Floating-Point Butterfly Architecture Based on Carry Select Adder Representation with Improvement in Speed Computation

Easam. Swapna Priya Sindhuja[1], A.Hari Prasad[2], P.Anil Kumar[3]
Student of M.TECH (DSCE), Dept. of ECE, KITS(S)
Associate Professor, Dept. of ECE, KITS(S)
Associate Professor, Dept. of ECE, KITS(S)

***Abstract*-Fast Fourier transform (FFT) coprocessor, having a significant impact on the performance of communication systems, and in DSP processors has been a hot topic of research in recent years. The FFT function is consists of consecutive multiply add operations over complex numbers, dubbed as butterfly units. Applying floating-point (FP) arithmetic to FFT architectures, specifically butterfly units, has become more popular recently. It offloads compute-intensive tasks from general-purpose processors by dismissing FP concerns (e.g., scaling and overflow/underflow). However, the major downside of FP butterfly is its slowness in comparison with its fixed-point counterpart. This reveals the incentive to develop a high-speed FP butterfly architecture to mitigate FP slowness. This brief proposes a fast FP butterfly unit using a devised FP fused-dot-product-add (FDPA) unit, to compute AB±CD±E, based on binary-signed-digit (BSD) representation. The FP three-operand BSD adder and the FP BSD constant multiplier are the constituents of the proposed FDPA unit. A carry-limited BSD adder is proposed and used in the three-operand adder and the parallel BSD multiplier so as to improve the speed of the FDPA unit. Moreover, modified Booth encoding technique is used to accelerate the BSD multiplier. The synthesis results show that the proposed FP butterfly architecture is faster than previous counterparts but requires more area.**

***Index Terms*- Carry Select Adder (CSA) representation, butterfly unit, Fast Fourier Trans-form (FFT), Floating-Point (FP), three-operand addition. BSD (Binary significant digit) adder.**

I. INTRODUCTION

Fast Fourier transform (FFT) circuitry consists of several consecutive multipliers and adders over complex numbers, hence an appropriate number representation must be chosen. Most of the FFT architectures have been using fixed-point arithmetic, until recently that FFTs based on floating-point (FP) operations are introduced. The main advantage of FP over fixed-point arithmetic is the wide dynamic range but it introduces at the expense of higher cost. Moreover, use of IEEE-754-2008 standard for FP arithmetic allows for an FFT coprocessor in collaboration with general purpose processors. This offloads compute-intensive tasks from the processors and leads to higher performance. The main drawback of the FP operations is their slowness in comparison with the fixed-point counterparts. A way to speed up the FP arithmetic is to merge several operations into a single Floating point unit, and hence save the delay, area, and power consumption. Using redundant number systems is another well-known way of overcoming Floating point slowness, where there is no word-wide carry propagation within the intermediate operations. Hence the overall delay of the circuit is reduced.

II.LITERATURE SURVEY

The Sequential Fast Fourier Transform:The Discrete Fourier Transform is an operation performed on a series of elements to convert the underlying signal in one domain to another domain (e.g., time to frequency or vice- versa). The result has many useful applications(DSP applications) and is one of the most widely used algorithms of the 20th and 21st centuries . The typical DFT operation performed on N elements $x_1, x_2 \ldots x_N$ is defined as

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k \frac{n}{N}}$$

As a result of the input range of each element of the array X is an additive that requires the cooperation of

every element of x. Thus, an N- DFT operation element in the input array O (N2). In the case of a large array of applications, compution of DFT is difficult and requires more time. Therefore, DFT is time-complexible algorithm, it is desirable to reduce the complexity of DFT algorithm. In order to reduce the computational complexity of DFT the scientist Cooley and Tukey introduces the Fast Fourier algorithm in 1965, to transform, to significantly reduce the complexity of computing discrete Fourier transform a divide and conquer approach is used in the algorithm which is nothing but a Fast Fourier Transform.The computational complexity is reduced in FFT algorithm compared to that of DFT algorithm. FFT method, we have no input in the range of a DFT (NlogN) to be able to count. However, for many years after its introduction, the development of some of the technology, "Cooley-Tukey algorithm" was held, and the theoretical aspects of the research FFT algorithm, or FFT analysis focused on the details of the proposals.

Cooley-Tukey Algorithm:

Cooley-Tukey algorithm, as introduced, accounting (both additions and multiplications) to calculate the discrete Fourier transform to reduce the need to use of divide and conquer approach. The Cooley-Tukey algorithm, outlined here, radix-2 D (decimation-in-time), and for a simple FFT algorithm serves as our base. This FFT algorithm introduces the butterfly architecture. To calculate an N point FFT requires N/2 butterflies.

The basic steps in the FFT algorithm:

- Decimate - two (i.e. source 2) of DFT to create small, the split of the original input into set of even and odd parts.(Divides the the samples as even and odd).
- Multiply each element of the source of the unity of the coalition (called twiddle factors)W.
- Butterfly - (see figure 1), the other with a small element of the corresponding and add each element of each of the DFT. Repeat procedure for N=4,8 ….

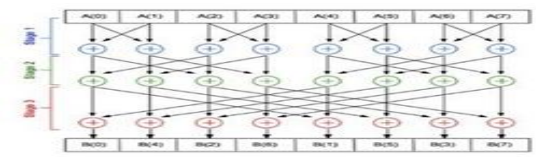

Fig.1. *Recursive process of radix 2 DIT-FFT*

Existing systems in FFT:

An FFT processor chip was originally designed. This chip is specifically used in Orthogonal frequency division multiplexing. OFDM FFT processor is the focal point of both the transmitter and receiver. FFT high performance, combined with low energy consumption rate, and which has high- throughput approach to the implementation of an ASIC is computationally demanding operation. Which is computationally intensive requires less number of additions and subtractions.

Architecture:

The FFT and IFFT has the property that, if FFT (Re(xi)+ jIm(xi)) = Re(Xi)+ jIm(Xi) and IFFT(Re(Xi)+ jIm(Xi)) = Re(xi)+ jIm(xi) where xi and Xi are N words long sequences of complex valued, samples and sub-carriers respectively, then 1/N * FFT(Im(Xi)+ jRe(Xi)) = Im(xi)+ jRe(xi). Therefore, it is necessary not only to discuss the implementation of the FFT equalizer.

To calculate the inverse transform, the real and imaginary part of the input and output are changed. N 1 / n scaling a power of two, so that the right binary word log2 (N) bits in the same switch. Even simpler, binary point left log2 (N) bits is shifted to remember. If ever, did not show up to change a bit, depending on how it is used, the output from IFFT, is required.

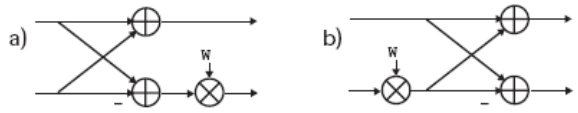

Fig.2 : *A radix-2 DIF butterfly (a) and a radix-2 DIT butterfly (b), where W is the twiddle factor.*

FFT algorithm is the basic building block can be realized with butterfly operation. Each butterfly performs addition and multiplication operations along with the twiddle factor The forward and inverse operations are possible with FFT known as DIT FFT and DIF FFT Decimation in time and decimation in frequency respectively, the two are shown in figure 2. DIF is the difference between the before and after the addition or subtraction and multiplication of the featured twiddle factor is in place. FFT based on the divide and conquer and due to the input range is N R source, known as the point length N = RP, so, and p positive integer is the most effective. An N-point FFT, to count the butterflies are connected to the p stages.

The map is an example of a hardware-point radix-made N = 16 -2 DIF FFT is shown in figure 2(a). The input data, x (N), the output of data occurs in a random order, however, X (N) to return to observe it. Reversed the order of the data generated to re-order bit is known and described by the figure. figure 2(b) FFT bit reversed order, which will result in the reshaping of the input and output, so it is possible to form the natural order. figure 2(a), moving vertically, parallel data paths and reconfigure the product to the natural order, the way to control cross connections while, and think. As well as all the arrows in figure 2(b), if one turned around from the FFT, DIF FFT is performed instead. So the decimation in frequency and decimation in time operations are performed and vice versa.

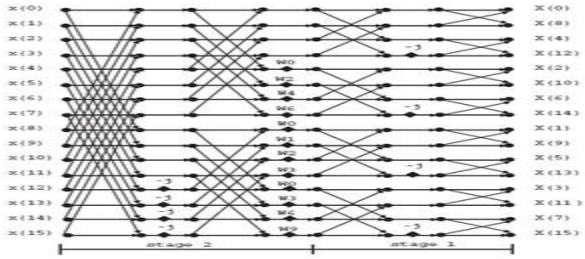

Fig.3*: N = 16-point radix-22 DIF FFT algorithm*

III.PROPOSED BUTTERFLY ARCHITECTURE

In this section we presents a butterfly architecture using redundant Floating Point arithmetic, which is useful for FP FFT coprocessors and contributes to the digital signal processing applications and also in Orthogonal frequency division multiplexing applications. Although there are other existing algorithms works on the use of redundant Floating point number systems, which are not optimized and which requires more delay. In butterfly architecture in which both redundant FP multiplier and adder are required. The novelties FDPA and BSD adder techniques used in the proposed design include the following

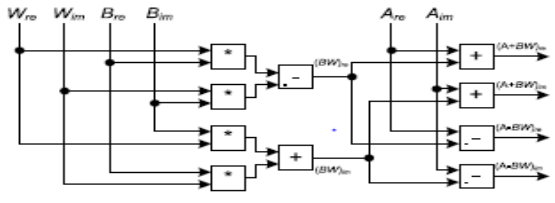

Fig.4: *FFT butterfly architecture with expanded complex numbers*

1) All the significants in the design of Floating point butterfly are represented in binary signed-digit (BSD) format and the corresponding carry-limited(which avoids carry propagation from one stage to next stage) adder is designed.

2) Design of the FP constant multipliers for operands with BSD significant.

3) Design of FP the three-operand adders for operands with BSD significant.

4) Design of the Floating point fused-dot-product-add (FDPA) units (i.e., AB ± CD ± E) for operands with the BSD significant format.

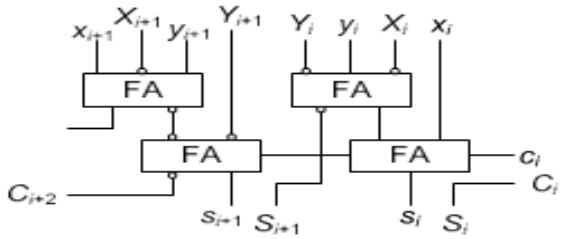

Fig.5: *BSD adder (two-digit slice)*

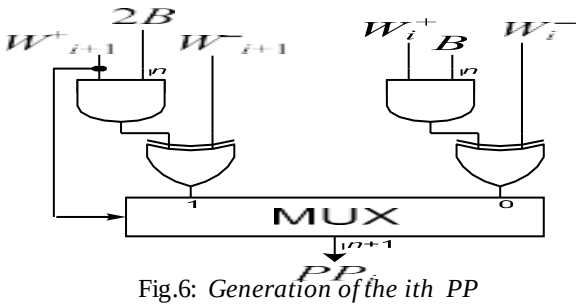

Fig.6: *Generation of the ith PP*

The FFT could be implemented in hardware based upon an efficient algorithm in which the N -input FFT computation is simplified to the computation of two (N /2)-input FFT. Continuing this decomposition of FFT that is divide and conquer leads to 2-input FFT block, also known as butterfly unit. The proposed butterfly unit is actually a complex fused-multiply– add with FP operands. Expanding the complex numbers, figure 4 shows the required modules. According to figure 5 and figure 6, the constituent operations for butterfly unit are a dot-product (e.g. $B_{re}\, W_{im} + B_{im}\, W_{re}$) followed by an addition/subtraction along with the twiddle factors which leads to the proposed FDPA operation(e.g., $B_{re}\, W_{im} + B_{im}\, W_{re} + A_{im}$). Implementation details of FDPA, over FP operands. The exponents of all the inputs are added(E1+E2+….) and represented in its two's

complement (after subtracting the bias), while the significant of Are, Aim, Bre, and Bim are represented in BSD.Which perform XOR operation between the MSB to get the Sign of the resultant number. Within this representation every binary position takes values of {−1, 0, 1} represented by one negative-weighted bit (negabit) and one positive-weighted bit (posibit).

Proposed Redundant Floating-Point Multiplier:

The proposed multiplier, is similar to that of other parallel multipliers, consists of two major steps, namely, partial product generation (PPG) and Partial product reduction (PPR). However, contrary to the conventional multipliers, the proposed multiplier keeps the product in redundant format and hence there is no need for the final carry-propagating adder. There is no carry propagation from one stage to the next stage. The exponents of the input operands are taken care of in the same way as is done in the conventional FP multipliers, however, normalization, deletion and rounding are left to be done in the next block of the butterfly architecture (i.e., three-operand adder).

1.Partial Product Generation: The Partial product generator step of the proposed multiplier is completely different from that of the conventional multiplier one because of the representation of the input operand bits (B, W , B^r, W^r).Moreover, given that W_{re} and W_{im} are constants, the multiplications in figure 8(over significant) can be computed through a series of shifters , adders.

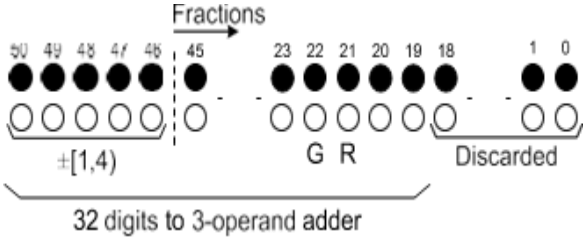

Fig.7: *Digits to three-operand adder.*

TABLE I
GENERATION OF A PP

$W_{i+1}^- W_{i+1}^+$	$W_i^- W_i^+$	$\|\|W_{i+1}^- W_{i+1}^+ W_i^- W_i^+\|\|$	PP_i
0 0	0 0	0	0
0 0	0 1	1	B
0 0	1 1	−1	$-B$
0 1	0 0	2	$2 \times B$
1 1	0 0	−2	$-2 \times B$

2.Partial Product Reduction: The main advantage of the PPR step is the proposed in this section in which the carry-limited addition over the operands represented in BSD format. This carry-limited addition circuitry is shown in fig.5(two-digit slice). Since each PP (PP_i) is (n + 1)-digit (n,…, 0) which is either B (n − 1, …, 0) or 2B (n,…, 1), the length of the final product may be more than 2n.This is possible by use of BSD adders in the intermediate stages.

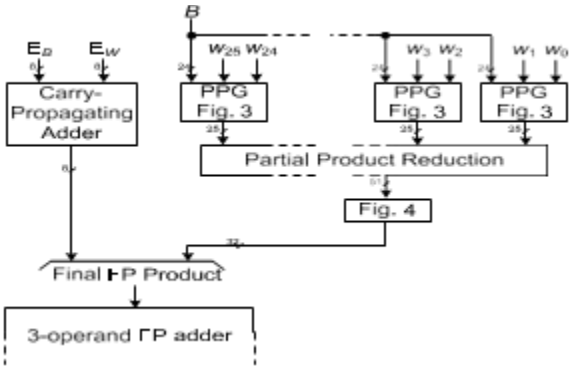

Fig.8: *Proposed redundant FP multiplier*

Assuming that the sign-embedded significant bits of inputs A and B (24 bits) are represented in BSD, while that of W is represented in modified Booth encoding technique (25 bits), the last PP has 24-(binary position) width (instead of 25), given that the most significant bit of W is always 1.The reduction of the PPs is done in four levels using 12 BSD adders .Given that B is in ±[1, 2) and in [1, 2), the final product is i±[1, 4) and would fit into 48 binary position (47…0). Consequently, positions 45 down to 0 are fractions. Similar to standard binary representation, Guard (G) and Round (R) positions are sufficient for correct rounding(Rounding is used to place the decimal point). Therefore, only 23 + 2 fractional binary positions of the final product are required to guarantee the final error $<2^{-23}$. Selecting 25 binary positions out of 46 fractional positions of the final product dismisses positions 0 to 20. However, the next step addition may produce carries to G and R positions. Nevertheless, because of the carry-limited BSD addition, contrary to standard binary addition, only we consider the positions 20 and 19 may produce such carries.

In overall, positions 0 to 18 of the final product are not useful and hence a simpler PPR tree is possible. Fig. 4 shows the required digits passed to the three-operand adder. Figure 8 shows the proposed redundant FP multiplier.

Proposed Redundant Floating-Point Three-Operand Adder:

The straightforward approach to perform a

three-operand FP addition is to concatenate two FP adders which leads to high latency, power, and area consumption. A better way is to use fused three-operand FP adders .In the proposed three-operand FP adder, a new alignment block is implemented and CSA–CPA are replaced by the BSD adders (Figure 5). Moreover, sign logic is eliminated. The bigger exponent between E_X and E_Y (called EBig) is deter- mined using a binary subtractor ($6 = E_X - E_Y$); and the significants of the operand with smaller exponent (X or Y) is shifted "6"-bit to the right. Next, a BSD adder computes the addition result (SUM = X + Y), using the aligned X and Y . Adding third operand (i.e., SUM + A) requires another alignment. This second alignment is done in a different way so as to reduce the critical path delay of the three-operand adder. First, the value of $6_A = EBig - E_A + 30$ is computed which shows the amount of right shifts required to be performed on extended (with the initial position of 30 digits shifted to left). Figure 8 shows the alignment of length of bits is implemented in the proposed three-operand FP adder. Next, a BSD adder adds the aligned third significant digit (58-digit) to SUM (33-digit) is generated from the first BSD adder.

Since the input operands have different number of digits, this adder is a simplified 58-digit BSD adder. Next steps are normalization and rounding, which are done using conventional methods for BSD representation . It is to be denoted that of the leading zero detection (LZD) block could be replaced by a four-input leading-zero-anticipation for speed up the process but which requires the of more area occupation. The other modification would replace our single path architecture with the dual path to sacrifice area and speed.

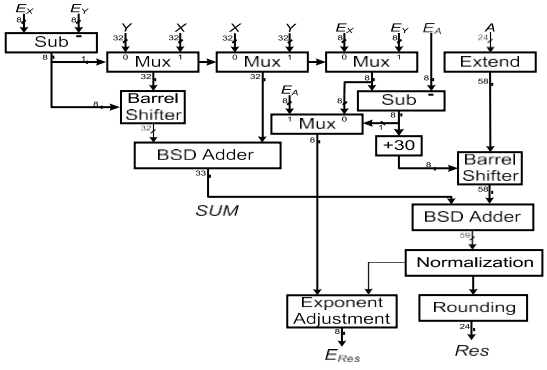

Fig.9: *Proposed FP three-operand addition*

IV. SIMULATION RESULTS

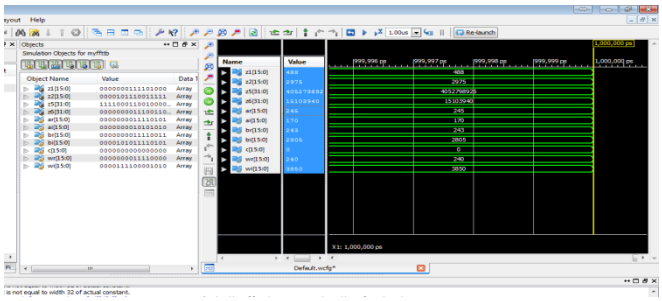

Fig.10: *simulation result*

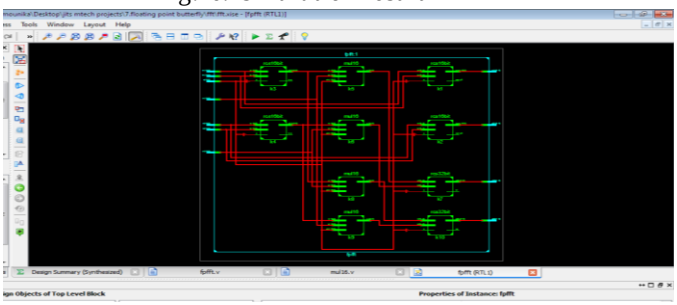

Fig.11: *Schematic of proposed method*

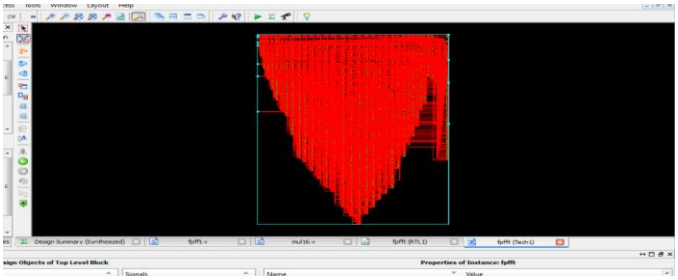

Fig 12: *Technology schematic of proposed method*

V. CONCLUSION

In this paper we have here in the processor FFT and FFT butterfly architecture, reading, writing and execution addresses. But the high-risk in the area, the advantage is which is faster than the previous works, we , proposed high-speed FP butterfly architecture. The reason behind this is the speed of development is twofold: FP butterfly eliminates the carry propagation from one stage to another stage between the significands and it is possible by 1) BSD representation, and 2) The use of proposed new FDPA unit. FP butterfly multiplications and additions need to be combined with the single FP unit.Thus, high-speed additional LZD, normalization , deleting and rounding can be achieved by FP units.So,the overall delay is reduced. The next research FP adder three dual line method applied to the structure and other unnecessary FP. FP representations may be planning on using for the efficient area utilization. Moreover, the design stage of the abolition of the use

of improved techniques (ie, repeating LZD, normalize,deletion and rounding) of the estimated costs, however, lead to faster architectures but requires more area.

REFERENCES

[1] E. E. Swartzlander, Jr., and H. H. Saleh, "FFT implementation withfused floating-point operations," IEEE Trans. Comput., vol. 61, no. 2,pp. 284–288, Feb. 2012.

[2] J. Sohn and E. E. Swartzlander, Jr., "Improved architectures for afloating-point fused dot product unit," in Proc. IEEE 21st Symp. Comput.Arithmetic, Apr. 2013, pp. 41–48.

[3] IEEE Standard for Floating-Point Arithmetic, IEEE Standard 754-2008,Aug. 2008, pp. 1–58.

[4] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs,2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.

[5] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculationof complex Fourier series," Math. Comput., vol. 19, no. 90, pp. 297–301,Apr. 1965.

[6] A. F. Tenca, "Multi-operand floating-point addition," in Proc. 19th IEEESymp. Comput. Arithmetic, Jun. 2009, pp. 161–168.

[7] 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.

[8] A. M. Nielsen, D. W. Matula, C. N. Lyu, and G. Even, "An IEEE compliant floating-point adder that conforms with the pipeline packet forwarding paradigm," IEEE Trans. Comput., vol. 49, no. 1, pp. 33–47,Jan. 2000.

[9] P. Kornerup, "Correcting the normalization shift of redundant binary representations," IEEE Trans. Comput., vol. 58, no. 10, pp. 1435–1439,Oct. 2009.

[10] 90 nm CMOS090 Design Platform, STMicroelectronics, Geneva, Switzerland, 2007.

[11] J. H. Min, S.-W. Kim, and E. E. Swartzlander, Jr., "A floating-point fused FFT butterfly arithmetic unit with merged multiple-constant multipliers," in Proc. 45th Asilomar Conf. Signals, Syst. Comput., Nov