side by side

The steps involved in 2 bit Vedic multiplier and when written in Verilog code

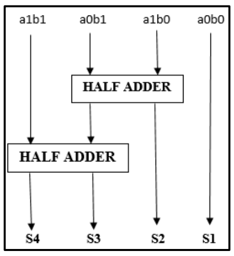


Fig. 3: 2×2 bit Vedic Multiplier

IV. Design & Simulation

A. Vedic Multiplier Design

1) 3x3 bit Vedic multiplier

The Vedic multiplication of two 3 bit binary numbers A2A1A0 and B2B1B0 is shown below.

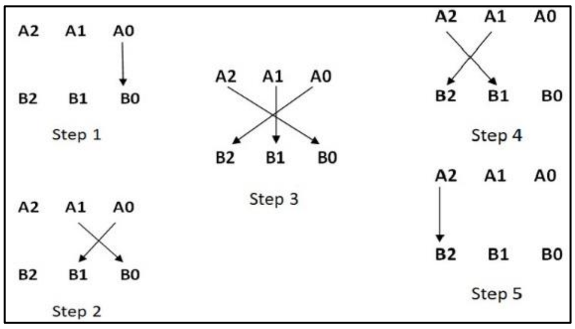


Fig. 4: 3 bit binary numbers multiplication

2) 9x9 Bit Vedic Multiplier

The 9x9 bit Vedic multiplier has nine 3x3 bit Vedic multiplier.

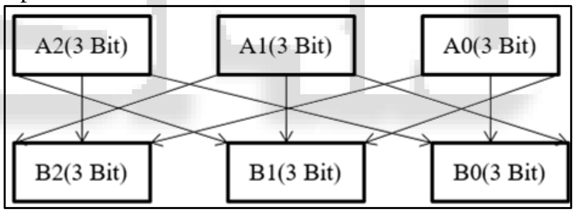


Fig. 5: 9x9 Bit Vedic multiplier

3) 27x27 Bit Vedic Multiplier

27×27 Vedic multiplier is designed using Urdhva Triyagbhiyam algorithm and it has inputs of A0, A1, A2, B0, B1, B2, each of which is 9 bit.

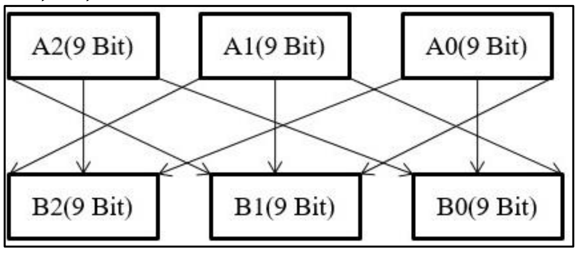


Fig. 6: 27x27 Bit Vedic Multiplier

4) 54x54 Bit Vedic Multiplier

54×54 bit Vedic multiplier is designed using Urdhva Triyagbhiyam algorithm and it has inputs A0, A1, B0, B1, each of which is 27 bit.

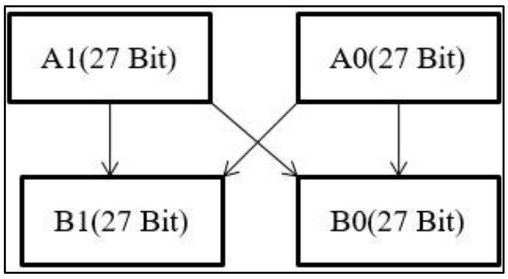


Fig. 7: 54x54 Bit Vedic Multiplier

V. Xilinx Synthesis & Result Analysis

The overall top view of the proposed method (64×64 bit high speed floating point multiplier using Urdhva Triyagbhyam algorithm) is given in the figure 7.1 there are two inputs input 1(64 bit), input 2(64 bit), and the outputs are result(64 bit), exception, overflow, underflow, zero. The exception, overflow, underflow, zero will determine the overall output of the proposed method, where the exception calculated by using bit-wise AND operation is performed to exponent bit and they are OR'ed, the zero is calculated by the exception, if the exception is 1 then the mantissa part become 0's the zero will be one, the overflow is calculated when the exponent's 11th bit, NOT of 10th bit, NOT of zero are AND'ed, the underflow is calculated when exponent's 11th bit, 10th bit, NOT of zero are AND'ed, when the exception is 1 the result will 64 bit decimal 0's, when the zero is 1 the sign bit is calculated and the exponent part mantissa parts are 0's, when overflow is 1 the sign bit is calculated, exponent part is of 1's, mantissa part is of 0's, when the underflow is 1 sign bit is calculated and the exponent and mantissa parts will be 0's, when exception, zero, overflow, underflow, all are 0 the sign bit, exponent part, mantissa part are calculated and are displayed in the result. The simulated output of 64×64 floating point multiplier using Urdhva Triyagbhyam algorithm is given in figure 7.2 where the inputs are given as input1, input2, and the output is given as result which is of 64 bit.

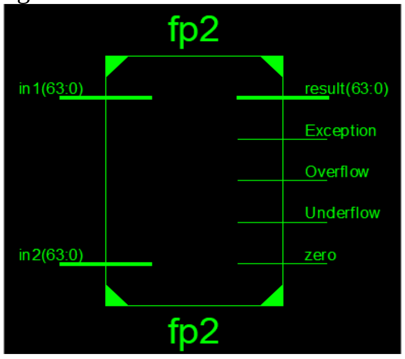
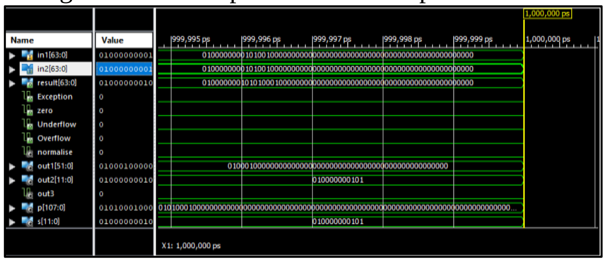


Fig. 8: Overall Top View of the Proposed Method



```
                                                                       1,000,000 ps
Name            Value         999,995 ps  999,996 ps  999,997 ps  999,998 ps  999,999 ps  1,000,000 ps
in1[63:0]       01000000001   0100000000101001000000000000000000000000000000000000000000000000
in2[63:0]       01000000001   0100000000101001000000000000000000000000000000000000000000000000
result[63:0]    01000000010   0100000000101010001000000000000000000000000000000000000000000000
Exception       0
zero            0
Underflow       0
Overflow        0
normalise       0
out1[51:0]      01000100000   0100010000000000000000000000000000000000000000000000
out2[11:0]      01000000010   010000000101
out3            0
p[107:0]        01010001000   0101000100000000000000000000000000000000000000000000000000000000000000...
s[11:0]         01000000010   010000000101
X1: 1,000,000 ps
```

Fig. 9: Simulated Output of 64×64 bit Floating Point Multiplier using Urdhva Triyagbhyam Algorithm

The listed number of LUTs verifies that the hardware requirement is reduced, thereby reducing the power consumption. The maximum combinational path delay is 6.3530ns.

State Logic Utilization	Used	Utilization
Number Of Slice Registers	0	0%
Number Of Slice LUT's	4,833	17%
Number Of Bonded IOBs	196	89%

Table 1: Device Utilization Summary for the Proposed System

VI. Conclusion

This paper represents an efficient double precision floating point design. High speed can be achieved by using this Vedic multiplier. Urdhva triyabghyam algorithm can reduce the delay, power and hardware requirements for multiplication of numbers. The design algorithm and results show that this Vedic multiplier requires less area and high speed as compared to the conventional multipliers. In future this multiplier design can be used in digital signal processing or VLSI signal processing application such as FFT, IFFT, and complex floating multiplier.

References

[1] Energy-efficient VLSI implementation of multipliers with double LSB operands-Vasileios Leon, Sotirios Xydis, Dimitrios Soudris, Kiamal Pekmestzi, IEEE Journal, October 2019.
[2] Performance Analysis of 64x64 bit Multiplier Designed Using Urdhva Tiryakbyham and Nikhilam Navatashcaramam Dashatah Sutras -Sai Venkatramana Prasada G S, Seshikala G, Niranjana Sampathila, IEEE Conference, and August 2018.
[3] A Binary High Speed Floating Point Multiplier – Konduri Arun, Srivatsam K, IEEE Conference, October 2017.
[4] Minimally Biased Multipliers For Approximate Integer And Floating Point Multiplication – Hassaan Saadat, Haseeb Bokhari, Sri Parameswaran, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, IEEE Journal Article, November 2018.
[5] Design Of Vedic Multiplier Using Urdhva Tiryakbyham Sutra –Harsha R, Anilkumar S R, Chandan R, Manjula N, International Journal Of Advance Research, Ideas And Innovations In Technology, IEEE Journal, February 2019.
[6] FPGA Implementation of 32 Bit Complex Floating Point Multiplier Using Vedic Real Multipliers With Minimum Path Delay-Deergha Rao K, Muralikrishna P V, Gangadhar C H, IEEE, November 2018.
[7] High Speed Signed Multiplier for Digital Signal Processing Applications-Saokar S.S, Siddamal S, International Conference on Signal Processing, Computing and Control, IEEE, March 2012.