Single Precision Floating Point Arithmetic Using Vedic Mathematics

Sarbjeet Singh[1], Ankit Trivedi[2]
[1]*ECE Department, Sant Longowal Institute of Engineering and Technology, Longowal, India*
[2]*Axis Institute of Technology and Management, India*
(E-mail: er.sarbjeet.singh@gmail.com, trisita92@gmail.com)

***Abstract*—Maneuver on the real numbers comprehensively necessitate floating-point arithmetic for expending its use in the numerous ambits of science, engineering, medical imaging, biometrics, motion capture and audio applications. Floating-point reckonings are extensively used in operations of real numbers. Computer organization has staunchly premeditated a floating-point arithmetic unit (FPU) coprocessor that stewards entirely on the operations of numbers with floating-point values and IEEE protocols are trailed by arithmetic of floating-point numbers. Single precision format specified by IEEE 754 incorporate 32-bit binary number, which have first bit (MSB) is sign bit, subsequently next 7 bits is exponent part and next 24 bits mantissa part. Infinities, underflow, overflow, inexact and NaN's are well expounded in this format. This exertion intent to form a competent FPU that is proficient in executing basic functions, reduced delay and memory constraint to supreme zenith. This paper encompasses FPU for single precision floating-point is premeditated and instigated expending Vedic mathematics. The scrupulous multiplier, Urdhva Triyakbyham sutra, discovered by Swami Jagadguru Krishna Sri Bharati Maharaja Tirthji, is expended in this work. This sutra is fundamentally vertical and crosswise multiplication of bits. With the aid of Vedic mathematics, better results have been attained when paralleled with conventional algorithms in terms of LUT's, IoB's, device utilizations and delay. The reduction of 23 percent in LUT's, 3.6 percent in IOB's and 18.7 percent in delay has been triumphed, when compared to conventional method. The design is simulated and implemented on Xilinx ISE Vivado 2014.4 by using VHDL coding and synthesised for Virtex-7. The result illustrates that FPU using Vedic mathematics has a prodigious influence on delay, speed and area.**

Keywords*— *Single precision; Floating-point arithmetic; Vedic mathematics; VHDL.

I. INTRODUCTION

Floating-point arithmetic is contemplated as a murky subject, which is quite astounding as floating-point is one of the fundamental topic in computer system. Floating-point data types are present roughly for every language of any operating system. Every range of Computers commencing from personal computers to supercomputers all are equipped with floating-point accelerators. Computer system has an FPU as its essential component. Usual actions that are manipulated by FPU are addition, subtraction, multiplication and division. Therefore, certain facet of floating-point arithmetic enacts a straight impression on the designing of systems of computer. Floating-point arithmetic has its extensively solicitations in the several fields of science, engineering, medical imaging, biometrics, motion capture and audio applications, as most of these cited fields are associated with operations on real numbers which indirect inference towards floating-point numbers. For most of mathematical operations on real numbers, floating-point operations are extensively used. In computer system gives options for number representation in two ways, either fixed point representation or floating-point representation these representations are selected contingent on user and the application where number representation is ought to be used.

One of worth mentioning benefits of using floating-point representation compared to representation of fixed-point (and integer) arithmetic is floating-point arithmetic's capability or capacity to address broader range of values. The floating-point presentation necessities vaguely additional storing space (to encode the place of the radix point), so when kept in the similar space as given to other representations, floating-point numbers attain their larger range at the cost of a little less precision [1]. Floating-point format, in specific is the standard Institute of Electrical and Electronics Engineers (IEEE) format. It is by far the most common method of representing a rough calculation to real numbers in computers because it proficiently picked up by most large computer processors.

II. FLOATING POINT REPRESENTATION

IEEE has designed an operating standard with arithmetic of floating-point i.e. IEEE 754 standard that describes floating-point representation and arithmetic. This standard is considerably used as a standard for real numbers on computers although there exist various other representations for real numbers. This standard engulf formats for representing floating-point numbers including negative numbers, de-normalized numbers special values i.e. infinities and not a number together with a set of floating-point operations being operate on special values. Four rounding modes and five exceptions are also specified in IEEE standard. The general form of the representation for floating-point number is given in equation (1):

$$Z = (-1)^S (M * 2^E) \quad (1)$$

Where

Z = any floating-point number

S = sign bit of Z

M = mantissa of Z

E = exponent of Z

Most prevalent IEEE 754 standards are equipped with single precision floating-point and double precision floating-point representation as shown in figure 1.The single precision format comprise of 32 bit binary number, which have first bit (MSB) is sign bit, then next 7 bits exponent part and next 24 bits mantissa part whereas in double precision format out of 64 bits Most significant bits is signed bit, then next 11 bits belongs to exponential part and final 52 bits are kept for mantissa part. For single precision with floating-point format and double precision with floating-point designs, mantissa that is stored with one hidden bit is represented as 1.M.

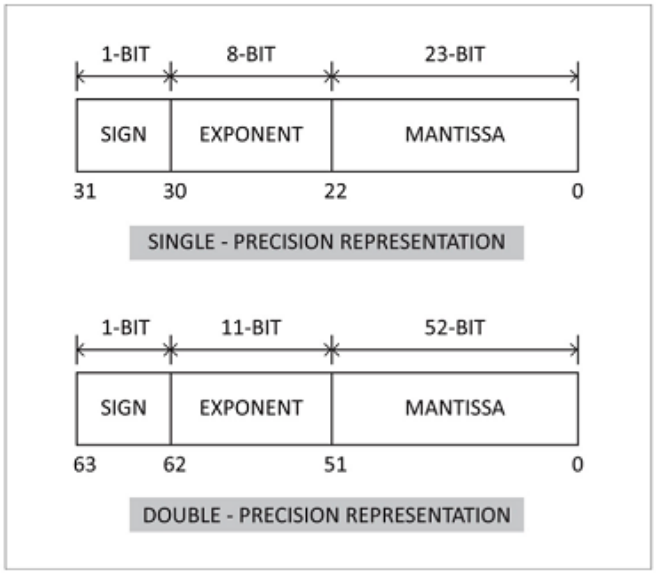

Fig. 1. Floating Point Representation

III. WORK ANTICIPATED

A. Arithmetic Unit for Floating Points

The prime aim in general for every arithmetic unit is to have substantially high speed; lower values of power consumption and efficient utilization of chip area. In this paper, single precision arithmetic unit has been anticipated which incorporates different units for addition, subtraction, multiplication and division units. In the proposed FPU as shown in figure 2, pre-normalization block is retained to adjusts the operands by executing the indispensable shifts before an addition or a subtract operation. Add block is used for execution of addition and purpose of subtraction block is subtraction of mantissa part. The purpose of Vedic multiply block is to execute the multiplication (Vedic mathematics) of the mantissa part. Divide block is used for division of the operands projected to be divided, determines the remainder. Post normalization block is employed for normalizing the outcome of add/ sub/ mul/ div operation to its IEEE754 form. Four cases of rounding discussed in this paper are: 00- even present in vicinity, 01- Zero, 10- infinity with Positive, 11- infinity with negative infinite and four cases are used for operation on fpu_operation: (000-addtion, 001-subtraction, 010-multiplication, 011-division).

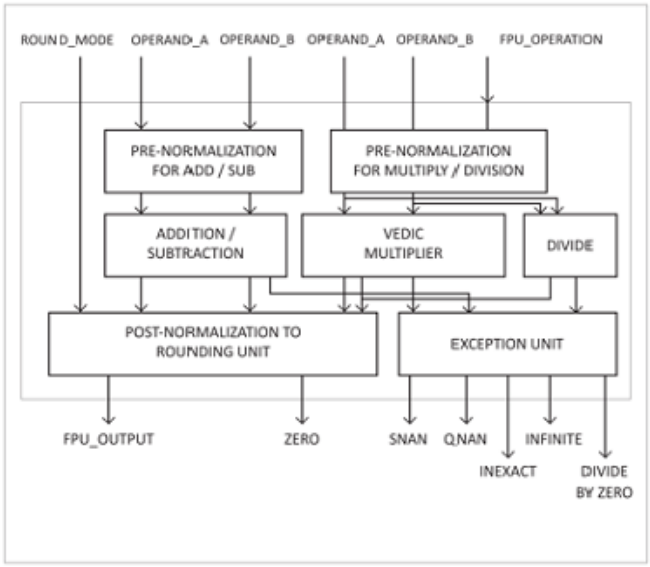

Fig. 2. Floating Point Arithmetic Unit

B. Floating Point Addition/Subtraction

The block as shown in figure 3 is used for both addition and subtraction. The steps convoluted in execution of these operations are as follows [2]:

1) Scrutinize for special values on inputs such as zero's, nan, etc.
2) Performing the necessary shifts before addition or subtraction operation to equalize exponents by modifying operands.
3) Addition and subtraction of mantissa is done accordingly
4) Normalization and rounding of the outcome are done using defined rounding modes. Additionally, exception on output is produced if any, such as NaN, Infinity, etc.

C. Floating Point Multiplier

This block is used for multiplication, which involves following steps [3]:

1) Input values are checked for a zero, a zero at input implies a zero at output.
2) Next check is done for sign of multiplication; sign is dependent on the sign of multiplying numbers. If numbers are having opposite sign, then result ought to have negative value. And if numbers have similar sign the result ought to be positive.
3) Exponent of both operands are then added and bias is subtracted bias from it then calculate exponent of result.
4) Multiplication of the mantissas.
5) Normalize the product and round the result by using specified rounding modes. Also produces exceptions.

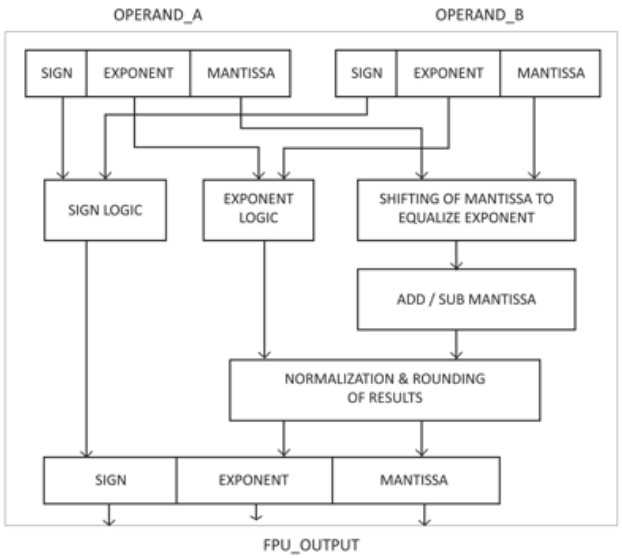

Fig. 3. Block diagram of floating-point Adder/ Subtractor

In this paper, Vedic mathematics is used for multiplication of mantissa of both operands. Vedic mathematics, originated from the Vedas, postulates one line, intellectual and tremendously fast methods along with quick crosschecking systems. Sri Bharati Krishna Tirthaji has bought in light the Vedic mathematics, and proposed that Vedic mathematics is a mathematical embellishment of sixteen simple mathematical formulae taken from the Vedas. It is based on 16 mathematical formulae (generally called Sutras) production along with several areas of mathematics such as trigonometry, arithmetical formulas, algebraic equation, geometry etc. These sutras are cited below with their concise meaning [4].

1) Ekadhikena Purvena means "one in addition to preceding one".
2) Nikhilam navatascaramam Dasatah implies "all from nine and the last from ten "
3) Urdhva – tiryagbhyam is the general formula by using vertical and crosswise multiplication and division of all numbers .
4) Paravartya Yojayet means invert and operate.
5) Sunyam Samya Samuccaye says the 'Samuccaya is also zero.' i.e., it should be kept in equal to zero.
6) Anurupye – Sunyamanyat says 'If one is present in ratio, the other one will be equal to zero'.
7) Sankalana – Vyavakalanabhyam means adding and subtracting.
8) Puranapuranabhyam means by the accomplishment or non-accomplishment.
9) Calana – Kalanabhyam means 'Sequential motion'.
10) Ekanyunena Purvena takes by one minus from previous one.
11) Gunakasamuchyah says the factors of the sum of number result are equal to the sum of the factors.
12) Gunitasamuchyah means the product of the sum result is equal to the sum of the product.
13) Sopaantyadvayamantyam means final and taken twice the last but one.
14) Vyashtisamanstih means do portion and complete.
15) Yaavadunam takes whatever the extent to fits deficiency.
16) Shesanyankena Charamena takes the remainders by the last digit.

Urdhva – tiryagbhyam sutra is employed for multiplication of 24*24 bits as shown in figure 4 and in same way, multiplication of 12*12 is done and with same Vedic sutra multiplication of 6*6 is done which is shown in figure 5.

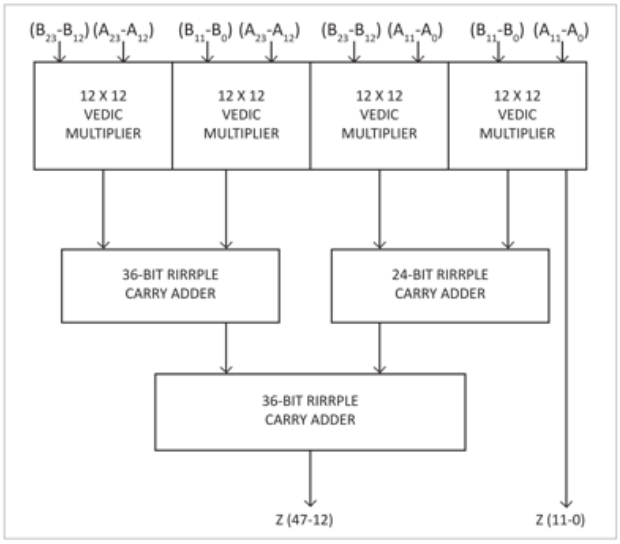

Fig. 4. Block diagram for 24*24 bit multiplier implementation

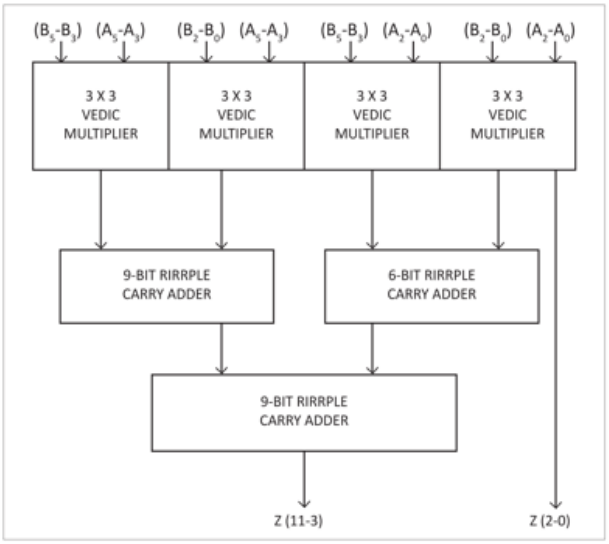

Fig. 5. Block diagram for 6*6 bit multiplier implementation

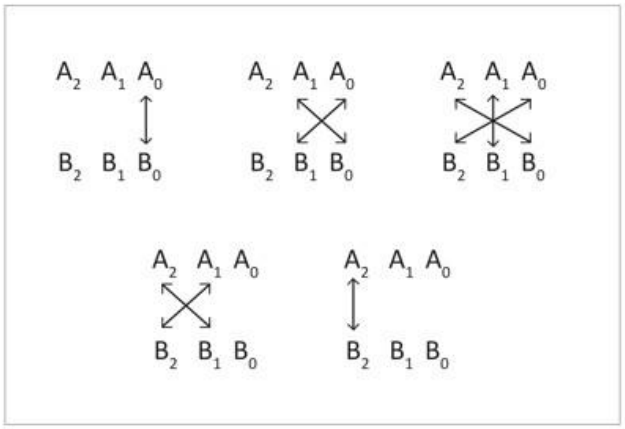

Fig. 6. Implementation of 3*3 bit multiplication

In 3*3 Vedic multiplication, first step is to multiply operand's last significant bits as A0*B0 [5]. In second step is evaluation of multiplication of A1*B0 and B1*A0 and adding the results. In the third step, multiplication of A2*B0, B2*A0 and A1*B1 respectively finally adding them all. In fourth step, multiplication of A2*B1 and B2*A1 then adding. In final step, multiplication of A2*B2 is done to get the required result. The hardware implementation of 3*3 multiplication is shown above in figure 6.

D. Floating Point Division

This block is used for division that involves following steps [6].

1) Inspect for zeros, nan, infinity on inputs.
2) Adding the exponents
3) Divide the mantissas

Normalize the product and round by the specified rounding mode. Also produce exceptions.

IV. RESULT AND ANALYSIS

The FPU has been coded in VHDL, simulated on Xilinx Vivado 2014.4 and synthesized for Virtex-7 FPGA [7]. The maximum combinational path delay of floating-point multiplier is 25.494 ns and fmax is at 100MHz [8]. The various results of arithmetic operation as shown below. As it can be anticipated from the figure 7, figure 8, figure 9 and figure 10, the various simulation results, which shows simulation of subtraction of two 32-bit floating-point number which is shown in figure 8. The round mode is 10 and the fpu_operation mode is 001. Figure 9 shows the simulation result of multiplication of two 32-bit floating-point numbers. The round mode is 01 and the fpu_operation mode is 010. Similarly figure 10 shows the simulation result of division of two 32-bit floating-point numbers. The round mode is 00 and the fpu_operation mode is 011

V. CONCLUSION

The designed FPU has capable of performing different operations such as addition, subtraction, multiplication and division. Addition, subtraction and division operations are implemented using conventional method. The multiplication is based on Vedic Mathematics. Xilinx VIVADO 2014.4 is used for designing of FPU with VHDL coding and is synthesized for Virtex-7 FPGA. The purposed FPU is effective for less memory utilization as shown by results. The future of this research lies with a vast amount of work yet to be done for improvising the efficiency of the FPU by using other Vedic sutras.

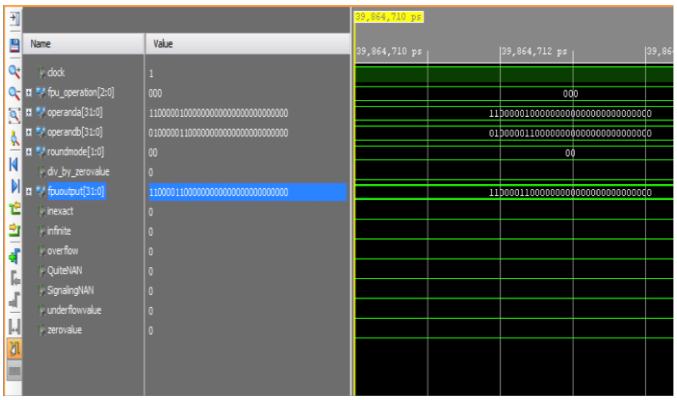

Fig. 7. Simulation result of addition of two 32-bit floating-point number. The round mode is 00 and the fpu_operation mode is 000

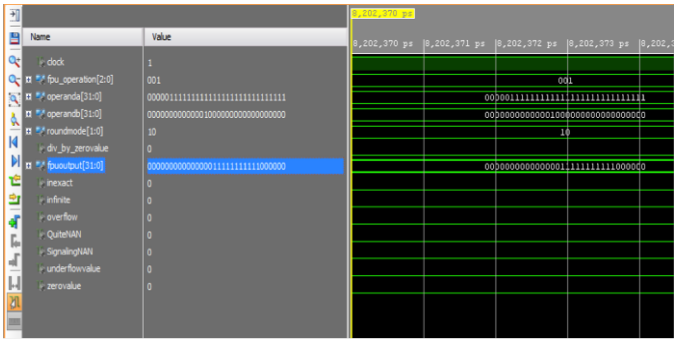

Fig. 8. Simulation result of subtraction of two 32-bit floating-point number. The round mode is 10 and the fpu_operation mode is 001

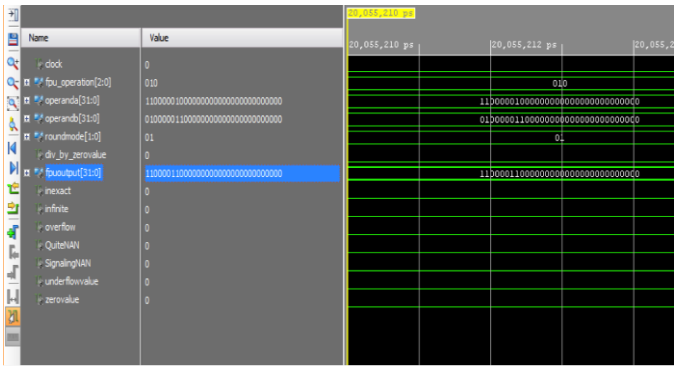

Fig. 9. Simulation result of multiplication of two 32-bit floating-point number. The round mode is 01 and the fpu_operation mode is 010

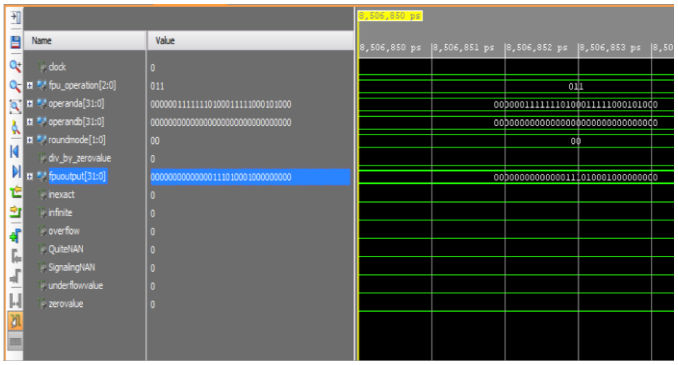

Fig. 10. Simulation result of division of two 32-bit floating-point number. The round mode is 00 and the fpu_operation mode is 011

REFERENCES

[1] S. S. Mahakalkar and S. L. Haridas, "Design of high-performance IEEE 754 floating point multiplier using Vedic mathematics, "in Computational Intelligence and Communication Networks (CICN), 2014 International Conference on. IEEE, 2014, pp. 985-988.

[2] Y. Bamsal, C. Madhu and P. Kaur, "High speed Vedic multiplier designs- a review," in Engineering and Computational Sciences (RAECS), 201 Recent Advances in. IEEE, 2014, pp. 1-6.

[3] I.Vaibhav, K. Saicharan, B. Sravanthi and D. Srinivasulu, "VHDL implementation of floating-point multiplier using Vedic mathematics, "in International Conference on Electrical, Electronics and Communications (ICEEC), 2014.

[4] S. B. K. Tirtha and V. S. Agrawala, Vedic mathematics. Motilal Banarsidass Publication, 1992, vol.10.

[5] A. Jain, B. Dash, A. K. Panda, and M. Suresh, "FPGA design of a fast 32-bit floating point multiplier unit, "in Devices, Circuits and Systems (ICDCS), International Conference on. IEEE, 2012, pp. 545-547.

[6] Y. S. Rao, M. Kamaraju, and D. Ramanjaneyulu, "An FPGA implementation of high speed and area efficient double precision floating point multiplier using Urdhva Triyagbhyam technique," in Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG), Conference on. IEEE, 2015, pp. 271-276.

[7] W. Jose, A. R. Silva, H. Neto and M. Vestias, "Efficient implementation of a single precision floating point arithmetic unit on FPGA," in Field Programmable Logic and Applications (FPL), 24th International Conference on. IEEE, 2014, pp. 1-4.

[8] R. K. Kodali, L. Boppana and S. S. Yenamachintala, "FPGA implementation of Vedic floating point multiplier," in Signal Processing, Informatics Communication and Energy Systems (SPICES), Conference on. IEEE, 2015, pp-1-4.