iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** VIII **Month of publication:** August 2017

DOI: http://doi.org/10.22214/ijraset.2017.8020

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

ternational Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VIII, August 2017- Available at www.ijraset.com

Double-Precision FP Butterfly Architecture with Reduced Delay by Using BSD Representation

Mohammed Parvez Ahmed[1], V. B. K. L. Aruna[2]
[1](*Studying M.Tech. in VLSI & ES specialization, Velagapudi Ramakrishna Siddhartha Engineering College, Jawaharlal Nehru Technological University -Kakinada, Kanuru, Vijayawada-7, Andhra Pradesh)*
[2](*Assistant Professor, Electronics and Communication Engineering department, Velagapudi Ramakrishna Siddhartha Engineering College, Jawaharlal Nehru Technological University -Kakinada, Kanuru, Vijayawada-7, Andhra Pradesh)*

Abstract: ***For many years FFT coprocessors has been hot topic of research, as they have a significant impact on the performance of communication systems. It mainly consists of butterfly units that are dubbed from consecutive multiply add operations over complex numbers. Nowadays applying floating-point arithmetic to butterfly units has become more popular. It reduces the compute-intensive tasks from general-purpose processors by dismissing FP related tasks. However, the major disadvantage of FP butterfly unit is its delayed calculations compared to its counterpart fixed point numbers. Hence there is a need to develop a faster FP butterfly FP butterfly architecture. A faster FP butterfly unit using a fused-dot product-add (FDPA) unit, to compute AB±CD±E, based on BSD representation. The proposed FDPA unit mainly consists of FP constant multiplier and a BSD adder. Hence the speed of the FDPA unit is improved. The modified Booth encoding is also used to accelerate the BSD multiplier. After synthesizing the proposed FP butterfly architecture, it is concluded that the unit is much faster than the previous counterparts.***

Keywords: FFT, Butterfly unit, Floating-point, Fixed-point, Single precision, Double precision, redundant number system, three operand addition.

I. INTRODUCTION

By using Floating point numbers the real numbers can be represented in the binary format; there are two different floating point number formats according to IEEE 754 standard

A. Binary interchange format
B. Decimal interchange format.

The main critical requirement for DSP applications is multiplying floating point numbers that involve large dynamic range. The main focus in this paper is only on double precision normalized binary interchange format. The IEEE 754 double precision binary format representation is shown in the Fig.1;

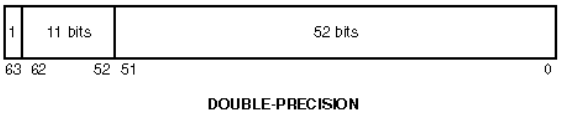


Fig. 1(a) Double Precision Floating point

It consists of one sign bit (S), an eleven bit exponent (E), and a fifty two bit fraction (M or Mantissa). An extra bit called as significand 1 is added to the fraction. If the exponent is greater than 0 and smaller than 2047, and if there is 1 in MSB of the significand then number is a normalized number. To multiply the two floating point numbers:

C. The exponent part of the two numbers is added and then the bias is subtracted from the result
D. The significand part along with the normalized 1 of the two numbers is multiplied
E. The sign of the product is calculated by XORing the sign of the two numbers

The final multiplication result is represented as a normalized number by keeping 1 in the MSB of the result that is also called as leading one.

Due to the tiny power consumption of CMOS circuits the huge chips can be fabricated. As the much of the power is dissipated as heat, and chips have very less heat dissipation capacity, power consumption is very critical at the chip level. Though the system in

which a chip is located can provide vast amounts of power, most of the chips are packed to dissipate less than 10 to 15 watts of power before they are damaged permanently.

There are several consecutive multipliers and adders in FFT circuit over complex numbers. To accommodate this, an appropriate number representation must be chosen. The fixed-point arithmetic is been used in most of the FFT architectures, until recently they are replaced by floating-point numbers [2], [3]. By using FP arithmetic over fixed-point arithmetic a vast dynamic range is introduced; but on the other hand cost is increased. Using IEEE-754-2008 standard for FP arithmetic, there will be an FFT coprocessor in collaboration with general purpose processors [4]. These reduce the computer-intensive tasks from the processors and also higher performance is achieved. However, the major disadvantage of FP butterfly unit is its delayed calculations compared to its counterpart fixed point numbers. Hence there is a need to develop a faster FP butterfly FP butterfly architecture. One way to reduce the delay is by merging several operations into a single FP unit, and hence delay, area and power consumption are saved. Another method is to use redundant number systems to overcome FP slowness, as there is no word-wide carry propagation.

A number system, defined by a radix r and a digit-set $[a, b]$ is redundant iff $b-a+1>r$ [5].

To convert a number form no-redundant to redundant format no carry is generated, but the reverse operation needs carry propagation. Hence when there are many consecutive arithmetic operations then redundant format is more beneficial. A butterfly architecture that uses redundant FP arithmetic is proposed in this paper. It is useful for floating point FFT coprocessors and also contributes to DSP applications. There are other works that use redundant FP number systems, but they are not utilized for butterfly architecture.

II. FAST FOURIER TRANSFORM (FFT)

The FFT is a very complex computation as there is intensive access of the memory and parallel calculations are to be done very frequently. To realize FFT algorithm in VLSI there should be pipelined architecture and parallelism. The multiplicative complexity should be as low as possible at algorithm level. The delay-feedback buffering strategy is to be used to reduce the memory size at algorithm level. There should be modular and regular models, local routing and very little control complexity. The DFT X(k) of an N-point sequence x(n) given by

$$X(k)=\sum_{n=0}^{N-1} x(n) W_N^{nk}; \qquad k=0,1,2\ldots\ldots..N-1 \qquad (1)$$

$$W_N^{nk}= \cos\left(\frac{2\pi.nk}{N}\right)- j.\sin\left(\frac{2\pi.nk}{N}\right) \qquad (2)$$

In equation (2) W_N^{nk} is called as twiddle factor.

The very first step on the algorithm level is to select an FFT radix which is mainly a trade-off between area, speed and power. As the radix increases the implementation and control becomes difficult. The number of butterfly elements in radix-2 FFT is more when compared to radix-4 FFT. Hence radix-4 is selected in this paper. The Fig.2 (a) shows the better understanding of radix-4 FFT algorithm

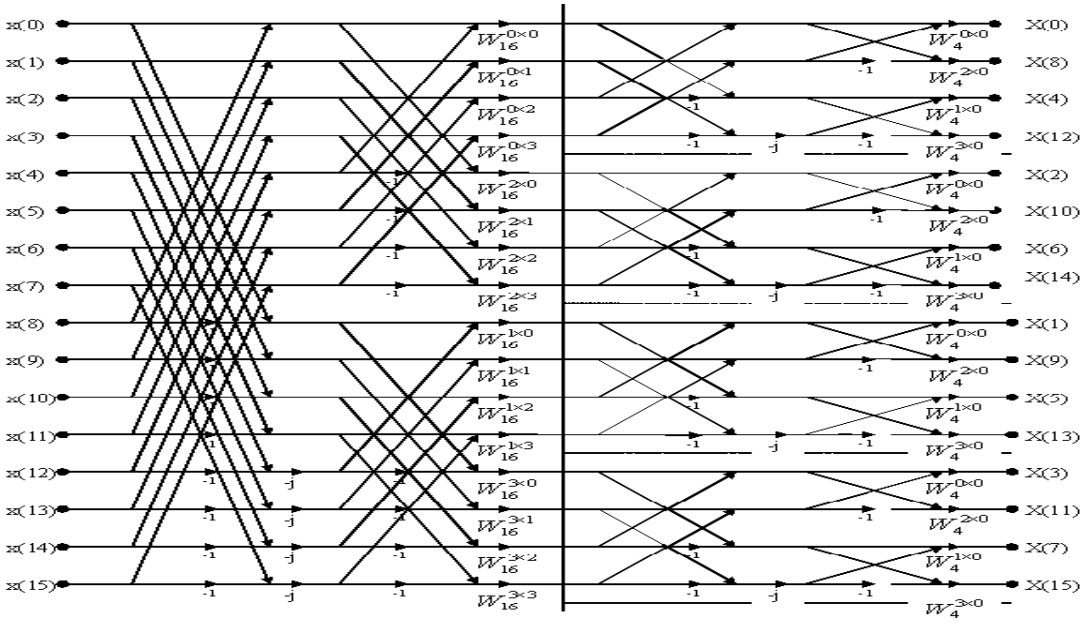


Fig. 2(a) Signal flow graph of 16 point radix-4 FFT

To implement complex multiplication of –j the real and imaginary parts of the incoming data are exchanged and then the sign of the imaginary part is inverted. The signal flow graph is mapped to the architecture as shown in the figure 2. By using Cooley and Turkey's algorithm, a DFT can be decomposed into successive smaller DFTs. The equations for the algorithm of high speed pipelined DIT FFT architecture is given by

$$\left[X(k)\ , X(k+\frac{N}{4}), X(k+\frac{N}{2}), X(k+\frac{3N}{4})\right]^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^0 F_0(k) \\ W_N^k F_1(k) \\ W_N^{2k} F_2(k) \\ W_N^{3k} F_3(k) \end{bmatrix}$$

Here

$F_{n1}(k) = \sum_{n_2=0}^{(n/4)-1} x\,(n_1 + 4n_2) W_{N/4}^{n_2.k}$

for n=0,1,2,3; k=0,1,2…….$\frac{N}{4} - 1$.

The twiddle factors can also be calculated and listed in figure 2(a). They are stored in the RAM with 16-bit fix point in the hardware.

A. Butterfly Unit

A butterfly unit is a portion of the computation that combines the results of smaller FFT into a larger FFT, or vice versa. The Butterfly Diagram builds on the Danielson-Lanczos Lemma and the twiddle factor to create an efficient algorithm.

A FP FFT butterfly architecture is made of FP multipliers followed by FP adders/subtractors, consider AB+CD+E

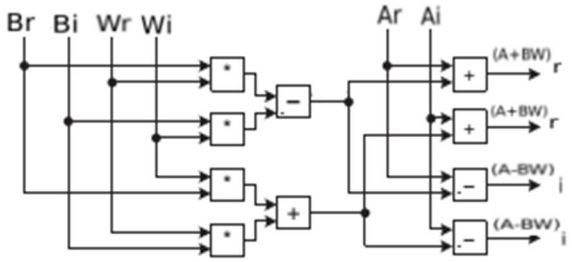


Fig. 2(b) FFT Butterfly Architecture

B. FFT for composite of two integers

The DFT of a vector $(x_0 \ldots x_{N-1}) \in C^N$ is given by

$$\widehat{x_k} = \sum_{j=0}^{n-1} x_j\, e^{-2\pi i \frac{jk}{n}}$$

As the pre factor 1/N is normal in all, it is usually omitted. To turn the one dimensional formulation of the DFT into a two dimensional one the following change of variables are used

$j=j(a, b) = aN_1+b$; $0 \le a < N_2$, $0 \le b < N_1$

$k=k(c, d) = cN_2+d$; $0 \le c < N_1$, $0 \le d < N_2$

It follows

$x_j = x(a, b)$, $\widehat{x_k} = \hat{x}(c,d)$ and $W_N = e^{-\frac{2\pi j}{N}}$:

$\hat{x}(c,d) = \sum_{a=0}^{N_2-1} \sum_{b=0}^{N_1-1} x(a,b)\ W_N^{(aN_1+b)(cN_2+d)}$

$= \sum_{b=0}^{N_1-1} W_N^{b(cN_2+d)} \sum_{a=0}^{N_2} x(a,b)\ W_{N_2}^{ad}$

$\hat{x}(b,d) = \sum_{a=0}^{N_2} x(a,b)\ W_{N_2}^{ad}$

Since $W_N^{acN_1N_2} = W_N^{acN} = 1$ and $W_N^{adN_1} = W_{N2}^{ad}$, this can be treated as calculating the first N DFT- values with length N2 and then calculating N DFT- values with length N1 with new data, But arithmetic complexity of O(N.N1 + N.N2) increases, but compared to the direct approach it is better. Due to this, it can be stated that the number of complex multiplications needed for a DFT is ten lakh sample points with the direct and FFT approach.

C. Binary Signed Digit

A binary signed number is a ternary number, such that each bit carries its own sign. By using this method carry-ripple path can be eliminated but at the expense of data conversion. This delay is less compared to carry propagation.

There is no chance of putting a + or – sign to a number in digital circuits, as they are operated in binary numbers. The binary numbers are measured in terms of bits and bytes (8 bits). One byte can have values ranging from 0 to 255, I.e. 256 different combinations of bits can be formed. Mathematical numbers are normally made up of a sign and a value in which '+' indicates a positive number and '–' indicates negative number. In binary numbers, if signed then the MSB bit represents the positivity or negativity of a number. If the MSB is 0 then the number is positive and if the MSB is 1 then the number is negative. This is called "Sign magnitude representation". For an n-bit signed binary number n^{th} bit represents the sign of the number and the remaining n-1 bits represent the magnitude.

Multiplication is a very crucial operation in digital applications. It consists of mainly two steps:

1) Partial product generation

2) Partial product addition

Hence the efficiency and delay of multiplication can be improved by either reducing the number of partial products or speeding up the addition process. The partial product also depends on the numbering system used to represent the operands. By using the conventional binary number system the delay is increased because of the dependency on carry chain propagation. But by using the BSD representation the delay can be fairly reduced. Several BSD encoding techniques are proposed in the related literatures

3) 2-bit encoding with two's complement representation

4) 2-bit encoding with positive bit and negative bit representation

5) 3-bit encoding with 1-out-of-3 representation

III. BUTTERFLY ARCHITECTURE

Based on an efficient algorithm an FFT could be realized in hardware, in which an N-input FFT is simplified into two N/2-input FFTs. By continuously decomposing leads to 2-input FFT block, known as butterfly unit. The proposed butterfly unit is a complex fused-dot-product-add operation with FP operands [1]. The required operations for butterfly unit are a dot-product followed by an addition operation which leads to the proposed FDPA operation. The exponents of all the inputs are taken, and the significands are represented in BSD. In BSD every binary position takes values of {-1, 0, 1} I.e. there is a negative-weighted bit and a positive-weighted bit. The carry-limited addition circuitry for BSD numbers is shown in the figure 3(a).

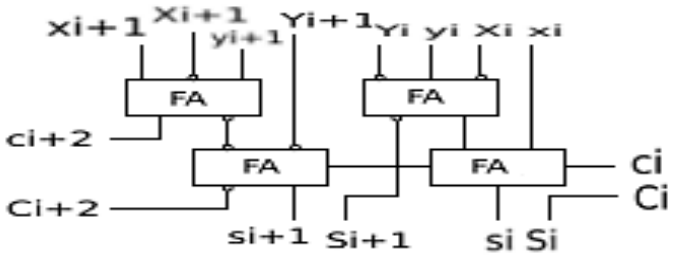


Fig. 3(a) BSD Adder (2-digit slice)

Here the capital letter represents negative bits and small letter represents positive bits. The main critical path of this adder is three full-adders.

The FDPA unit mainly consists of

A. Redundant FP Multiplier

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VIII, August 2017- Available at www.ijraset.com

1) *Partial Product Generation:* The PPG used here is completely different from the other normally used multipliers. To reduce the number of partial products the multiplier W_i and W_r are stored in modified booth encoding with radix-4 technique [5]. By using this technique we can reduce the number of partial products to n/2 (n is the number of bits in multiplier) whereas by using conventional multiplier number of partial products would be n. These are generated according to the modified booth encoding.

W_{i+1}	W_i	W_{i-1}	PP
0	0	0	0
0	0	1	+B
0	1	0	+B
0	1	1	+2B
1	0	0	-2B
1	0	1	-B
1	1	0	-B
1	1	1	0

Table 1: Modified Booth Encoding in Radix-4

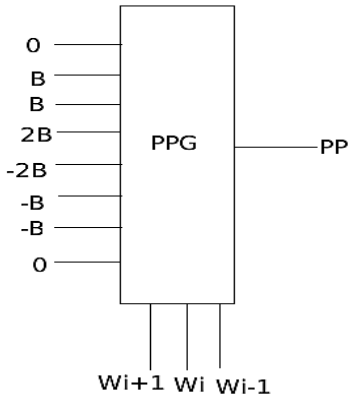


Fig. 3(b) Partial Product Generator

2) *Partial Product Addition:* The partial products so obtained are added by using BSD addition. Every partial product is shifted two bits left with respect to the previous partial product and then they are added.
3) *Partial Product Reduction:* In double precision floating point architecture we get total of 107 bits after addition of partial products, but we can only save up to 52 bits in mantissa, so the remaining bits are to be discarded. The four MSB bits are discarded which includes the normalized one and the last 51 bits are discarded. The bits from 52 to 103 are stored in the mantissa.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VIII, August 2017- Available at www.ijraset.com

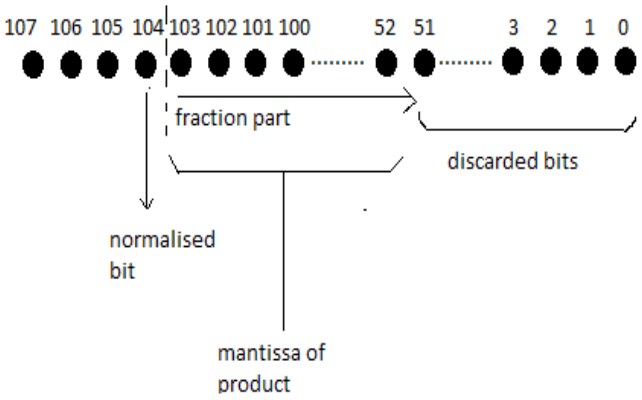


Fig. 3(c) Partial Product Reduction

4) *Exponent Addition:* The exponents of the multiplier and multiplicand are added by using a normal carry propagation adder and the bias is subtracted as bias gets added two times. The exponent is stored in the product floating point number.
5) *Setting the Sign Bit:* If one of the multiplier and multiplicand is positive and the other is negative then the product is negative, but if both are either positive or negative then product is positive. This can be accomplished by XOR gate. The sign bit is set by XORing the sign bits of multiplier and multiplicand.

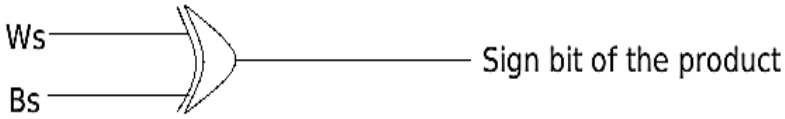


Fig. 3(d) Sign setting

The overall multiplier unit is given by

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VIII, August 2017- Available at www.ijraset.com

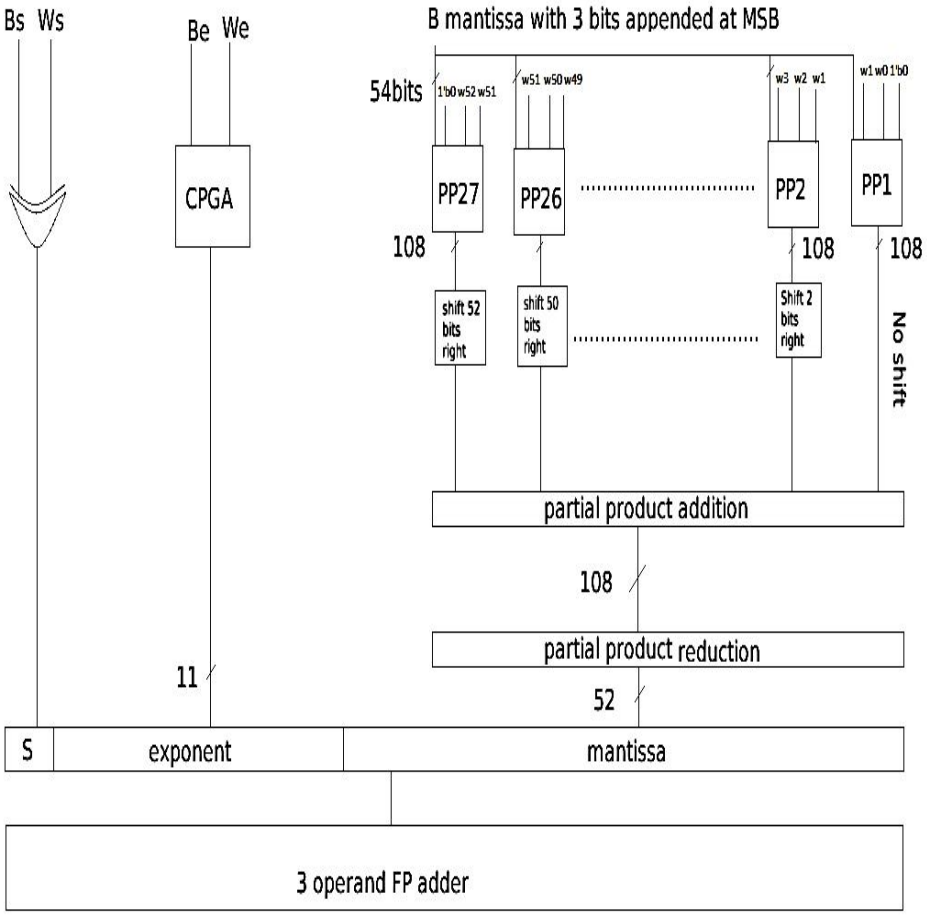


Fig. 3(e) Multiplier Block

B. Floating Point Three-Operand Adder

The obvious method to add three floating point numbers is by using concatenated two operand adder, but this leads to high latency, area and power consumption. So in order to overcome this three operand FP adder is used [7], [8].

First of all the operand with the bigger exponent among the three operands is determined using a comparator as shown in the figure below:

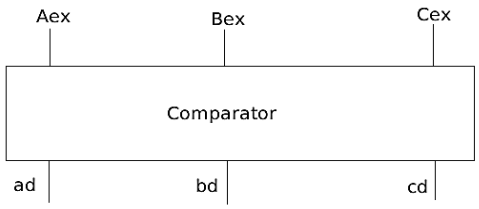


Fig. 3(f) Comparator of the Exponents

Then the operands are assigned in ad, bd and cd respectively depending on the size of their exponents.

The exponent of the second operand bd is subtracted from the exponent of the first operand ad and bd is shifted right equal to the difference. Similarly the exponent of third operand cd is subtracted from the exponent of the first operand ad and cd is shifted right equal to the difference.

If the MSB of 'ad, bd and cd ' is 1 then the two's complement of the mantissas is stored. If the MSB is 0 then the mantissa is stored as it is. This is accomplished by using a 2×1 MUX, the MSB is used as select bit. Then the mantissas of all the three operands are added by using BSD addition.
Then the three operands are added as shown in the figure below:

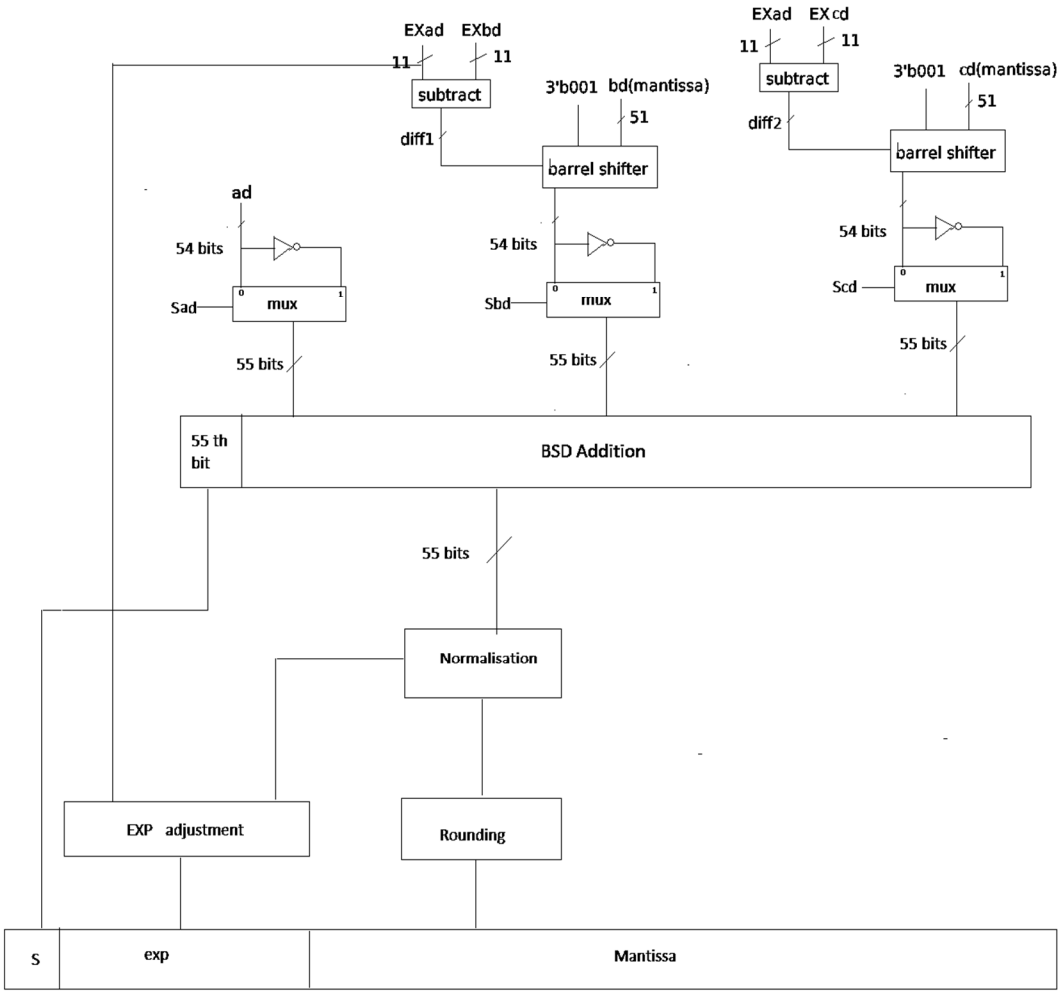


Fig. 3(g) 3-operand FP adder

C. BSD Addition

1) Sign extension bits 0 and 1 are added to the MSB of each operand depending on positive and negative number respectively.
2) Then X and Y are calculated such that X= ad | bd | cd, Y= ad & bd & cd .
3) Then from the LSB of Y the bits are inverted as soon as a 1 is observed in Y.
4) The inversion is stopped when a 0 is observed in X.
5) This process is repeated until the MSB of X and Y.
6) The converted number is stored in Yb.
7) And finally the sum is given by S= X xor Yb [9].

IV. EVALUATION AND COMPARISONS

A. Delay Comparisons of Normal multiplier and Modified Booth Multiplier for Single-Precision Floating-Point

A normal multiplier was also designed for single precision floating-point multiplier and the results are compared.

BLOCK	DELAY
Normal Multiplier	7.866nS
Modified Booth Multiplier	7.236nS

Table 2: Delay comparison of different multiplier blocks

B. Delay Comparisons of Concatenated 2-Operand Adder and 3-Operand Adder for Single Precision Floating-Point

A truncated 2-operand adder is also designed for single precision and its delay is compared with the 3-operand adder

BLOCK	DELAY
Truncated 2-operand adder	29.430nS
3-operand addition	14.739nS

Table 3: Delay comparisons of two adders

C. Overall Comparison of Butterfly Unit for Single-Precision Floating-Point

A butterfly unit with normal multiplier and truncated 2-operand adder is designed in single precision floating point; it is referred to as butterfly unit 1. Its delay is compared with its counterpart which is referred to as butterfly unit 2.

BLOCK	DELAY
Butterfly unit 1	33.487nS
Butterfly unit 2	19.269nS

Table 4: Delay comparisons of two butterfly units

From the above comparisons, it is observed that Butterfly unit with modified booth multiplier and 3-operand adder has less delay compared to the Butterfly unit with normal multiplier and truncated 2-operand adder.

Hence in double precision floating point the butterfly unit with less delay is designed and the final delay observed is 45.760nS

V. CONCLUSION

A high-speed FP butterfly architecture is proposed, which is faster than previous works because:

A. Modified Booth multiplier that reduces the number of partial products and also delay

B. BSD representation of the significands which eliminates carry-propagation

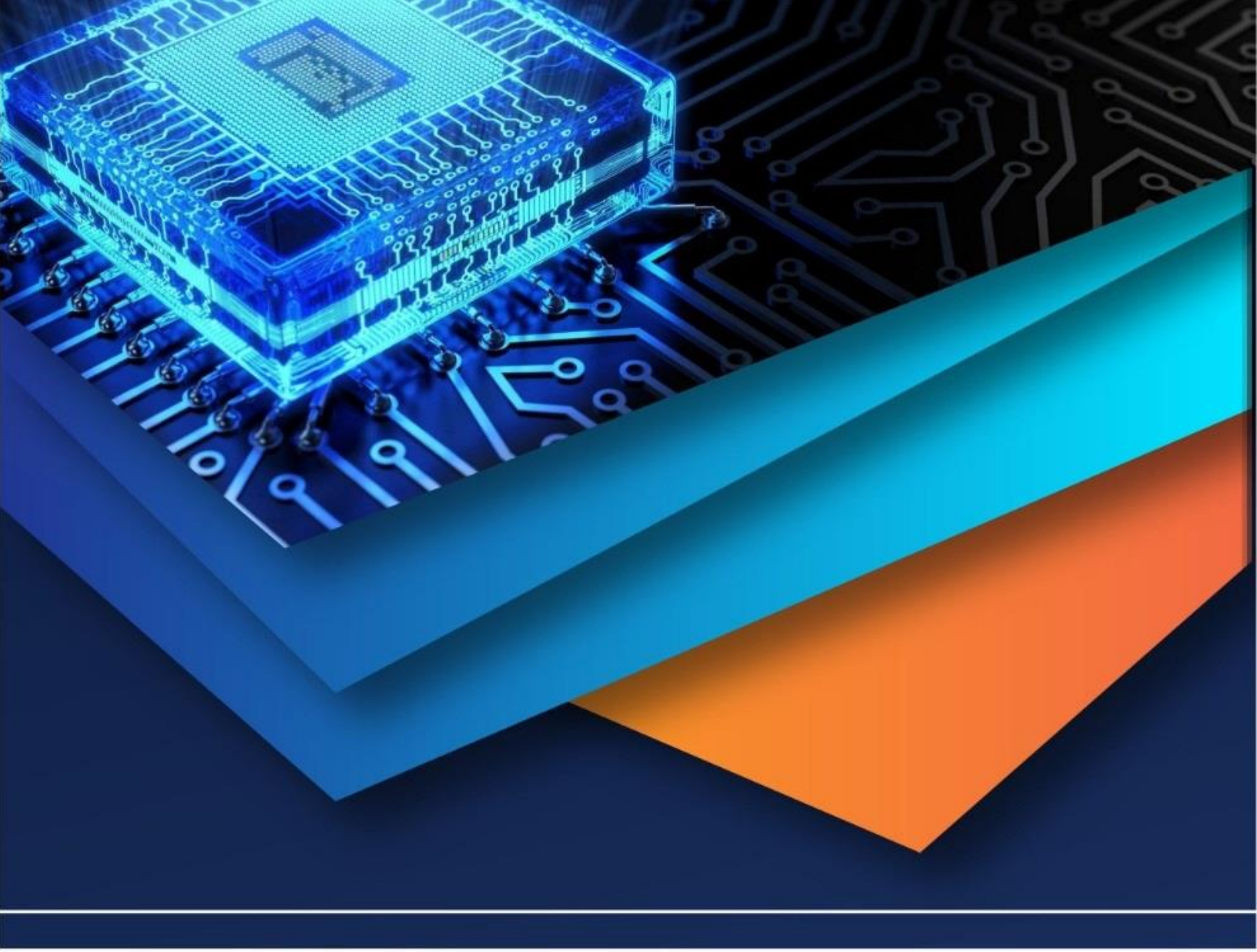
C. The new FDPA unit proposed in this brief

The proposed unit integrates the multiplications and additions that are necessary in FP butterfly. Hence the higher speed is achieved

The overall delay for Floating point double precision butterfly unit is 45.760nS

REFERENCES

[1] Amir Kaivani and Scokbum Ko, “Floating point Butterfly Architecture Based on Binary Signed-Digit Representation,” IEEE Trans., March 2016.
[2] E. E. Swartzlander, Jr., and H. H. Saleh, “FFT implementation with fused floating-point operations,” IEEE Trans. Comput., vol. 61, no. 2, pp. 284–288, Feb. 2012.
[3] Sohn and E. E. Swartzlander, Jr., “Improved architectures for a floating-point fused dot product unit,” in Proc. IEEE 21st Symp. Comput. Arithmetic, Apr. 2013, pp. 41–48
[4] IEEE Standard for Floating-Point Arithmetic, IEEE Standard 754-2008,Aug. 2008, pp. 1–58.
[5] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs,2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.
[6] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” Math. Comput., vol. 19, no. 90, pp. 297–301, Apr. 1965.
[7] A.F. Tenca, “Multi-operand floationg-point addition.” in Proc. 19th IEEE Symp. Comput. Arithmetic, Jun. 2009, pp. 161-168
[8] Y. Tao, G. Deyuan, F. Xiaoya, and R. Xianglong, “Three-operand floating-point adder,” in Proc. 12th IEEE Int. Conf. Comput. Inf. Technol., Oct. 2012, pp. 192–196.
[9] R. Hashemian, “Fast addition using a new number system,” IEEE 30th Asilomar Conf. Signals, Systems and Computers, Nov. 1996, pp.884-888.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY