A Novel Distributed Arithmetic Approach for Computing a Radix-2 FFT Butterfly Implementation

Kevin N. Bowlyn[1], and Nazeih M. Botros[2]
1. Ph.D. Candidate, Dept. of Electrical & Computer Engineering, Southern Illinois Univ., Carbondale, IL, USA
2. Professor Emeritus, Dept. of Electrical & Computer Engineering, Southern Illinois Univ., Carbondale, IL, USA

Abstract – *In this paper, we present a new integration technique for computing a multiplierless fast Fourier transform (FFT) butterfly structure using the Distributed Arithmetic (DA) and Complex Binary Number System (CBNS) approach. With the combine DA-CBNS structure, each complex number is being represented as a single entity instead of two without having to do the divide and conquer approach. The results show a 100% reduction in the use of any dedicated multipliers and 50% reduction in the use of arithmetic operations. The design was implemented at various N-point computations and a preliminary analysis shows that for a 28-bit butterfly structure, its area size and power consumption were found to be reduced by 50% and 59%; for a 32-bit structure, 67% and 58%; and for a 56-bit structure, 40% and 49%, respectively, when compared to the MAC approach.*

Keywords: Distributed Arithmetic (DA), Fast Fourier Transform (FFT), Complex Binary Number System (CBNS), Multiply and Accumulate (MAC), Digital Signal Processing (DSP)

1 INTRODUCTION

In Digital Signal Processing (DSP) and Digital Imaging Processing (DIP), Fourier Series (FS) and Fourier Transform (FT) play a vital role in many signal processing applications. Most electronic devices and electronic medical devices such as computerized tomography (CT) scan, cellular phones, GPS navigation devices, Bluetooth, and radios all rely on signal processing for high speed performance. Due to the increased demand for a faster implementation of the above algorithms for low cost power and area design, many researchers have been working on an efficient method on how to improve the computational performance within the signal processing applications. In this paper, we present a technique that will solve the computational complexities of the FFT algorithm by eliminating the need for any dedicated multipliers and by representing complex numbers as a single entity instead of two as seen in the traditional way of computing the FFT algorithm.

1.1 Fast Fourier Transform

The fast Fourier transform (FFT) algorithm was first introduced by Gauss in 1805 and later developed by Cooley and Tuckey in 1965. Cooley and Tuckey [1] introduced a new and efficient way for computing the discrete Fourier transform algorithm (DFT) by introducing butterfly structures for computing the FFT algorithm in a reduced amount of time. The FFT algorithm eliminates several repeated arithmetic calculation that is found when computing the DFT algorithm. There are two types of FFT algorithms: Decimation in Time (DIT) and Decimation in Frequency (DIF). Due to the uniqueness of the FFT structure, the FFT algorithm is able to compute the DFT algorithm faster with fewer amounts of hardware structure, time, power, and speed. Figure 1 shows the overall structure of an 8-point radix-2 algorithm and Table 1 shows the comparison of savings between the traditional DFT and the radix-2 FFT algorithm.

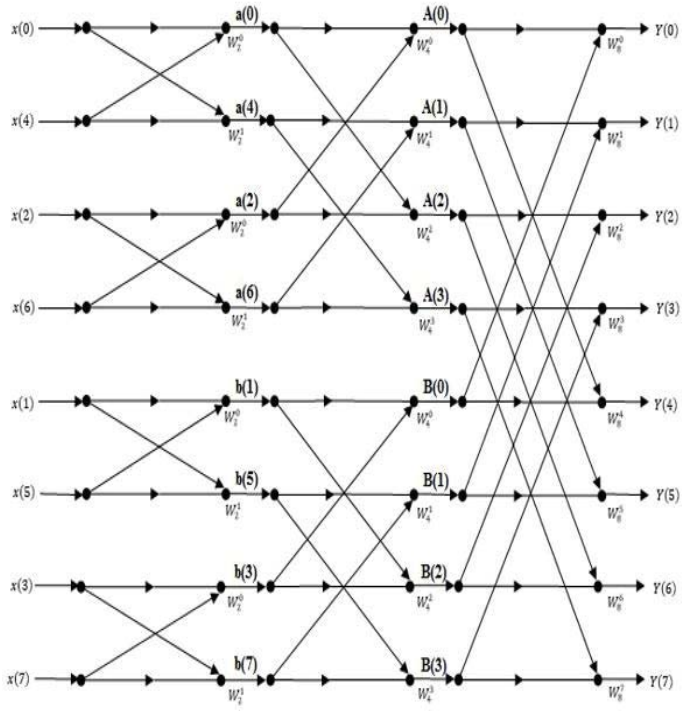

Figure 1. Full 8-point Radix-2 FFT Implementation

Table 1. N-point comparison between DFT vs. Radix-2 FFT computations

Number of Points	Direct computation of DFT		Radix-2 FFT	
	Real Multipliers	Real Adders	Real Multipliers	Real Adders
N	$(N^2)*2$	$(N(N-1))*2$	$(N/2)\log_2 N)*4$	$(N\log_2 N)*3$
4	32	24	16	24
8	128	112	48	72
16	512	480	128	192
64	8192	8064	768	1152
256	131072	130560	4096	6144
1024	2097152	2095104	20480	30720

1.2 Multiply and Accumulate

Traditionally, the sum of products (SOP) is computed by the given expression below:

$$y = \sum_{K=1}^{K} A_K x_K \qquad (1)$$

where A_K is a matrix of constant values and x_K is a matrix of input variable vectors. The multiply and accumulate (MAC) approach commonly computes its inner product from equation (1) by using dedicated multipliers and adders. These dedicated multipliers are fast but they consume a large amount of hardware especially when the size of the multiplier increases linearly. If for example K = 512, the number of dedicated multipliers that will be needed to compute this SOP is 512 multipliers and 511 adders. Therefore, it will take K numbers of multipliers and K – 1 numbers of adders to compute this algorithm.

1.3 Distributed Arithmetic

The Distributed Arithmetic (DA) approach is a technique that has being employed for over four decades and it has recently been utilized in many different DSP applications. It was first developed in 1973 by Croisier et al [2] and in 1974 Peled and Liu [3] later remodified the architectural structure. The DA approach is a bit serial in nature and it computes the inner dot product between two vectors: fixed and varying input by shifting and adding only. No dedicated multipliers are needed to compute this algorithm. The DA system utilizes a ROM-based look up table (LUT) to store all precomputed values in a memory table. Figure 2 shows a 3-tap DA based architecture structure. From figure 2, the input data sets are being delivered in one-bit-at-a-time (1BBAT) into the ROM-LUT. These input values form the address line of the ROM-LUT. Ts is a timing signal that controls the adder/subtractor circuit. Ts = 1 when the most significant bit (MSB) is attained (sign bit time) otherwise Ts is zero. This bit is needed to determine when the final result y[n] is ready. The DA computes its multiplication process by shifting and adding with less hardware structure compared to the MAC approach. This allows for the additional multiplier operation to be eliminated by employing a memory to store linear combinations of coefficients.

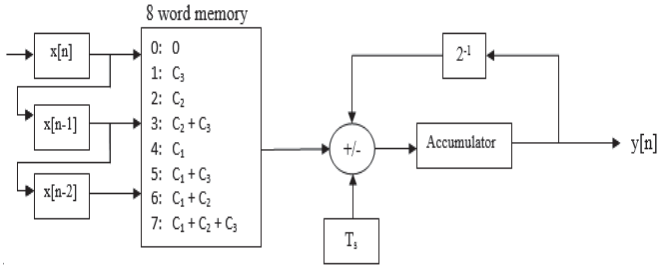

Figure 2. DA-based Implementation of a 3-tap 2^K Filter

The term DA means its arithmetic operations are evenly distributed throughout by adding all of its equal shift together rather than being lumped sum as observed with the MAC approach. A full derivation of the DA approach can be found in [4] [5] [6] [7] [8].

1.4 Complex Binary Number System

In the 21st century, complex numbers play a fundamental role in many different engineering applications such as DIP and DSP. Within today's current technology, complex numbers are being computed by using the divide and conquer approach in which the real and imaginary numbers are being separated. With the complex binary number system (CBNS) technique, which was first introduced by Penny [9, 10] in 1964 and later developed by Jamil [11] in 2000, complex numbers are now being represented as a single entity instead of two without having to do the divide and conquer approach. This is accomplished by representing complex numbers into the (-1+ j) base form. A full conversion of this algorithm can be found in [8] [12] [13] [14] [15]. With this representation, the addition of two complex numbers can now be computed as one single complex addition instead of two separated complex additions and the multiplication of two complex numbers can be obtained by one single complex multiplication and addition instead of four multiplications and two additions/subtractions.

1.5 Paper Structure

The remainder of this paper is organized as follows. Section 2 presents the FFT butterfly algorithm for the DA-CBNS structure. The DA-CBNS butterfly implementation is presented in section 3. Results and work-in-progress are presented in sections 4 and 5, respectively. Finally, conclusion is presented in section 6.

2 FFT BUTTERFLY ALGORITHM DA-CBNS

The FFT algorithm comprises of the use of its butterfly structures which is the essential heart of the algorithm. Figures 3 and 4, shown below, show the overall representation of a 2-point DIT and DIF butterfly structure. Within the DIT algorithm, the multiplication process is done before its complex addition and subtraction while for the DIF algorithm, the multiplication process is done after its complex addition and subtraction. Each FFT algorithm is comprised of N number of butterfly structures per stage. For an 8-point FFT algorithm shown in Figure 1, twelve butterfly structures were needed to compute this algorithm. This resulted in the total numbers of multipliers and adders to compute this algorithm to be 48 and 72, respectively. Within the traditional approach, each butterfly structure consists of four multipliers and six adders/subtractions while with the proposed DA-CBNS 8-point structure, the total number of multipliers and adders were greatly reduced to zero and three, respectively. The additional adder was due to the DA based architecture system. This resulted in a 100% reduction in the use of no dedicated multipliers and 50% reduction in the arithmetic operations. Another factor with this design is that each complex number is treated as a single entity and not two as seen in the traditional MAC approach where the real and imaginary parts are being computed separately.

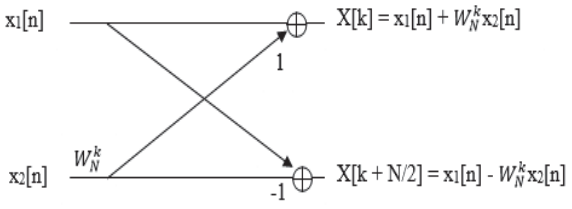

Figure 3. DIT Butterfly Structure

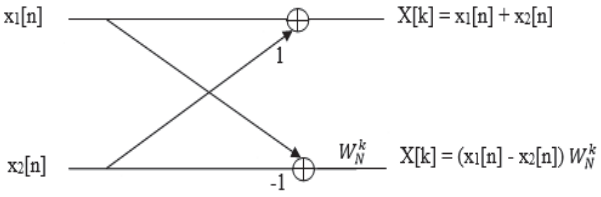

Figure 4. DIF Butterfly Structure

3 DA-CBNS BUTTERFLY IMPLEMENTATION

For a DIT and DIF butterfly structure, the number of arithmetic computations that is needed to compute each butterfly structure in calculating the radix-2 FFT algorithm will consist of four multiplications and six additions. In integrating the DA-CBNS technique within the FFT algorithm, the arithmetic computations for each butterfly structure will only consist of one complex multiplication and addition, and two complex additions/subtractions. Figure 5, shows the block diagram for the DA-CBNS implementation. The input values of the butterfly structure was first converted from its complex decimal base-10 number to its (-1 + j) CBNS complex binary base representation bit value. The multiplication process is accomplished by using the DA technique in which no dedicated multipliers were needed in computing this algorithm. Within the DA system, the twiddle factor 'wn' was loaded into a ROM-based LUT-less structure in which the DA computes the multiplication process by shifting and adding only. The final output result of the butterfly structure was outputted in its (-1 + j) base representation. This final result was then converted to its complex decimal base-10 number for verification.

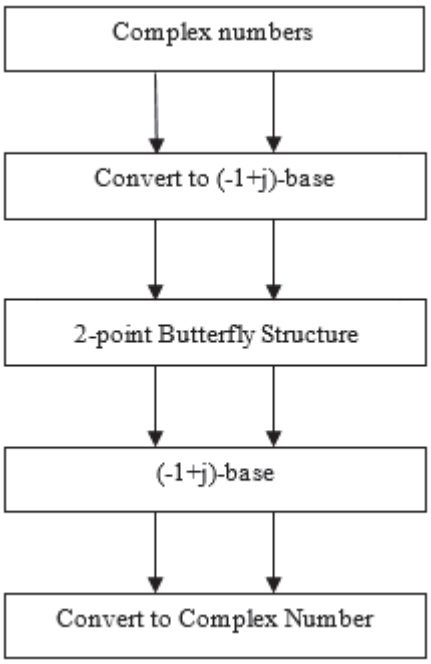

Figure 5. DA-CBNS 2-point FFT Butterfly Implementation

4 RESULTS

In comparing the power and area consumption between the traditional butterfly structure compared with the proposed DA-CBNS butterfly structure, the DA-CBNS design utilized less LUT resources and by far had the lowest amount in terms of its area size and power consumption. Table 2 shows the comparison of these results. For a 28-bit DA-CBNS butterfly structure in which the traditional approach was broken up into two 14-bits, to represent its real and imaginary parts, its area size and power consumption was reduced to 50% and 59%, respectively. For the 32-bits, its area and power consumption was reduced to 67% and 58% and for the 56-bits structure 40% and 49%, respectively. In terms of speed and

simulation run-time, the DA-CBNS for each of the butterfly structures: 28-bits, 32-bits, and 56-bits were significantly faster than the traditional approach that used dedicated multipliers. In regards to the simulation run-time, the DA-CBNS algorithm was able to simulate the design faster than its counterpart. Both of these designs were implemented by coding in VHDL using Xilinx ISE design suite software program version 14.2 on a Virtex 5 FPGA chip.

Table 2. DA-CBNS vs. Traditional MAC Comparison Results

Butterfly Structure	Area DA-CBNS (%)	Area Traditional (%)	Power DA-CBNS (mW)	Power Traditional (mW)
28 bits	1	2	22.40	54.19
32 bits	1	3	29.55	70
56 bits	3	5	67.48	131.53
Butterfly Structure	LUT's DA-CBNS	LUT's Traditional	Run-Time DA-CBNS (mins)	Run-Time Traditional (mins)
28 bits	1274	3286	9	25
32 bits	1705	4265	11	27
56 bits	3975	8033	17	37
Butterfly Structure	Speed DA-CBNS (Mhz)	Speed Traditional (Mhz)	Area Reduction (%)	Power Reduction (%)
28 bits	99.87	60.23	50	59
32 bits	87.42	47.19	67	58
56 bits	34.80	21.25	40	49

5 FUTURE WORK

These techniques will be tested on a larger scale for computing the FFT structure and will be further analyzed to see which technique will give the best overall performance in terms of cost, power, area size, speed, and time. This proposed DA-CBNS method will overall decrease the cost for implementing this structure compared to traditional approach for computing the algorithm. Given that there is a great demand for new and improved technology for DIP and DSP applications, in calculating the SOP, the method discussed in this paper will be used in implementing the new radix-2 FFT structures employing DA and CBNS at different N points.

6 CONCLUSION

The DA-CBNS technique illustrates the vital importance of having complex numbers being represented as a single entity instead of two. By integrating these algorithms, this resulted in 100% reduction in the use of no dedicated multipliers and 50% reduction in the use of arithmetic operations when compared to the traditional approach for computing the FFT algorithm. An efficient algorithm is needed for improving the computational performance and reducing its computational arithmetic operation complexities of the FFT algorithm due to the fast and rapid development with the DIP and DSP fields. This technique will result in less hardware structure and it will also result in a decrease in its area size, power consumption, and cost, as multipliers occupy a large volume of hardware and they are fairly costly in implementing. As stated before, in calculating the radix-2 FFT algorithm for both DIT and DIF, the number of arithmetic computations for each butterfly structure will consist of four multiplications and six additions/subtractions. With the DA-CBNS technique, the arithmetic computations for each butterfly structure will consist of only one complex multiplication and addition and two complex additions/subtractions.

7 REFERENCES

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation,* vol. 19, pp. 297-301, 1965.

[2] A. Croisier, D. J. Esteban, M. E. Levilion and V. Rizo, "Digital filters for PCM encoded signals". U.S. Patent 3.777.130, April 1973.

[3] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Transactions on Acoustic, Speech, and Signal Processing,* Vols. ASSP-22, no. 6, pp. 456-462, 1974.

[4] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutor review," *IEEE ASSP Magazine,* pp. 4-19, 1989.

[5] W. Huang, "Implementation of adaptive digital FIR and reprogrammable mixed-signal filters using distributed arithmetic," PhD Thesis, Dept. Elect. & Comput. Eng., Georgia Institute of Tech., Atlanta, 2009.

[6] R. Guo and L. S. DeBrunner, "A novel adaptive filter implementation scheme using distributed arithmetic," *Signals, System and Computers,* pp. 160-164, 2011.

[7] H. Yoo and D. V. Anderson, "Hardware-efficient distributed arithmetic architecture for high-order digital filters," *IEEE International Conference on Acoustics, Speech and Signal Processing,* vol. 5, pp. 125-128, 2005.

[8] K. N. Bowlyn and N. M. Botros, "A novel distributed arithmetic multiplierless approach for computing complex inner products," *2015 International Conference on Parallel and Distributed Processing Techniques and Applications,* pp. 606-612, 2015.

[9] W. Penney, "A numerical system with a negative base," *Mathematical Student Journal,* pp. 1-2, 1964.

[10] W. Penney, "A binary system for complex numbers," *Journal of the ACM,* vol. 12, no. 2, pp. 247-248, 1965.

[11] T. Jamil, N. Holmes and D. Blest, "Towards implemention of a binary number system for complex numbers," *Proceedings of the IEEE SoutheastCon 2000,* pp. 268-274, 2000.

[12] T. Jamil, "The complex binary number system," *IEEE*

Potentials, vol. 20, no. 5, pp. 39-41, 2002.

[13] T. Jamil and U. Ali, "Effects of multiple-bit shift-right operations on complex binary numbers," *Proceedings of IEEE SoutheastCon 2007,* pp. 759-764, 2007.

[14] T. Jamil, "Impact of shift operations on (-1+j)-base complex binary numbers," *Journal of Computers,* vol. 3, no. 2, pp. 63-71, 2008.

[15] T. Jamil, "An introduction to complex binary number system," *Fourth International Conference on Information and Computing,* pp. 229-232, 2011.