e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 04
February 2018

HIGH SPEED IMPLEMENTATION OF NOVEL FFT BASED ON PARALLEL PREFIX TECHNIQUE

[1]REBBA SANTHI SWARUPA RANI, [2]K. RAVI KUMAR.
[1]M.tech-Scholar, Dept of ECE, Malineni Lakshmaiah Women's Engineering College, Guntur, A.P, India
[2]Associate Professor, Dept of ECE, Malineni Lakshmaiah Women's Engineering College, Guntur, A.P, India

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, since many signal processing applications need high throughput more than low latency. The FFT function consists of consecutive multiply add operations over complex numbers, dubbed as butterfly units. Recently, floating-point (FP) arithmetic is applied to FFT architectures, specifically butterfly units has become more popular. It offloads compute-intensive tasks from general-purpose processors by dismissing FP concerns for example scaling and overflow/underflow. However, the major downside of FP butterfly is its slowness in comparison with its fixed-point counterpart. So, parallel prefix (PP) multiplier is proposed which is high speed. The Parallel prefix multiplier provides fast of operation when compared with the FP multiplier. It gives better performance than the existed system.

Key words: Binary-signed digit (BSD) representation, butterfly unit, complex number system, fast Fourier transform (FFT), floating point (FP), redundant number system, three operand addition.

I. INTRODUCTION

Fast Fourier transform (FFT) circuitry contains several consecutive multipliers and adders over complex numbers. Hence an appropriate number representation must be chosen wisely. Most of the FFT architectures have been utilizing fixed-point arithmetic. Recently, FFTs depends on floating-point (FP) operations are growing today. The main advantage of FP over fixed-point arithmetic is the wide dynamic range; but it is 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.

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. Hence delay, area, and power consumption are reduced. Redundant number system is other well-known way of overcoming slowness of FP, 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 $\beta - \alpha + 1 > r$.

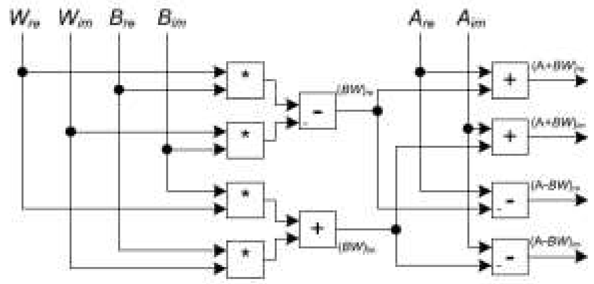

FIG. 1. FFT BUTTERFLY ARCHITECTURE WITH EXPANDED COMPLEX NUMBERS

The transformation from non-redundant, to a redundant format is a carry-free operation, hence, the reverse conversion requires carry propagation. This makes redundant representation more useful where several consecutive arithmetic operations are performed prior to the final result. This proposes a butterfly architecture utilizing redundant FP arithmetic, useful for FP FFT coprocessors and provides to the applications of digital signal processing. While, there are other works on the usage of redundant FP number systems but 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) Design of all the significands are represented in binary signed digit (BSD) format and the corresponding carry-limited adder.

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

3) FP three-operand adders for operands with BSD significands are designed.

4) FP fused-dot-product-add (FDPA) units for operands with BSD significands are designed.

II.EXISTED SYSTEM

A. Existed Redundant Floating-Point Multiplier: The Existed multiplier, contains two major steps, 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 in the same way as 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 due to its representation of the input operands. Moreover, given that *Wre* and *Wim* are constants, the multiplications 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. Fig. 2 shows the required circuitry for the generation of PP *i* where each PP consists of ($n+1$) digits (i.e., binary positions).

2) Partial Product Reduction: In this PPR step major component is the carry-limited addition over the operands which are represented in BSD format. The length of the final product may be more than $2n$.

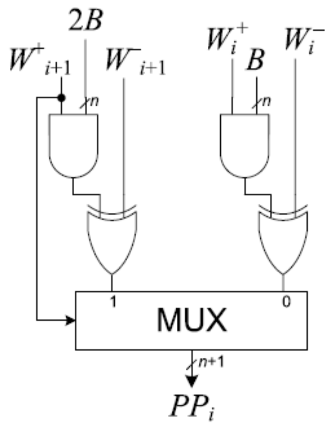

FIG. 2. GENERATION OF THE *I*TH PP

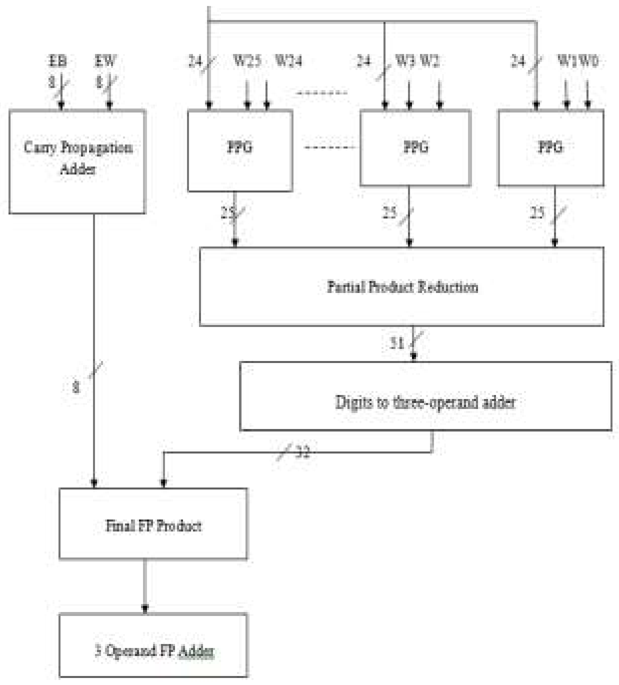

FIG. 3. EXISTED REDUNDANT FP MULTIPLIER

III.PROPOSED SYSTEM

In Parallel Prefix adders the execution of an operation is in parallel. This is done by segmentation the operation in smaller pieces which are computed in parallel. The output is depends on the initial inputs. Parallel Prefix Adder (PPA) is equivalent to carry look ahead adder (CLA). A Carry look ahead adder is a type of adder used in digital logic. CLA is designed to overcome the latency introduced by repelling effect of carry bits in RCA. A CLA improves speed by reducing carry bits. It calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of larger bit value.

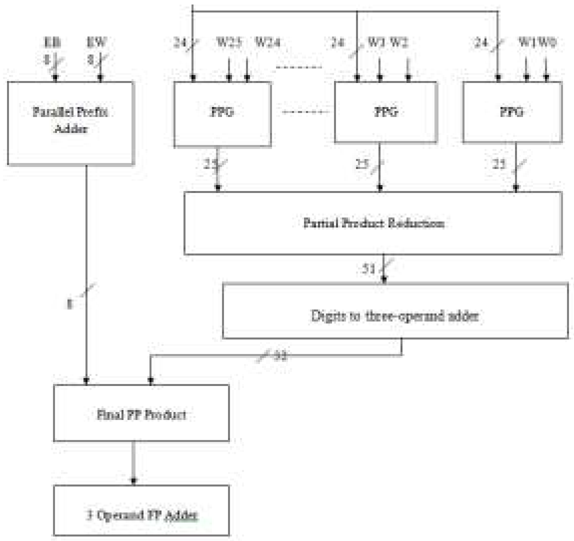

FIG. 3. PROPOSED REDUNDANT PP MULTIPLIER

CLA uses the concept of generating (G) and propagating (P) carries. These two are differ in the way their carry generation block is implemented. The main advantage of PPA is the carry reduces the number of logic levels by essentially generating the carries in parallel. PPA fastest adder with focus on design time and is the choice for high performance adder in industry.

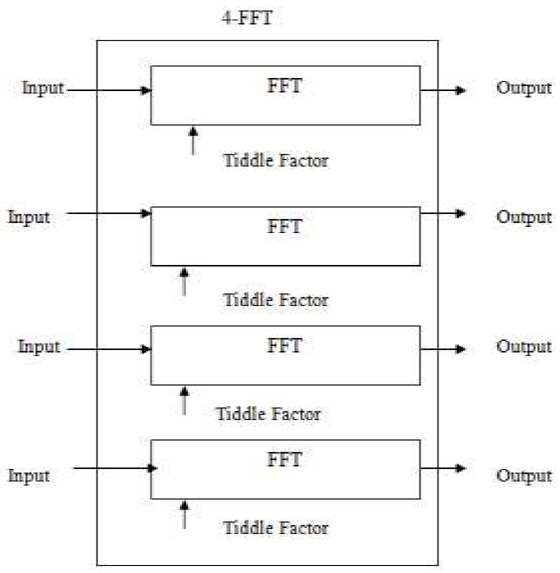

FIG 4. FFT DESIGN WITH PPA & PPM

The proposed work focuses on two new techniques for reducing the hardware overhead and increasing the error correction capability. The technique analyzed in the previous work has certain limitation due to the complexity of handling larger number of FFTs. Partial summation is used for calculating its parity at the input and the output side of the FFT. It sums all possible node values of 4-point FFT along with the twiddle factors.

IV.RESULTS

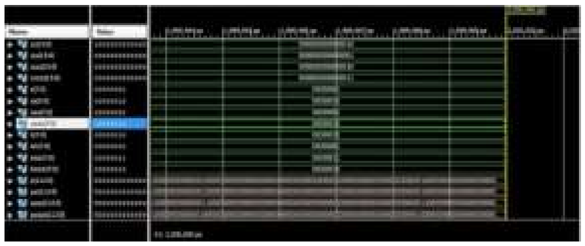

FIG 5. OUTPUT WAVEFORM-1

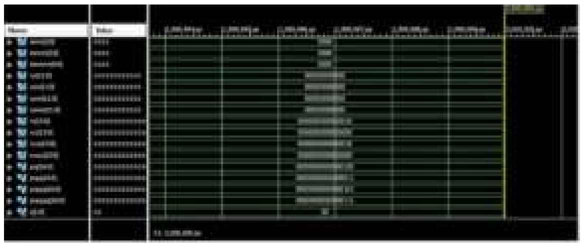
FIG 6. OUTPUT WAVEFORM-2

V.CONCLUSION

A high-speed FP butterfly architecture is presented 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 predicted on applying dual-path FP architecture to the three-operand FP adder and utilizing other redundant FP representations. Hence, utilization of improved techniques in the termination phase of the design would lead to faster architectures, though higher area costs are expected. Parallel prefix (PP) multiplier is utilized in the proposed system which provides high speed and efficient performance.

VI.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. 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 packetforwarding 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. 2011, pp. 520–524.