FP Fused –Dot Product Add Unit based on Binary Signed Digit Representation

V. Kalaiselvi
Applied Electronics/ECE
Gnanamani College of technology

R. Lakshmi
AP/ECE
Gnanamani College of Technology

***Abstract*:- Fast Fourier transform (FFT) coprocessor, having a significant impact on the performance of communication systems, has been a hot topic of research for many years. The FFT function consists of consecutive multiply add operations over complex numbers, dubbed as butterfly units. Applying floating-point (FP) arithmetic to FFT archi- tectures, 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 is used to accelerate the BSD multiplier. The synthesis results show that the proposed FP butterfly architecture is much faster than previous counterparts but at the cost of more area.**

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 wisely. Most of the FFT architectures have been using fixed-point arithmetic, until recently that FFTs based on floating-point (FP) operations grow [1], [2]. The main advantage of FP over fixed-point arithmetic is the wide dynamic range it introduces; but at the expense of higher cost. Moreover, use of IEEE-754-2008 standard [3] 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 in a single FP unit, and hence save delay, area, and power consumption [2]. Using redundant number systems is another well-known way of overcoming FP slowness, where there is no word-wide carry propagation within the intermediate operations.A number system, defined by a radix r and a digit-set $[\alpha, \beta]$, is redundant iff $\beta - \alpha + 1 > r$ [4].

II. LITERATURE REVIEW

Literature review reveals that many leading libraries all books can be requested online, An important problem in the realization of floating-point subtraction is the identification of the position of the first nonzero digit in a radix-represented number**.**This paper describes two fused floating-point operations and applies them to the implementation of fast Fourier transform (FFT) processors. The fused operations are a two-term dot product and an add-subtract unit. The FFT processors use “butterfly” operations that consist of multiplications, additions, and subtractions of complex valued data. Both radix-2 and radix-4 butterflies are implemented efficiently with the two fused floating-point operations. When placed and routed using a high performance standard cell technology, the fused FFT butterflies are about 15 percent faster and 30 percent smaller than a conventional implementation. Also the numerical results of the fused implementations are slightly more accurate, since they use fewer rounding operations.

III . FFT BUTTERFLY ARCHITECTURE

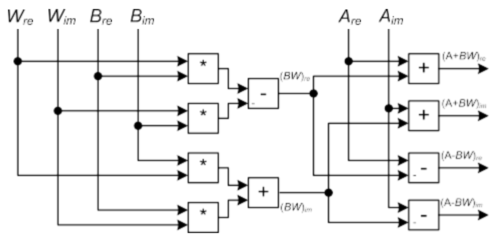


Fig. 1. FFT butterfly architecture with expanded complex numbers

The conversion, from nonredundant, to a redundant format is a carry-free operation, however, the reverse conversion requires carry- propagation [4]. This makes redundant representation more useful where many consecutive arithmetic operations are performed prior to the final result. This brief proposes a butterfly architecture using redundant FP arithmetic, which is useful for FP FFT coprocessors and contributes to digital signal processing applications. Although there are other works on the use of redundant FP number systems, they are not optimized for butterfly architecture in which both redundant FP multiplier and adder are required. The novelties and techniques used in the proposed design include the following.

1) All the significands are represented in binary signed- digit (BSD) format and the corresponding carry-limited adder is designed.
2) Design of FP constant multipliers for operands with BSD significands.
3) Design of FP three-operand adders for operands with BSD significands.
4) Design of FP fused-dot-product-add (FDPA) units (i.e., AB ± CD ± E*)* for operands with BSD significands.

The rest of this brief is organized as follows. Section II is devoted to the proposed FP butterfly architecture while the evaluations are discussed in Section III. Finally, the conclusion is drawn in Section IV.

A. PROPOSED BUTTERFLY ARCHITECTURE

The FFT could be implemented in +hardware based on an efficient algorithm [5] in which the *N* -input FFT computation is simplified to the computation of two (*N* /2)-input FFT. Continuing this decomposition 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, Fig. 1 shows the required modules.According to Fig. 1, the constituent operations for butterfly unit are a dot-product (e.g., *Bre Wim Bim Wre)* followed by an addition/subtraction which leads to the proposed FDPA oper- ation (e.g., *Bre Wim Bim Wre Aim)*. Implementation details of FDPA, over FP operands, are discussed below.

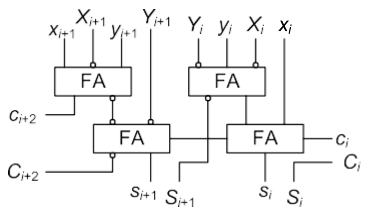


Fig. 2. BSD adder (two-digit slice).

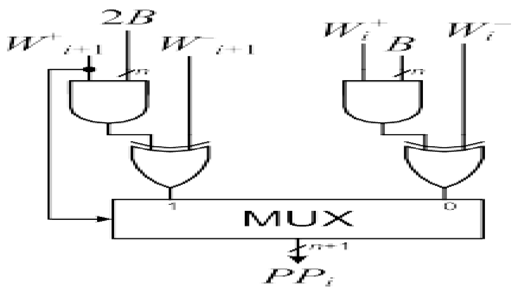
The exponents of all the inputs are assumed and represented in two’s complement (after subtracting the bias), while the significands of *Are, Aim, Bre,* and *Bim* are represented in BSD. Within this repre-sentation every binary position takes values of {1, 0, 1} representedby one negative-weighted bit (negabit) and one positive-weighted bit (posibit).The carry-limited addition circuitry for BSD numbers is shown in Fig. 2, where capital (small) letters symbolizes negabits (posibits). The critical path delay of this adder consists of three full-adders.The proposed FDPA consists of a redundant FP multiplier followed by a redundant FP three-operand adder.

B. PROPOSED REDUNDANT FLOATING-POINT MULTIPLIER

The proposed multiplier, likewise other parallel multipliers, consists of two major steps, namely, partial product generation (PPG) and PP reduction (PPR). However, contrary to the conventional multipliers, our multiplier keeps the product in redundant format and hence there is no need for the final carry-propagating adder.The exponents of the input operands are taken care of in the same way as is done in the conventional FP multipliers; however, normalization 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 PPG step of the proposed multiplier is completely different from that of the conventional one because of the representation of the input operands (*B*, *W* , *B*^{j}, *W*^{j}*)*.

Moreover, given that *Wre* and *Wim* are constants [5], the multiplica- tions in Fig. 1 (over significands) can be computed through a series of shifters and adders.With the intention of reducing the number of adders, we store the significand of *W* in modified Booth encoding [4].Given the modified Booth representation of *Wre* and *Wim* , one PP, selected from multiplicand *B*, is generated per two binary positions of the multiplier *W* , Fig. 3. Generation of the *i*th PP



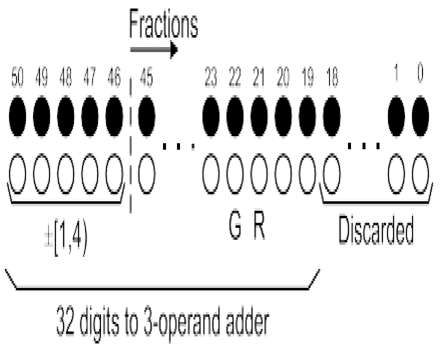


Fig. 4. Digits to three-operand adder.

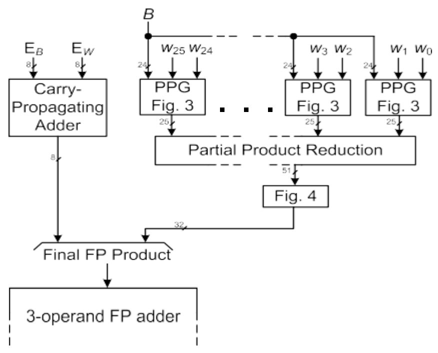


Fig. 5. Proposed redundant FP multiplier.

Assuming that the sign-embedded significands of inputs A and B (24 bits) are represented in BSD; while that of W is represented in modified Booth encoding (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 W in [1, 2), the final product is [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. Therefore, only 23 2 fractional binary positions of the final product are required to guarantee the final error $<2^{-23}$. Selecting 25 binary positionsout 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 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.Fig. 5 shows the proposed redundant FP multiplier.

C.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 ieee transactions on very large scale integration (vlsi) systems.

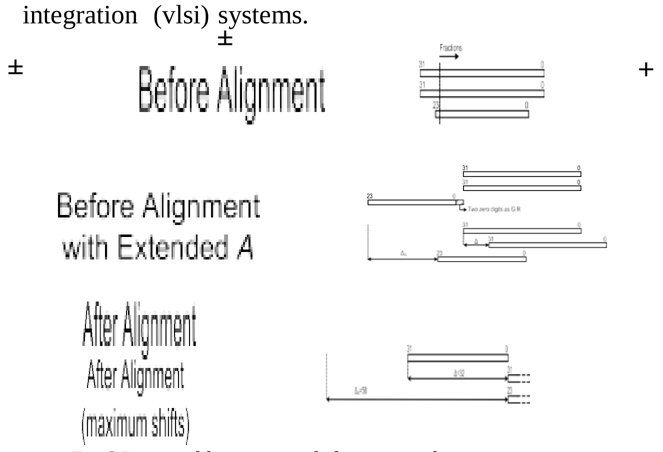

±
±
+
Before Alignment
Fractions
Before Alignment with Extended A
After Alignment
After Alignment (maximum shifts)

Fig.6.Proposedthree-operand alignment scheme

high latency, power, and area consumption. A better way is to use fused three-operand FP adders [6], [7].In the proposed three-operand FP adder, a new alignment block is implemented and CSA–CPA are replaced by the BSD adders (Fig. 2). Moreover, sign logic is eliminated.The bigger exponent between E_X and E_Y (called EBig*)* is determined using a binary subtractor (O E_X E_Y); and the significand of the operand with smaller exponent (X or Y) is shifted O -bitto the right. Next, a BSD adder computes the addition result (SUMX 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 ofO_A EBig E_A 30 is computed which shows the amount of right shifts required to be performed on extended A (with the initial position of 30 digits shifted to left). Fig. 6 shows the alignmentsimplemented in the proposed three-operand FP adder.Next, a BSD adder adds the aligned third significand (58-digit) to SUM (33-digit) 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 [8], [9]. It should be noted that the leading zero detection (LZD) block could be replaced by a four-input leading-zero-anticipation [2] for speed up but at the cost of more area consumption. The other modification would replace our single path architecture with the dual path to sacrifice area for speed.The proposed FP three-operand adder is implemented as shown in Fig. 7 in which new alignment and addition blocks are introduced. Moreover, due to the sign-embedded representation of the significands (i.e., BSD), a sign logic is not required.

A comparison of the proposed design with the conventional one is shown in Table II.The critical path of the three-operand adder, according to Fig. 7, consists of two 8-bit carry-propagating subtractors (0.25 ns each), a MUX (0.07 ns), a 30 block (0.17 ns), a barrel shifter (0.29 ns), and the final BSD adder (0.16 ns); plus normalization and rounding (0.75 ns) and registers (0.22 ns).The conversion, from nonredundant, to any redundant format is a carry-free operation, however, the reverse conversion requires carry-propagation.Given that the proposed butterfly architecture is meant to be used in an FFT unit, the reverse conversion is done in the very last iteration of the FFT unit. Therefore, the latency of each stage is equal to that of a butterfly unit plus those of registers. Moreover, a lookup table is required to store the constant values of 256 twiddle factors.There might be a need for one more step in the end to con- vert BSD result to nonredundant representation.

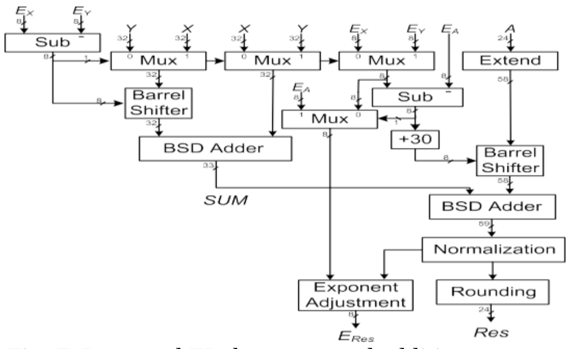
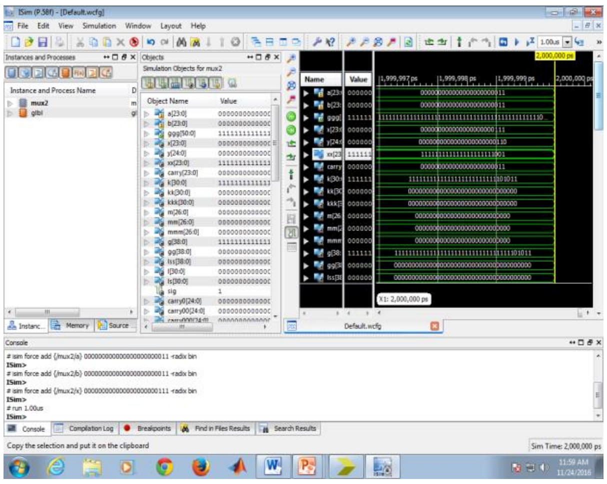


Fig. 7. Proposed FP three-operand addition.

D.EVALUATIONS AND COMPARISON

The proposed design is synthesized by Synopsys design compiler using the STM 90-nm CMOS standard library [10] for 1.00 VDD and 25 °C temperature in which an FO4 latency is 45 ps and the area of a NAND2 is 4.4 μm^2.The critical path delay of the proposed butterfly architecture, equal to that of FDPA, consists of a constant multiplier, a three-operand FP adder plus registers.The total consumed area of the proposed butterfly unit is evaluated as 375 347 μm^2 of which 8337 μm^2 is for registers.The major works on FP butterfly architecture are [1] and [11]. However, Sohn and Swartzlander, Jr., [2] have proposed a very fast FP dot-product unit which can be used in the design of a high-performance butterfly unit. Replacing the dot-product unit of [1] with this faster one, leads to a high-speed butterfly architecture.Table IV shows the comparison of the proposed butterfly archi- tecture with those of the fastest previous works. As a result, the proposed design, simulated in 90 nm (versus 45 nm), is yet much faster than those of previous works. Moreover, scaling the area of the proposed design to 45-nm technology results in the value of about $(\sqrt{375\,347}/2)^2 = 93\,836\ \mu m^2$ which is almost equal to that of [11].

E. MODEL OUTPUT



```
# isim force add {/mux2/a} 0000000000000000000000011 -radix bin
ISim>
# isim force add {/mux2/b} 0000000000000000000000011 -radix bin
ISim>
# isim force add {/mux2/x} 0000000000000000000000111 -radix bin
ISim>
# run 1.00us
ISim>
```

F .CONCLUSION

A high-speed FP butterfly architecture, which is faster than previous works but at the cost of higher area. The reason for this speed improvement is twofold: 1) BSD representation of the significands which eliminates carry-propagation and 2) the new FDPA unit proposed in this brief. This unit combines multiplications and additions required in FP butterfly; thus higher speed is achieved by eliminating extra LZD, normalization, and rounding units. Further research may be envisaged on applying dual-path FP architecture to the three-operand FP adder and using other redundant FP representations. Moreover, use of improved techniques in the termination phase of the design (i.e., redundant LZD, normalization, and rounding) would lead to faster architectures,though highera reacosts are expected.

G .REFERENCES

[1] 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.

[2] J. 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.

[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 calculation of 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 IEEE Symp. 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. Tech- nol.*, 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 mul- tipliers," in *Proc. 45th Asilomar Conf. Signals, Syst. Comput.*, Nov. 2011, pp. 520–524.