International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 10 (2015) pp. 25995-26003

http://www.ripublication.com

Designing of ALU using Vedic Multiplier for Single Precision IEEE 754 Standard Floating Point Numbers

Satnam Singh Shergill and Arvind Kumar

ECE Department, U.I.E.T., Panjab University, Chandigarh, India
satnam1_shergill@yahoo.co.in
ECE Department, U.I.E.T., Panjab University, Chandigarh, India
arvind_uiet@yahoo.co.in

Abstract

With ever increasing dependence of computing devices (such as personal computers, laptops, smart phones) in almost every sphere of life, we need faster and reliable processing units and these processing units should also be small enough so as not to cause any space or weight problems. Use of Vedic Mathematics in designing components of CPU such as multiplier [1] has been successful in increasing speed and reducing space consumed by the device. ALU made using Vedic Multiplier for floating point numbers has also shown similar improvements which encourage us to use Vedic Mathematics in our digital endeavors. This work performed on Xilinx Virtex 4 device has shown reduction of approximately 15% in delay and approximately 27% in equivalent gate count.

Keywords—Floating Point, IEEE 754, Vedic, Urdhva-Tiryakbhyam, ALU.

I. Introduction

Since the quality of image and video signals is getting better at exponential speeds as we are getting better with the hardware that processes those signals. But through the results of this work it can be implied that higher quality of image and video signals can be processed at the existing hardware without needing any hardware improvement or extension of any kind. The use of Vedic Mathematics in developing multiplication algorithms for floating point numbers has already shown improvements [1]. We have chosen floating point numbers because floating point numbers are the choice of the programmers working on high end computer game developments and because it's the floating point numbers that require special hardware called FPU (Floating Point Unit) on many chips because their processing is resource extensive.

II. VEDIC MULTIPLICATION ALGORITHMS

Vedic Mathematics is so called because it is said that the group of techniques described as Vedic Mathematics were rediscovered from Vedas between 1911 and 1918 by Sri Bharati Krishna Tirthaji (1884-1960). These techniques are called ‘sutras’. There are sutras for all the basic functions of mathematics. From these sutras there are about 4 sutras that can be used for multiplication [2]. These are as follows:

i. **Nikhilam Navatashcaramam Dashatah:** Nikhilam Navatashcaramam Dashatah Sutra is a specific method of multiplication in Vedic Mathematics which shows how to multiply numbers which are closer to the power of 10 (10, 100, 1000……)
ii. **Anurupyena:** Anurupyena is another specific method of multiplication in Vedic Mathematics which shows how to multiply numbers which are not closer to the power of 10 but are closer to each other.
iii. **Urdhva-Tiryakbyham:** Urdhva-Tiryakbyham is a general method of multiplication in Vedic Mathematics which provides shortcut to multiply any types of numbers.
iv. **Vinculum:** Vinculum is special method of multiplication in Vedic Mathematics which is used with Urdhva Tiryakbhyam whenever we have bigger digits like 6, 7, 8 and 9.

From the above four sutras, the one that can be applied to every multiplication is Urdhva-Tiryakbhyam.

III. IEEE 754 STANDARD FOR FLOATING POINT NUMBERS[3]

Floating Point numbers are represented using IEEE 754 standard. According to this standard, there are bits reserved for sign, exponent and mantissa (also called significand). There are two types of floating point representation in this standard:

i. Single Precision (32 bit)
ii. Double Precision (64 bit)

Reservation of bits in Single Precision and Double Precision Floating Point format:

31	30-23	22-0
Sign Bit	Exponent	Mantissa

Fig. 1: Single Precision bit reservation

63	62-52	51-0
Sign Bit	Exponent	Mantissa

Fig. 2: Double Precision bit reservation

Sign bit represents the sign of the number. '0' represents positive and '1' represents negative number.

Since the exponent needs to represent both positive and negative values, a bias is added to the real value of the exponent in this standard. For single precision, the bias that is added is 127 and for double precision it is 1023. For example, if we want to represent 2^{23} in single precision format, we will have to store 150 in the exponent field. Similarly, for double precision, if we want to represent 2^{23}, we will need to store 1046 in the exponent field.

The mantissa has 23 bits in case of single precision and 52 bits in case of double precision. But, in both cases, an implicit '1' is considered to be appended to the left of the mantissa, thus making the effective length of mantissa to be 24 bit and 53 bit in single precision and double precision formats respectively.

IV. ARITHMETIC LOGIC UNIT FOR FLOATING POINT NUMBERS

The ALU that we designed is for floating point numbers of single precision as described in IEEE 754 standard. There are three arithmetic functions and three logic functions that we incorporated in our design.

The three arithmetic functions are as follows:

Table 1: Arithmetic Functions implemented in the proposed ALU.

S.No.	Function	Technique
1.	Multiplication	Urdhva-Tiryakbhyam Sutra of Vedic Mathematics
2.	Addition	Ripple Carry Adder
3.	Subtraction	Ripple Carry Adder

The three logic functions are as follows:

Table 2: Logic Functions implemented in the proposed ALU.

S.No.	Function
1.	Not
2.	And
3.	Or

We used three select lines to select one of the above stated functions.

Table 3: Select Line values for different functions.

Select Line	Function
010	Multiplication
000	Addition
001	Subtraction
100	NOT
101	AND
110	OR

V. DESIGN OF VEDIC MULTIPLICATION UNIT

For implementing ALU, we first implemented the Vedic Multiplier [1] using Urdhva-Tiryakbhyam sutra.

The sign of the result of the multiplier was ascertained by applying XOR function on the sign bits of the inputs to the multiplier.

The exponent was calculated by adding the two exponents of the input and then subtracting the bias from the result. This was done because we know that in single precision IEEE 754 standard the exponent is stored with a bias of 127. So,

$$\text{Result}_{\text{exponent}} = \text{Input1}_{\text{exponent}} + \text{Input2}_{\text{exponent}} - 127 \qquad (1)$$

For mantissa, we first implemented Urdhva-Tiryakbhyam sutra for 3x3 bit multiplier [1]. Urdhva-Tiryakbhyam literally translates to vertically and crosswise. 3x3 bit Urdhva-Tiryakbhyam sutra can be explained as:

$A=a_2a_1a_0$

$B=b_2b_1b_0$

$a_2b_2/a_2b_1+a_1b_2/\overline{a_2b_0+a_1b_1}+a_0b_2/a_1b_0+a_0b_1/a_0b_0$

Then following Karatsuba-Ofman technique [4] we implemented 6x6 multiplier and from 6x6 multiplier 12x12 multiplier was implemented and from that we implemented 24x24 bit multiplier.

Karatsuba-Ofman algorithm says that every 2n digit multiplication can be broken down into four n bit multiplications by dividing each of the inputs into two parts: higher and lower bits. Let A and B be two binary multiplicands, then A and B can be described as follows:

$$A = \sum_{k=0}^{2n-1} 2^k \mathrm{a_k}$$

$$B = \sum_{k=0}^{2n-1} 2^k \mathrm{b_k}$$

Now, A and B can be broken down into two parts as described below:

$$A = 2^n \sum_{k=0}^{n-1} 2^k \mathrm{a_{k+n}} + \sum_{k=0}^{n-1} 2^k \mathrm{a_k}$$

$$B = 2^n \sum_{k=0}^{n-1} 2^k \mathrm{b_{k+n}} + \sum_{k=0}^{n-1} 2^k \mathrm{b_k}$$

Then, A and B can be written as:

$A=A_H2^n + A_L$

$B=B_H2^n + B_L$

The result of their multiplication can be found by:

$$\begin{aligned} \text{Result} &= A * B \\ &= (A_H2^n + A_L)*(B_H2^n + B_L) \\ &= A_HB_H2^{2n} + A_HB_L2^n + A_LB_H2^n + A_LB_L \end{aligned}$$

From the above expression, we can very well ascertain that to implement it, we need four multipliers and three adders. So, to implement 24x24 bit multiplier from 12x12 bit multiplier we used four 12 bit multipliers and three 24 bit adders which is in contrary to the [1] because they used two 36 bit adders and one 24 bit adder. In our design there was no need of two different adders. This technique of using only one type of adder was taken from [5].

The following figure shows the realization of 24x24 bit multiplier using 12x12 bit multiplier.

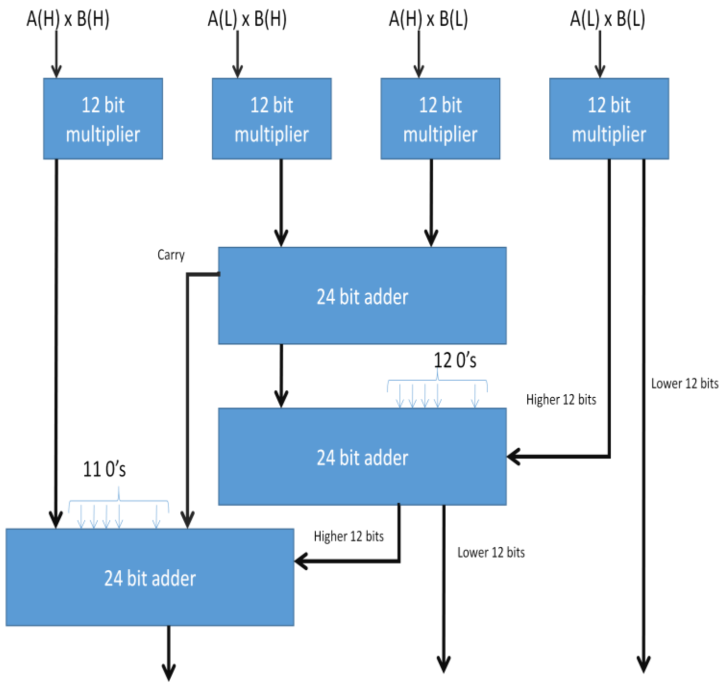

Fig. 3: Realization of 24x24 bit multiplier using 12x12 bit multiplier.

Since our output also has to be in IEEE 754 format, we have rounded off the 48 bit result to 24 bit result using rounding towards zero technique also called truncation. For this we have taken out only the most significant 24 bits of this 48 bit result. Note that while the range of floating point numbers is very high, there can be precision loss while handling cases like this.

VI. DESIGN OF ADDITION AND SUBTRACTION UNIT

For addition and subtraction [6], we have used ripple carry adder. The addition and subtraction was implemented as shown in the following flowchart.

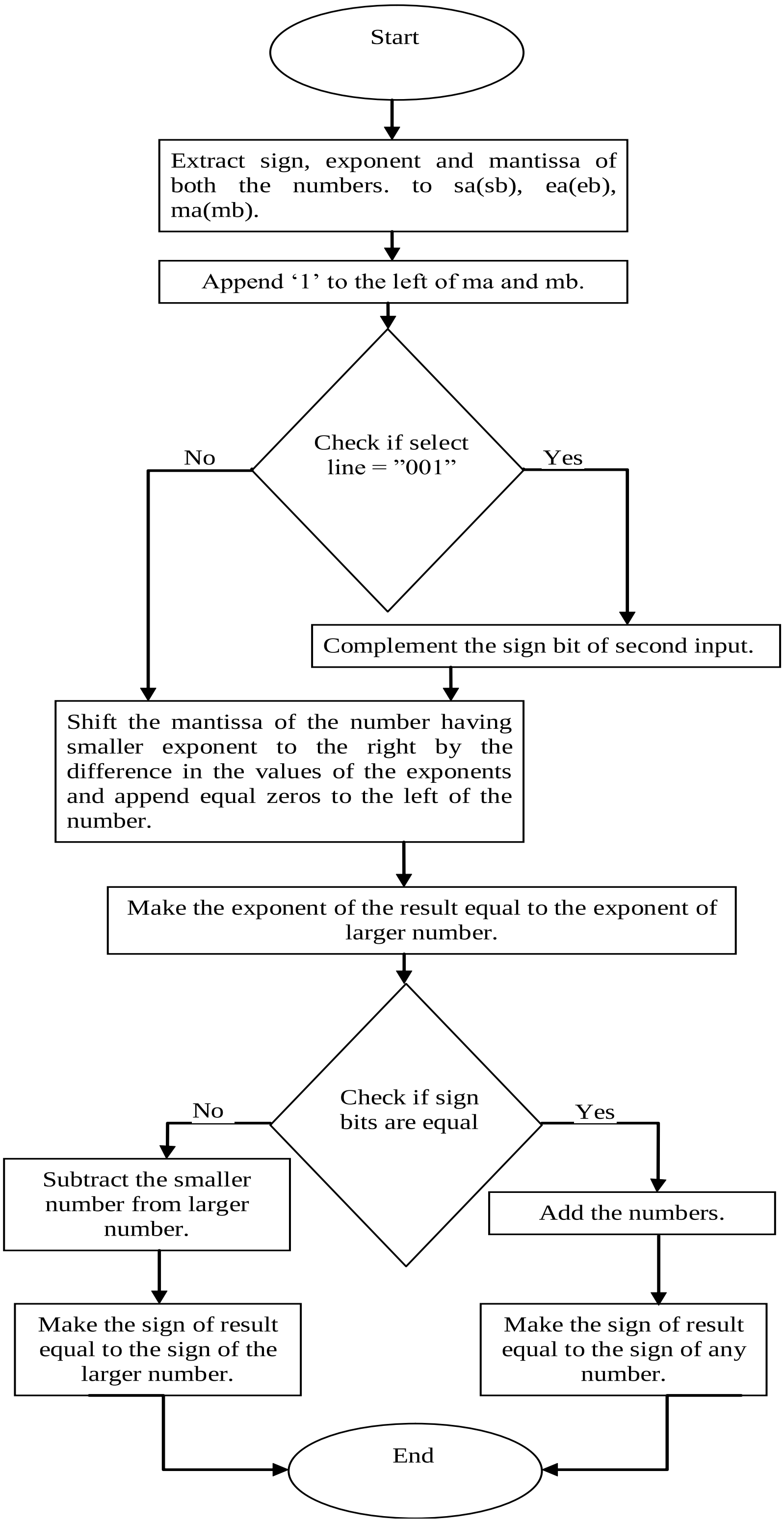

Fig. 4: Flowchart of floating point addition and subtraction.

VII. DESIGN OF LOGIC UNIT

For the implementation of Logic Unit, we considered whole 32 bit field as input to the gates. For NOT function we used first input or A as the field to be complemented. For AND, we Anded both the inputs i.e. A and B bit by bit. Similarly for OR function, we ORed the inputs bit by bit.

VIII. IMPLEMENTATION OF ARITHMETIC AND LOGIC UNIT

For the implementation of above explained multiplication, addition, subtraction and logic units, we used the tool Xilinx ISE 9.2i and wrote the program in VHDL for Virtex 4 series FPGA xc4vlx 160-12ff1148. The simulations were done using the inbuilt Xilinx ISE Simulator. The simulations were run using various examples and all the results of the simulations were verified to be correct.

IX. RESULTS AND COMPARISON

The synthesis report generated by the tool showed that the design used 3205 of the available 4 input LUT's and the slices used were 1695. The total equivalent gate count was 21,122 and the worst case delay was 45.034 ns.

For comparison purposes, we implemented another ALU based on general shift and add multiplication algorithm and similar reports were generated for it also. The 4 input LUT's that it used were 4088 and the slices used were 2099 and the total equivalent gate count was 26,852. The worst case delay in that ALU was found to be 51.820 ns.

The comparison table is as shown below:

Table 4: Comparison of results.

Parameter	ALU using Vedic Multiplier	ALU using general multiplier	Remarks
4 input LUT's used	3205	4088	27.56% less for the proposed ALU
Slices used	1695	2099	23.83% less for the proposed ALU
Total equivalent gate count	21,122	26,852	27.13% less for the proposed ALU
Delay	45.034 ns	51.820 ns	15.07 % less for the proposed ALU

X. CONCLUSION

As seen from the results the proposed ALU uses much less space in terms of 4 input LUT's used, slices used and total equivalent gate count as compared to that used by the conventional ALU. Further, in the proposed ALU the delay has also reduced substantially as shown in the table 4. Thus, from the results we can conclude that

Vedic Multiplication algorithms can be used while designing an ALU with much less space requirements and with much less delays. Further as research subjects we can include more arithmetic operations into the ALU based on Vedic Mathematics such as division, square root etc and study their impacts on total space requirements and speed of the ALU.

REFERENCES

[1] Korra Tulasi Bai, J. E. N. Abhilash,2013, "A New Novel Low Power Floating Point Multiplier Implementation Using Vedic Multiplication Techniques," International Journal of Engineering Research and Applications (IJERA), Vol. 3, Issue 4, pp.1801-1804.

[2] Rahul Bhangale, "Multiplication in Vedic Mathematics," http://mathlearners.com.

[3] Steve Hollasch, "IEEE Standard 754 Floating Point Numbers," http://steve.hollasch.net.

[4] M. Ramalatha, K. Deena Dayalan, P. Dharani, S. Deborah Priya,2009, "High Speed Energy Efficient ALU Design using Vedic Multiplication Techniques," International Conference on Advances in Computational Tools for Engineering Applications, Zouk Mosbeh, Lebanon, pp. 600-603.

[5] Yashkumar. M. Warkari, Prof L.P.Thakare, Dr. A.Y. Deshmukh, 2014, "Design of Floating Point Multiplier for Fast Fourier Transform Using Vedic Sutra," International Journal of Engineering Research and General Science, Vol. 2, Issue 3, pp. 375-382.

[6] Itagi Mahi P. and S. S. Kerur, 2013, "Design and Simulation of Floating Point Pipelined ALU Using HDL and IP Core Generator," Proceedings of National Conference on 'Women in Science & Engineering' (NCWSE 2013), SDMCET Dharwad, India, pp. 263-268.