Korra Tulasi Bai, J. E. N. Abhilash / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 4, Jul-Aug 2013, pp.1801-1804

A New Novel Low Power Floating Point Multiplier Implementation Using Vedic Multiplication Techniques

Korra Tulasi Bai, J. E. N. Abhilash
Dept. Of Electronics & C communication SCET, Narsapur, W. G. Dist
Associate. Professor, Dept of E.C.E SCET, Narsapur, W. G. Dist

Abstract

In this paper, Vedic Multiplication Technique is used to implement IEEE 754 Floating point multiplier. The Urdhva-triyak bhyam sutra is used for the multiplication of Mantissa. The underflow and over flow cases are handled. The inputs to the multiplier are provided in IEEE 754, 32 bit format. The multiplier is implemented in VHDL and Virtex-5 FPGA is used.

Keywords: Vedic Mathematics, Urdhva-triyakbhyam sutra,Floating Point multiplier, Field Programmable Gate Araay (FPGA).

I. INTRODUCTION

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 formats. The Single consist of 32 bits and the Double consist of 64 bits. The formats are composed of 3 fields; Sign, Exponent and Mantissa. The Figure 1 shows the structure of Single and Double formats of IEEE 754 standard. In case of Single, the Mantissa is represented in 23 bits and 1bit is added to the MSB for normalization, Exponent is represented in 8 bits which is biased to 127, actually the Exponent is represented in excess 127 bit format and MSB of Single is reserved for Sign bit. When the sign bit is 1 that means the number is negative and when the sign bit is 0 that means the number is positive. In 64 bits format the Mantissa is represented in 52 bits, the Exponent is represented in 11 bits which is biased to 1023 and the MSB of Double is reserved for sign bit

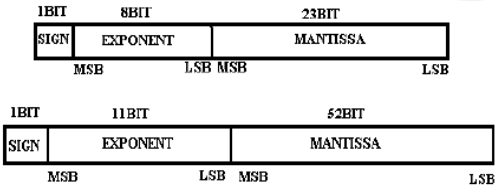


Fig 1. IEEE Format for single and double

Multiplication of two floating point numbers represented in IEEE 754 format is done by multiplying the normalized 24 bit mantissa, adding the biased 8 bit exponent and resultant is converted in excess 127 bit format, for the sign calculation the input sign bits are XORed. In this paper, we propose the Vedic Multiplication algorithm [3] for multiplication of 24 bit mantissa. The details of Vedic Multiplication with their advantages over the conventional multiplication method are discussed in the section III.

The paper describes the implementation and design of IEEE 754 Floating Point Multiplier based on Vedic Multiplication Technique. Section II explores the basics of IEEE 754 floating point representation and implementation of floating point multiplier using Vedic multiplication technique. Section III describes the idea behind Vedic multiplication. Section IV comprises of result and conclusion.

II. FLOATING POINT MULTIPLICATION

The multiplier for the floating point numbers represented in IEEE 754 format can be divided in four different units:
Mantissa Calculation Unit
Exponent Calculation Unit
Sign Calculation Unit
Control Unit

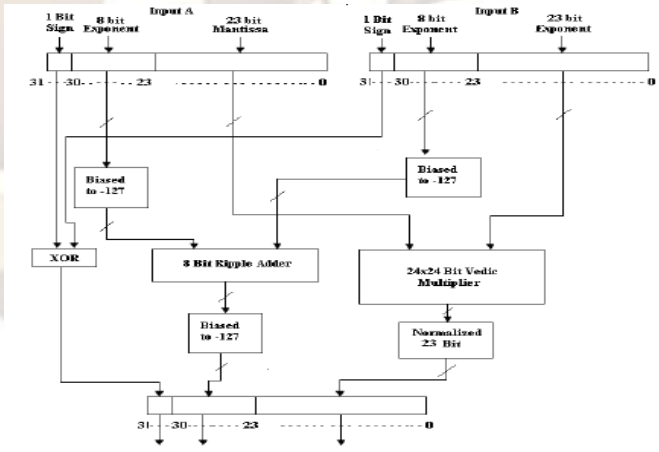


Fig 2. Proposed architecture for Floating point Multiplier

The standard format for representation of floating point number is $(-1)^S\ 2^E\ (b0 \cdot b1b2 \ldots bp-1)$
The biased exponent e =E+127, and the fraction f = b1*b*2 ----*bp*−1.

The Mantissa Calculation Unit requires a 24 bit multiplier if 32 bit single IEEE 754 format is considered [4]. In this paper we propose the efficient use of Vedic Multiplication Technique for this 24 bit multiplier. The Exponent Calculation Unit is implemented in this paper using 8 BIT Ripple Carry Adder [5, 6].The advantages of ripple carry adder in addition to its implementation ease are low area and simple layout [7]. The Control Unit raises the flag when NaN, Infinity, zero,underflow and overflow cases are detected. The control unit raises appropriate flag accordingly when the cases occurs. The various cases and its constituent flags are:

If e = 255 and f ≠ 0, then NaN
If e = 255 and f = 0, then Infinity
If 0 < e < 255, then Number is (-1) s 2e-127(1 · f)
If e = 0 and f ≠ 0, then (-1)s 2-126(0 · f) (demoralized numbers)
If e = 0 and f = 0, then zero.

Figure 2 shows the proposed architecture for the Floating point multiplier. Consider the multiplication of two floating point numbers A and B, where A = -19.0 and B = 9.5. The normalized binary representation are A=-1.0011x24 and B = 1.0011x23. IEEE representations of operands are:

	Sign	Exponent	Mantissa
A =	1	10000011	00110000000000000000000
B =	0	10000010	00110000000000000000000

Here, MSB of the 32 bit operand shows the sign bit, the exponents are expressed in excess 127 bit and the mantissa is represented in 23 bit. Sign of the result is calculated by XORing sign bits of both the operands A and B [8]. In this case sign bit obtained after XORing is 1. Exponents of A and B are added to get the resultant exponent. Addition of exponent is done using 8 bit ripple carry adder Figure 3.

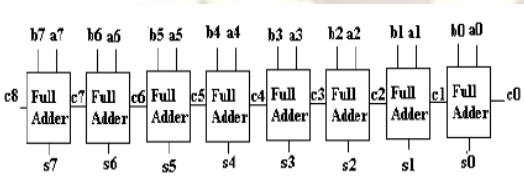


Fig 3. Eight BIT Ripple Carry Adder

After addition the result is again biased to excess 127 bit Code. For this purpose 127 is subtracted from the result. Two's complement subtraction using addition is incorporated for this purpose. If ER is the final resultant exponent then, ER = EA + EB – 127 where EA and EB are the exponent parts of operands A and B respectively. In this case ER = 10000110. Mantissa multiplication is done using the 24 bit Vedic Multiplier. The mantissa is expressed in 23 bit which is normalized to 24 BIT by adding a 1 at MSB.

The normalized 24 bit mantissas are
100110000000000000000000
100110000000000000000000
Multiplication of two, 24 bit mantissa is done using the Vedic Multiplier. In this case 48 bit result obtained after the multiplication of mantissa is 101101001000000000000000000000000000000000000000 Now setting up three intermediate results the final result (normalizing the mantissa by eliminating most significant 1) we obtained is:
1 10000110 01101001000000000000000
This result is deduced as
AxB = -19.0 x 9.5 = -180.5 = -1.01101001x2134-127 = (-10110100.1) 2 = (-180.5) 10

III. MULTIPLIER DESIGN

The performance of Mantissa calculation Unit dominates overall performance of the Floating Point Multiplier. This unit requires unsigned multiplier for multiplication of 24x24 BITs. The Vedic Multiplication technique is chosen for the implementation of this unit. This technique gives promising result in terms of speed and power [9].The Vedic multiplication system is based on 16 Vedic sutras or aphorisms, which describes natural ways of solving a whole range of mathematical problems. Out of these 16 Vedic Sutras the Urdhva-triyakbhyam sutra is suitable for this purpose. In this method the partial products are generated simultaneously which itself reduces delay and makes this method fast. The method for multiplication of two, 3 BITs number is shown Figure 4. Consider the numbers A and B where A = a2a1a0 and B = b2b1b0. The LSB of A is multiplied with the LSB of B:

s0=a0b0;

Then a0 is multiplied with b1, and b0 is multiplied with a1 and the results are added together as:

c1s1=a1b0+a0b1;

Here c1 is carry and s1 is sum. Next step is to add c1 with the multiplication results of a0 with b2, a1 with b1 and a2 with
b0. c2s2=c1+a2b0+a1b1 + a0b2;

Next step is to add c3 with the multiplication results of a1 with b2 and a2 with b1.
c3s3=c2+a1b2+a2b1;
Similarly the last step
c4s4=c3+a2b2;

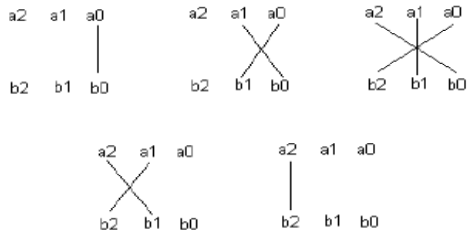


Fig 4. The Vedic Multiplication method

Now the final result of multiplication of A and B is c4s4s3s2s1s0.

The block diagram of 24x24 BIT multiplier is shown in Figure 5.This multiplier is modelled using structural style of modelling using VHDL. In this paper first a 3x3 Vedic multiplier is implemented using the above mentioned method. The 6x6 is designed using four 3x3 multipliers. After that the 12x12 is implemented using four 6x6 BIT multiplier. Finally the 24x24 BIT multiplier is made using four 12x12 BIT multipliers. The 24x24 BIT multiplier requires four 12x12 BIT multipliers and two 24 BIT ripple carry adders and 36 BIT ripple carry adders.

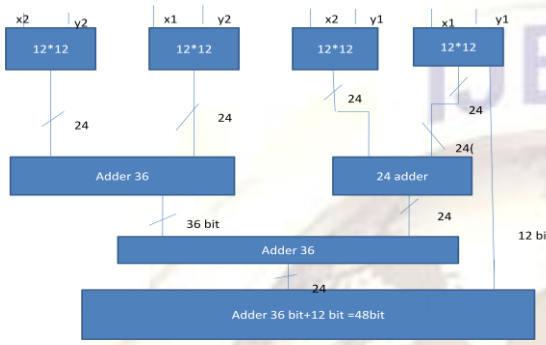


Fig 5. Block diagram of 24x24 BIT Vedic multiplier.

A 24*24 Vedic multiplier is design by using four 12*12 Vedic multipliers based urdhva triyakbhyam.

Here first block 12X12 multiplier consists of lower 12 bits of x i.e. x(11 down to 0) and y(11 down to 0), second block 12*12 Vedic multiplier inputs are x(23 down to 12) and y(11 down to 0) out off 24 bit output of first block lower adder 12 bits are separated and higher order bits are appended as 12 lower bits infront augmented with "000000000000" now second block 24 bits and above 24 generate bits are added and 24 bits sum is generated using 24 bit ripple carry adder.

Higher order 12 bit of y(23 down to 12) and lower order 12 bits of x i.e x(11 down to 0) are multiplier and appended infront with "000000000000" to make 36 data similarly both higher order 12 bits of x and y i.e x(23 downto 12) and y(23 downto 12) are multiplier and 36 bit data is formed by appending "000000000000".

The above two 36 bits are added to generate a 36 bit data in the right side resultant 36 bits are added to generate final 36 bits the resultant of 24*24 multiplier is 48 bits consists of 36 higher bits from 36 bit adder output and lower order 12 bits 36+12=48 bits

The number of LUTs and slices required for the Vedic Multiplier is less and due to which the power consumption is reduced [9]. Also the repetitive and regular structure of the multiplier makes it easier to design. And the time required for computing multiplication is less than the other multiplication techniques.

An Overflow or Underflow case occurs when the result Exponent is higher than the 8 BIT or lower than 8 BIT respectively. Overflow may occur during the addition of two Exponents which can be compensated at the time of subtracting the bias from the exponent result. When overflow occurs the overflow flag goes up. The under flow can occur after the subtraction of bias from the exponent, it is the case when the number goes below 0 and this situation can be handled by adding 1 at the time of normalization. When the underflow case occur the under flow flag goes high.

IV. RESULT AND CONCLUSION

The multiplier is designed in VHDL and simulated using Modelsim Simulator. The design was synthesized using Xilinx ISE 10.1 tool targeting the Xilinx Virtex 5 xc5vlx30-3-ff324 FPGA. A test bench is used to generate the stimulus and the multiplier operation is verified. The over flow and under flow flags are incorporated in the design in order to show the over flow and under flow cases.

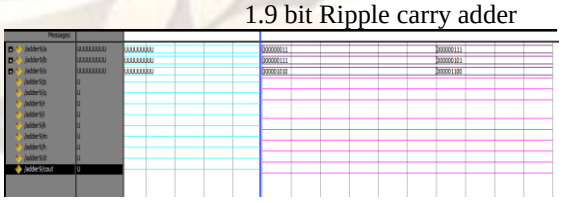
The paper shows the efficient use of Vedic multiplication method in order to multiply two floating point numbers. The lesser number of LUTs verifies that the hardware requirement is reduced, thereby reducing the power consumption. The power is reduced affectively still not compromising delay so much. The Table 1 shows the summary of the multiplier tested.

Parameters	This work	[10]
Device	virtex5	vertex2p
Power consumption	29.72mW	55mW
Time delay	5.246ns	3.070ns
Number of LUT'S	1032	1316
Number of I/O'S	99	100
Power delay product	155.92pJ	168.85pJ

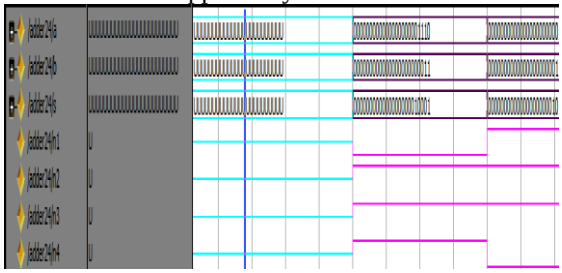
Table:1 Comparisons of results

V. SIMULATION RESULTS:

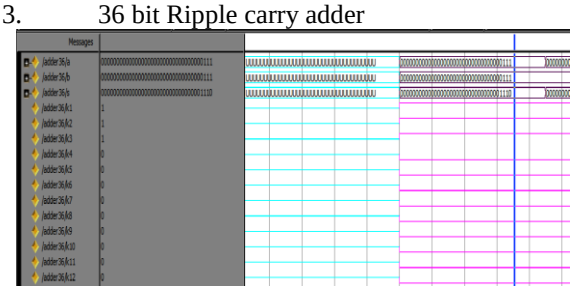
1.9 bit Ripple carry adder



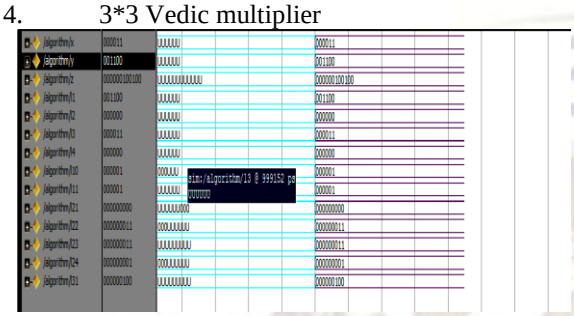
2. 24 bit Ripple carry adder



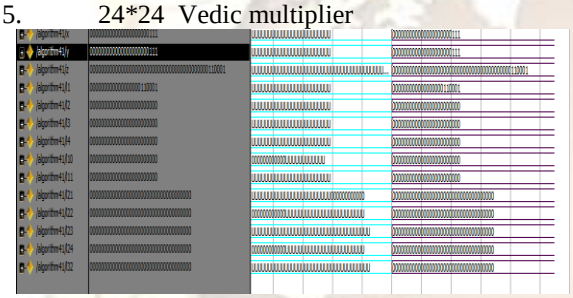
3. 36 bit Ripple carry adder



4. 3*3 Vedic multiplier



5. 24*24 Vedic multiplier



REFERENCES

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

[2] Brian Hickmann, Andrew Krioukov, and Michael Schulte, Mark Erle,"A Parallel IEEE 754 Decimal Floating-Point Multiplier," In 25th International Conference on Computer Design ICCD, Oct. 2007

[3] Jagadguru Swami Sri Bharati Krisna Tirthaji Maharaja, "Vedic Mathematics Sixteen Simple Mathematical Formulae from the Veda,"1965.

[4] Metin Mete ÖZBLEN, Mustafa GÖK, "A Single/Double Precision Floating-Point Multiplier Design for Multimedia Applications," Journal