RES MILITARIS
REVUE EUROPEENNE D ETUDES EUROPEAN JOURNAL OF MILITARY STUDIES
Social Science Journal

Efficient Realization of Fast Fourier Transform based on Distributed Arithmetic

By

M.Bharathi
Research Scholar,VTU & Assistant Professor, Department of ECE, School of Engineering and Technology, Mohan Babu University,Tirupati,Chittoor District, AndhraPradesh,India
Email Id: bharathi.m@vidyanikethan.edu, bharathi891@gmail.com

Dr. Yasha Jyothi M Shirur
Professor, Department of ECE, BNM Institute of Technology, Bangalore, VTU
Email: yashamallik@gmail.com, yashajyothimshirur@bnmit.in

Abstract

Global positioning system (GPS) plays a major role in real time communications and our spotlight is to calculate the communication delay in GPS system. The backend operation of GPS navigation system is convolution. GPS devices convolve two signals, one from satellite and other is shifted version of time. Fast Fourier Transform (FFT) is recognized as one of the most efficient building-block in performing convolution between two real time signals. Some of the applications includes Signal Processing, Data Analysis, Medical Imaging and digital formats like JPEG, MP3, H.264. Since Distributed Arithmetic (DA) takes the values in bit-serial fashion it suits for real time signal processing applications. DA is a method that can be employed to perform inner dot product between two signals: Fixed and varying signal by using shifters and adders. This project reveals a new integration approach for computing using DA. The proposed design will be implemented for 8-bit butterfly structure by incorporating DA technique. Finally, the results of proposed butterfly structure will be composed with existing butterfly structure for equal length. The performance evaluation of LUT and LUT-less based FFT butterfly structure designs can be evaluated using Xlinix ISE.

Keywords: Distributed Arithmetic (DA), Fast Fourier Transform, LUT, LUT-less (adder based)

Introduction

Digital signal processing (DSP) is crucial in telecommunications, especially in wireless cellular networks. It is used for digital filtering, signal estimation, and data encoding and decoding. The most crucial factor is the level of processing complexity needed to handle multiple access interference. A specialised microprocessor known as a digital signal processor (DSP) is made to fulfil the high-speed operational needs of digital signal processing. Here, we'll take a quick look at the DSP technology.

To create a DSP system for a particular application, either a dedicated processor-based DSP or a general-purpose processor-based DSP might be employed.

(i) DSP based on Dedicated Processors: Array operations and multiply-accumulate, filtering[5] operations are the sole focus of DSP processors. DSP-based systems are self-contained, transportable, inexpensive, and well-suited for real-time applications[10],[11].

(ii) General-purpose microprocessors or computers are used in these systems, which are based on DSP. The software was created to make it possible for computers to do DSP operations. Computer programmes can be made for digital filtering[9], z-transform, Fourier transform, and other operations. As a result, the utility of computers can be increased. These systems can be upgraded and adjusted. It is possible to share computer resources including networking, storage, displays, and printing. However, the computational efficiency of such systems is low. If just DSP functions need to be performed, specialised processor-based solutions are ideal. Discrete time signals are used in DSP. Both the time and frequency domains can be used to analyse discrete time signals. The two most used tools for signal analysis are the Fourier transform and discrete Fourier transform.

DSP applications can be implemented on discrete time systems. Included are FFT algorithms, estimates, the design and implementation of digital filters, and other computational DSP techniques. Any DSP application needs to know this fundamental information.

Fast Fourier Transform (FFT)

The DFT can be carried out using the FFT (Fast Fourier Transform) algorithm with less calculations. In order to boost computing efficiency, a divide and conquer approach is employed. An N-point DFT is divided up into smaller and smaller DFTs as part of this procedure. Based on this fundamental idea, FFT algorithms are a group of effective computer approaches. In general, there are just two FFT algorithms. Two well-known FFT algorithms are the Decimation in Time (DIT) and Decimation in Frequency (DIF) methods.

When the sequence is N = 2m in length, the radix is 2, and there are m phases in the computation. In radix-2 FFT, the N-point sequence is divided into two N/2-point sequences, each of which is then split into two N/4-point sequences, and so on, until we end up with 2-point sequences. After computing DFTs for 2-point sequences, two 2-point sequences are combined into DFTs for a 4-point sequence, two 4-point sequences into DFTs for an 8-point sequence, and so on until the N-point DFT is obtained.

To directly compute DFT for a sequence of length N, N2 complex multiplications are needed. A maximum of N/2 log 2N multiplications can be achieved using the FFT technique.

Existing Fft

When N is expected to be a power of two, $N = 2^m$, it changes to radix-2 FFT. For instance, the sequence a(n) above can be divided into two N/2 sequences when $N = 4 = 2^2$, with the first sequence containing samples a(0) and a(2) and the second sequence containing examples a(1) and a(3). The DFT for N = 4 can be evaluated using radix-2 DIT and DIF methods, starting with the fundamentals.

When utilising radix-2 FFT, the 8-point DFT is computed in three steps. Consequently, for N = 8 = 23, r = 2 and m = 3. Four 2-point sequences make to the 8-point sequence. For each 2-point sequence, the 2-point DFT is calculated. Four 2-point DFTs are used to create two 4-point DFTs, while two 4-point DFTs are used to create an 8-point DFT.

Let the eight-sample sequence a be defined as a(0), a(1), a(2), a(3), a(4), a(5), a(6), and a(7) of a (n). Sequences of two samples should be created from the eight samples.

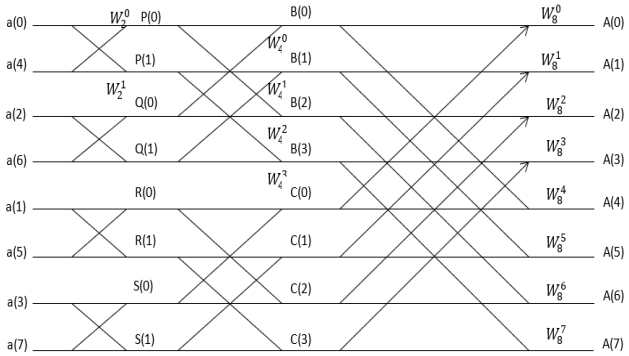

Figure 1: *Illustration of complete flow graph obtained by combining all the three stages for N=8*

Normal order		Bit reversed order	
a(0)	a(000)	a(0)	a(000)
a(1)	a(001)	a(4)	a(100)
a(2)	a(010)	a(2)	a(010)
a(3)	a(011)	a(6)	a(110)
a(4)	a(100)	a(1)	a(001)
a(5)	a(101)	a(5)	a(101)
a(6)	a(110)	a(3)	a(011)
a(7)	a(111)	a(7)	a(111)

As can be seen below, a(n) is decimated into 4 bits-reversed 2-point sequences of numbers.

(i) a(0) and a(4)
(ii) a(2) and a(6)
(iii) a(1) and a(5)
(iv) a(3) and a(7)

The 8-point sequence is a(n) = a(0), a(1), a(2), a(3), a(4), a(5), a(6), a(7).
4-point sequences b(n) = a(0), a(2), a(4), a(6); c(n) = a(1), a(3), a(5), a(7) derived from a(n).
2-point sequences produced from b(n): p(n) = a(0), a(4); q(n) = a(2), a(6)
2-point sequences produced from c(n), r(n) = a(1), a(5); s(n) = a(3), a(7)
Below are the relationships between the samples of distinct sequences.

p(0) = b(0) = a(0) r(0) = c(0) = a(1)
p(1) = b(2) = a(4) r(1) = c(2) = a(5)
q(0) = b(1) = a(2) s(0) = c(1) = a(3)
q(1) = b(3) = a(6) s(1) = c(3) = a(7)

Let's first talk about the problems with the DIT and DIF FFT algorithms that have been found, and how the DA methodology is used in the suggested way to fix them.

1. When multipliers are utilized, computations take a long time and take up extra space in the hardware design.
2. The speed is decreased because the input to output latency is longer.
3. The performance of the DSP processor is hampered by excessive arithmetic operations (multipliers).

Distributed Arithmetic:

A method of implementation without multipliers is distributed arithmetic (DA).

DA [1]is a technique that conducts the inner dot product between two signals—fixed and variable signals—using shifters and adders.

$$Y = \sum_{k=1}^{K} A_k x_k$$

DA stores values in a bit-serial format that is ideal for real-time processing[4].

Proposed Design Using Distributed Arithmetic

Distributed Arithmetic (DA) with a numerical example

$$Y = \sum_{k=1}^{K} A_k x_k \quad has\ only\ 2^k\ possible\ values$$

Distributed Arithmetic (DA) with a numerical example

Address =a3, a2, a1, a0 = 2,3,4,5;
Inputs = b3, b2, b1, b0 = 1,2,4,5;
Step 1: 0*1+1*2+0*4+1*5=7; Y=7
Step 2: 0*1+1*2+0*4+0*5=2; Y=7+2=9
Step 3: 0*1+0*2+1*4+1*5=9; Y=9+9=18
Step 4: 0*1+0*2+1*4+0*5=4; Y=18+4=22

Mathematical Calculations of 2-Point FFT Using Radix-2 Based on DA:

Consider the general DFT equation

$$A(k) = \sum_{n=0}^{N-1} a(n) W_N^{kn} \; ; \; 0 \le k \le N\text{-}1$$

Let us design the 2-point DFT sequence based on Distributed Arithmetic[2].

$$A(k) = \sum_{n=0}^{1} a(n) W_2^{kn} \; ; \; 0 \le k \le 1 \text{ (since N=2)}$$

Expand the equation by substituting the values n = 0 and 1

$$A(k) = a(0) W_2^{k(0)} + a(1) W_2^{k(1)}$$

Later after substituting the entire equation is in terms of variable ‘

The formulated equation consists of inputs a(0), a(1) are constants and the twiddle factor

$$A(k) = a(0) + a(1) W_2^k$$

based on variable k value

$$A(k) = a(0) + a(1) W_4^k + a(2) W_4^{2k} + a(3) W_4^{3k}$$

$$A(0) = a(0) + a(1) W_2^0$$
$$A(1) = a(0) + a(1) W_2^1$$

We know the phase factor is symmetric

$$W_2^1 = -W_2^0 \qquad .: (\ W_2^0 = 1\ )$$

$$A(0) = a(0) + a(1)W_2^0$$
$$A(1) = a(0) - a(1)W_2^0$$

Now, the next step is to incorporate the proposed DA technique to the existing FFT algorithm[3].

The inner dot product of two signals, one fixed and the other changing, is performed using the DA method. Let the fixed signal coefficients be the constant inputs and the twiddle factor of changeable variable k be the constant inputs. The values are taken in a bit-serial form by DA.

As a result, the fluctuating signal is represented in bit-serial form, with constant coefficients as inputs. The twiddle factor which is varying is represented in 2-bit binary representation

$$W_2^0 = W1, W2\ bits$$

Architectural Design of 2-Point FFT based on DA

The output of our proposed design is based on value we get from the bits combination

The output A(0) is 0 when the value of bits is 0
A(0) = 0*a(0) + 0*a(1)=0

The output A(0) is 0 when the value of bits is W_2^0a(1), when the value of bits is 1
A(0) = 0*a(0) + 1*a(1)= W_2^0a(1)

The output A(0) is a(0), when the value of bits is 2.
A(0) = 1*a(0) + 0*a(1)=a(0)

The output A(0) is a(0) + W_2^0a(1) , when the value of bits is 3 A(0) = 1*a(0) + 1*a(1)= a(0) + W_2^0a(1)

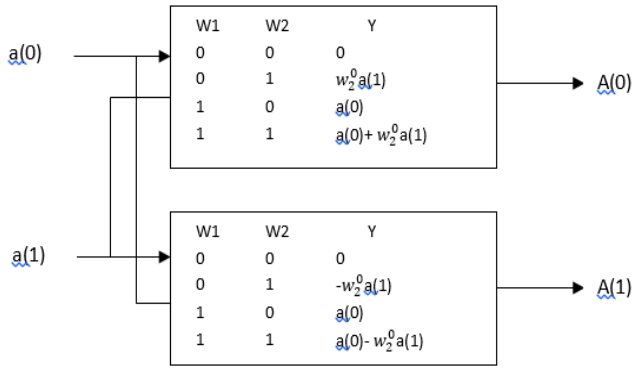

Figure 2 *Design of 2-point FFT using radix-2 based on DA*

Mathematical Calculations 4-Point FFt Using Radix-2 Based On DA

Consider the general DFT equation
$A(k) = \sum_{n=0}^{N-1} a(n)W_N^{kn}$; $0 \le k \le N-1$

Let us design the 2-point DFT sequence based on Distributed Arithmetic
$A(k) = \sum_{n=0}^{3} a(n)W_4^{kn}$; $0 \le k \le 3$

Expand the equation by substituting the values n = 0 ,1,2 & 3

$$A(k) = a(0)W_4^{k(0)} + a(1)W_4^{k(1)} + a(2)W_4^{k(2)} + a(3)W_4^{k(3)}$$

Later after substituting the entire equation is in terms of variable 'k'

The formulated equation consists of inputs a(0), a(1), a(2), a(3) are constants and the twiddle factor based on variable k value

$$A(0) = a(0) + a(1) + a(2) + a(3)$$
$$A(1) = a(0) + a(1)W_4^1 + a(2)W_4^2 + a(3)W_4^3$$
$$A(2) = a(0) + a(1)W_4^2 + a(2) + a(3)W_4^2$$
$$A(3) = a(0) + a(1)W_4^3 + a(2)W_4^2 + a(3)W_4^1$$

We know the phase factor is symmetric

$$W_4^2 = -W_4^0 ;\ W_4^0 = 1, W_4^{-1} = -j, W_4^2 = -1, W_4^3 = j)$$
$$W_4^3 = -W_4^1$$
$$A(k) = \sum_{n=0}^{3} a(n)W_4^{kn} ;\ 0 \le k \le 3$$
$$A(0) = B(0) + W_4^0\, C(0)$$
$$A(1) = B(1) + W_4^1\, C(1)$$
$$A(2) = B(0) + W_4^2\, C(0) = B(0) - W_4^0 C(0)$$
$$A(3) = B(1) + W_4^3\, C(1) = B(1) - W_4^1\, C(1)$$
$$A(1) = 1*B(1) + 1*C(1) = B(1) - W_4^1 C(1)$$

***Architectural Design of 4-Point FFT based on* DA**

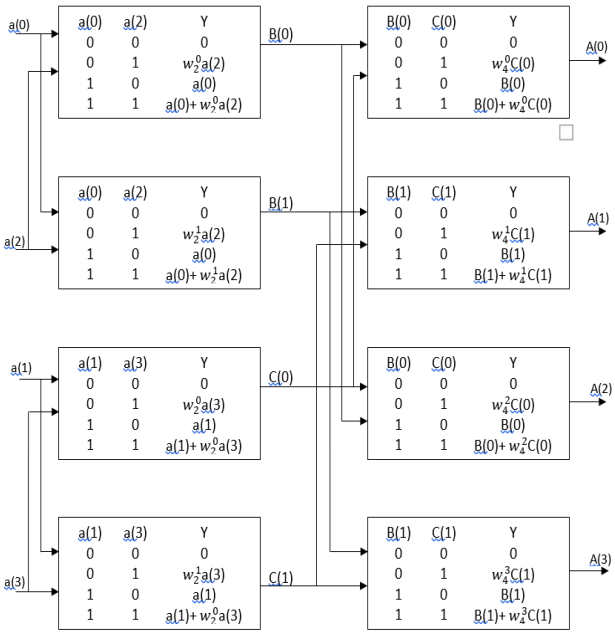

Figure 3 ***Design of 4-point FFT using radix-2 based on DA***

Now, the next step is to incorporate the proposed DA technique to the existing FFT algorithm. The inner dot product of two signals, one fixed and the other changing, is performed using the DA method[6]. Let the fixed signal coefficients be the constant inputs and the twiddle factor of changeable variable k be the constant inputs. The values are taken in a bit-serial form by DA. As a result, the fluctuating signal is represented in bit-serial form, with

constant coefficients as inputs. The twiddle factor which is varying is represented in 2-bit binary representation

$$W_2^0 = W1, W2\ bits$$

As the digital signal can't be represented with negative symbol the minus sign is directly incorporated in the design.

The imaginary value is not possible in digital format so we considered it to represent as "1".

The output of our proposed design is based on value we got from the bits combination[7,8]

The output A(1) is 0 when the value of bits is 0
A(1) = 0*B(1) + 0*C(1)=0

The output A(1) is 0 when the value of bits is $-W_4^1$C(1), when the value of bits is 1
A(1) = 0*B(1) + 1*C(1) = $-W_4^1$C(1)

The output A(1) is B(1), when the value of bits is 2.
A(1) = 1*B(1) + 0*C(1)=B(1)

The output A(1) is $B(1) - W_4^1$C(1) , when the value of bits is 3.

Results &Discussion

The below are the RTL view and simulation results of 2-point FFT based DA

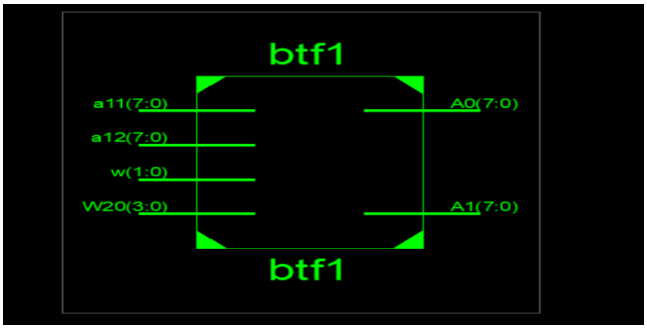

Figure 4: *RTL View of 2-point FFT based on DA*

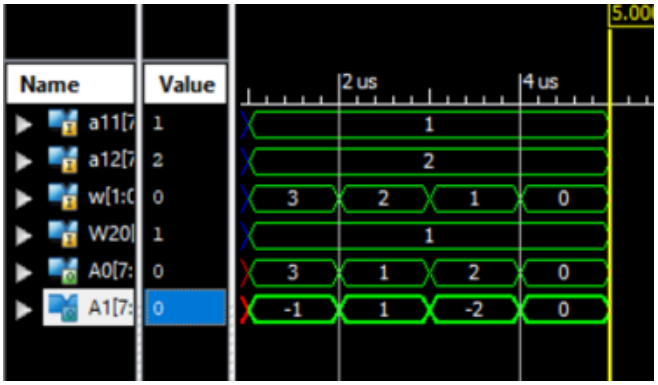

Figure 5: *Simulation of 2-point FFT based on DA*

Calculations:
Inputs: a0, a1 =1,2

Select lines = w [1:0] =3
Outputs: A0 = a0 + w_2^0a1 = 1 +(1) *(2) = 3
A1 = a0 - w_2^0a1 = 1 – (1) *(2) = -1

Select lines=w= [1:0] = 2
Outputs: A0 = a0 = 1
A1 = a0 = 1

Select lines= w [1:0] = 1
Outputs: A0 = w_2^0a1 = (1) *(2) = 2
A1 = -w_2^0a1 = (-1) *(2) = -2

Select lines= w [1:0] = 0
Outputs: A0 = 0
A1 = 0

The below are the RTL view and simulation results of 4-point FFT based DA

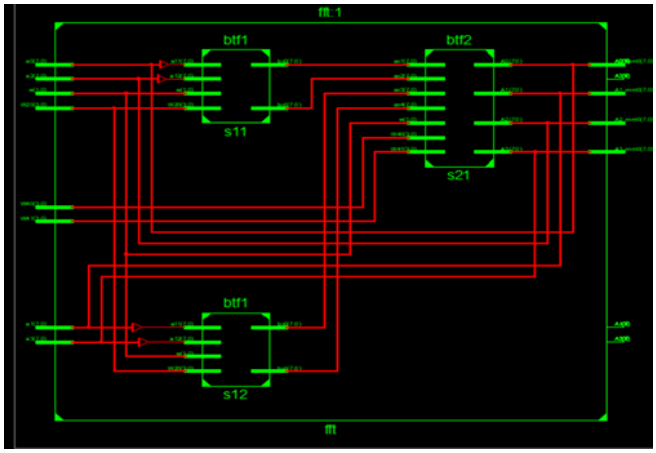

Figure 6: *RTL View of 4-point FFT based on DA*

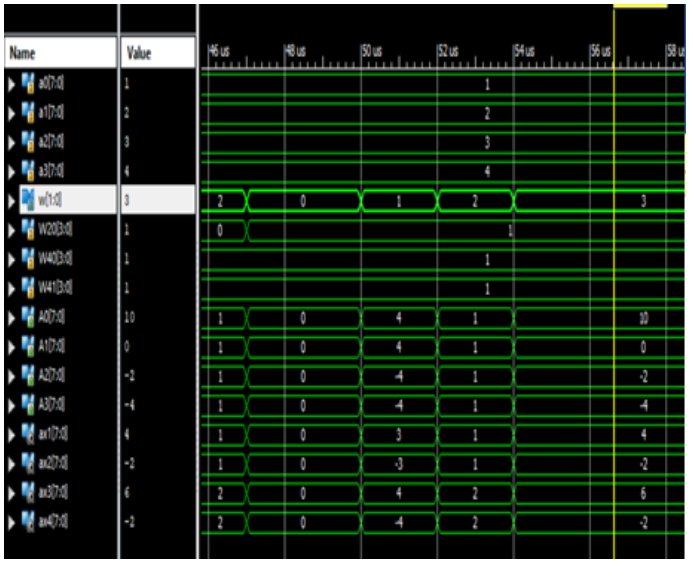

Figure 7: *Simulation of 4-point FFT based on DA*

Calculations:
Inputs: a0, a1, a2, a3=1,2,3,4

Select lines= w [1:0] = 3

a0	1	4	10	--A0
a1	2	-2	0	--A1
a2	3	6	-2	--A2
a3	4	-2	-4	--A3

Outputs: A0, A1, A2, A3 = 10,0, -2, -4

Select lines= w [1:0] = 2

a0	1	1	1	--A0
a1	2	1	1	--A1
a2	3	2	1	--A2
a3	4	2	1	--A3

Outputs: A0, A1, A2, A3 = 1,1,1,1

Select lines= w [1:0] = 1

a0	1	3	4	--A0
a1	2	-3	4	--A1
a2	3	4	-4	--A2
a3	4	-4	-4	--A3

Outputs: A0, A1, A2, A3 = 4,4, -4, -4

Select lines= w [1:0] = 0
Outputs: A0, A1, A2, A3 = 0,0,0,0

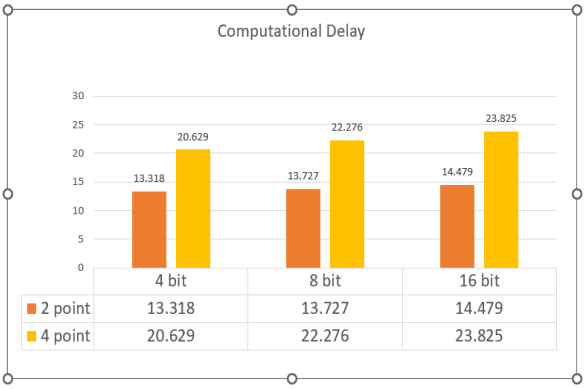

Figure 8: *Comparison of 2 point and 4 point FFT considering various bits*

Conclusion

The FFT algorithm is the most effective method for calculating the DFT of a sequence. The computation time is greatly lengthened by the usage of multipliers. The outcome is a slower speed and a longer output delay. The performance of DSP processors is constrained by the high arithmetic operators. We shall therefore employ a DA-based strategy with a multiplier-less implementation. For multiplication in this case, pre-calculated lookup tables are used. The information is kept in bit-serial format..

According to Figure 8, the suggested 2,4-point FFT's computational latency takes into account a variety of bits, including a 4,8,16-bit representation. The computation takes 20.629 ns for a 4-bit representation and 13.318 ns for a 4-point FFT. The computational delay for a 2-point FFT in an 8-bit format is 13.727 ns, while for a 4-point FFT it is 22.276 ns. The computational delay for 2-point FFT for 16-bit representation is 14.479ns, and for 4-point FFT it is 23.825ns.

Here, projected and actual FFT logic delays are being compared. The existing FFT has a logic delay of 82.6 percent, compared to 79.4 percent for the proposed 2-point 4-bit FFT. The proposed FFT is therefore quicker than the current FFT. Similarily, the proposed logic delay for the 4-point, 4-bit FFT is 77.4%, whereas the actual logic delay is 83.5%, which is 7.3% higher. The proposed FFT's speed in relation to existing FFTs can be calculated with the use of this logic delay factor.

References

D. Xue, V. DeBrunner and L. S. DeBrunner, "Hardware Implementation of Discrete-Hirschman Transform Convolution Using Distributed Arithmetic," 2019 53rd Asilomar Conference on Signals, Systems, and Computers, 2019, pp. 1587-1590, doi: 10.1109/IEEECONF44664.2019.9048809.

K. Bowlyn and S. Hounsinou, "An Improved Distributed Multiplier-Less Approach for Radix-2 FFT," in IEEE Letters of the Computer Society, vol. 3, no. 2, pp. 54-57, 1 July-Dec. 2020, doi: 10.1109/LOCS.2020.3014354.

Z. Li, H. Jia, Y. Zhang, T. Chen, L. Yuan and R. Vuduc, "Automatic Generation of High-Performance FFT Kernels on Arm and X86 CPUs," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 8, pp. 1925-1941, 1 Aug. 2020, doi: 10.1109/TPDS.2020.2977629.

Pramod Kumar Meher; Thanos Stouraitis, "DA-Based Circuits for Inner-Product Computation," in Arithmetic Circuits for DSP Applications, IEEE, 2017, pp.77-112, doi: 10.1002/9781119206804.ch3.

S. Ahmad, S. G. Khawaja, N. Amjad and M. Usman, "A Novel Multiplier-Less LMS Adaptive Filter Design Based on Offset Binary Coded Distributed Arithmetic," in IEEE Access, vol. 9, pp. 78138-78152, 2021, doi: 10.1109/ACCESS.2021.3083282.

B. K. Mohanty and P. K. Meher, "An Efficient Parallel DA-Based Fixed-Width Design for Approximate Inner-Product Computation," in IEEE Transactions on VeryLargeScale Integration (VLSI) Systems, vol. 28, no. 5, pp. 1221-1229, May 2020, doi: 10.1109/TVLSI.2020.2972772.

D. Ray, N. V. George and P. K. Meher, "An Analytical Framework and Approximation Strategy for Efficient Implementation of Distributed Arithmetic-Based Inner-Product Architectures," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 1, pp. 212-224, Jan. 2020, doi: 10.1109/TCSI.2019.2948791.

T. Xu, A. Fumagalli and R. Hui, "Efficient Real-Time Digital Subcarrier Cross-Connect (DSXC) Based on Distributed Arithmetic DSP Algorithm," in Journal of Lightwave Technology, vol. 38, no. 13, pp. 3495-3505, 1 July1, 2020, doi: 10.1109/JLT.2019.2937787.

M Balaji, and N. Padmaja, “High-Speed Pipelined Architecture-Based Residue Number System for FIR Filter Design” Gongcheng Kexue Yu Jishu/Advanced Engineering Science, Vol. 54, Issue. 6, pp. 2056-2066, August, 2022.

Morasa, Balaji, and Padmaja Nimmagadda. 2022. "Low Power Residue Number System Using Lookup Table Decomposition and Finite State Machine Based Post Computation." Indonesian Journal of Electrical Engineering and Computer Science 6(1): 127–34.

Yasmine Begum, M. Balaji, V. Satyanarayana, Quantum dot cellular automata using a one-bit comparator for QCA gates, Materials Today: Proceedings, Volume 66, Part 8, 2022, Pages 3539-3546, https://doi.org/10.1016/j.matpr.2022.06.416.