DOI: 10.21917/ijct.2012.0070

A RADIX-4/8/SPLIT RADIX FFT WITH REDUCED ARITHMETIC COMPLEXITY ALGORITHM

Shaik Qadeer[1], Mohammed Zafar Ali Khan[2] and Syed Abdul Sattar[3]

[1]*Department of Electrical & Electronics Engineering, Muffakham Jah College of Engineering and Technology, India*
E-mail: haqbei@gmail.com

[2]*Department of Electrical Engineering, Indian Institute of Technology Hyderabad, India*
E-mail: zafar@iith.ac.in

[3]*Department of Electrical and Electronics Engineering, Royal Institute of Technology and Science, India*
E-mail: syedabdulsattar1965@gmail.com

Abstract

In this paper we present alternate form of Radix-4/8 and split radix FFT's based on DIF (decimation in frequency) version and discuss their implementation issues that further reduces the arithmetic complexity of power-of-two discrete Fourier Transform. This is achieved with circular shift operation on a subset of the output samples resulting from the decomposition in these FFT algorithms and a proposed dynamic scaling. These modifications not only provide saving in the calculation of twiddle factor, but also reduce the total flop count to ≈$4N\log_2 N$ almost 6% fewer than the standard Radix-4 FFT algorithm $\approx 3\frac{11}{12} N \log_2 N$ ***, 5% fewer than the standard Radix-8 FFT, and*** $\approx 3\frac{7}{9} N \log_2 N$ ***, 5.5% fewer than the standard split radix FFT.***

Keywords:

DFT (Discrete Fourier Transform), FFT (Fast Fourier Transform), Radix-4(R4), Radix-8(R8) and Split Radix (SR) FFT and Flop Count

1. INTRODUCTION

The Fast Fourier Transform (FFT) has become almost ubiquitous and most important in high speed signal processing, it computes the discrete Fourier transform (DFT) of size N [1]. Using this transform, signals can be moved to the frequency domain where filtering and correlation can be performed with fewer operations. It has been widely applied in the analysis and implementation of digital communication systems and television terrestrial broadcasting systems, such as the xDSL (de)modulator, phase correlation system, mobile receiver, as well as fault characterization and classification [6].

When considering the alternate implementations, the FFT/IFFT algorithm should be chosen to consider the execution speed, hardware complexity, and flexibility and precision. Most of this mentioned parameters depend on the exact count of arithmetic operations (real additions and multiplications), herein called flops(floating-point operation), required for a DFT of a given size N which remains an intriguing unsolved mathematical question.

Table.1. Flop Counts (Real Additions + Multiplications) of Standard Radix-4 and our Radix-4/8/Split Radix FFT

SIZE(N)	NEW R4	NEW R8	NEW SR	STD. R4
64	1088	1055	1074	1184
128	2677	-	2624	2901
256	6367	-	6207	6880
512	14773	14386	14340	15926
1024	33631	-	32540	36191
2048	75444	-	72808	81077
4096	167262	163135	161079	179550
8192	367282	-	352345	393906
16384	800090	-	768057	857434

The standard Radix-4 FFT algorithm [1] flop count is:

$$T(N) = 4\frac{1}{4} N \log_2 N - \frac{43}{6} N + \frac{32}{3} \quad (1)$$

The Standard Radix-8 FFT algorithm [1] flop count is:

$$T(N) \approx 4\frac{1}{12} N \log_2 N \quad (2)$$

and standard split Radix FFT algorithm [1] flop count is:

$$T(N) = 4N\log_2 N - 6N + 8 \quad (3)$$

Here we present a modified version of the Radix-4/8 and split radix FFT algorithms that, is based on the DIF version of conjugate pair respective radix FFT algorithm [7], and it lowers the flop count for Radix-4 to:

$$T(N) = 4N \log_2 N - \frac{43}{6} N + \frac{32}{3} \quad (4)$$

for Radix-8 to:

$$T(N) \approx 3\frac{11}{12} N \log_2 N \quad (5)$$

and for split radix to:

$$T(N) \approx 3\frac{7}{9} N \log_2 N - 6N + 8 \quad (6)$$

The saving commence from N ≥ 64 are enumerated in Table.1. In this paper we first discuss the mathematical modeling

of the conjugate pair Radix-4, Radix-8, and split radix FFT, analyses its arithmetic cost, describe its implementation, then discuss a modification to save arithmetic complexity and finally conclude. The organization of paper is as follows: Conjugate-Pair Radix-4 FFT followed by its modified algorithm is discussed in section 2 and section 3 respectively. Section 4 covers the related Radix-8 FFT algorithm whereas section 5 covers split radix FFT algorithm followed by conclusions in section 6.

2. CONJUGATE PAIR RADIX-4 FFT

The starting point of our improved algorithm is not the standard Radix-4 FFT, but rather a variant called "Conjugate Pair FFT," that was initially proposed to provide saving in the number of twiddle factor evaluations [8], [3], [4], [5]. We use it for other reason (saving in the arithmetic complexity cost): To make use of symmetry in the twiddle factor due to shifting in reducing computation that is not feasible with normal Radix-4 FFT formulation. To derive the algorithm recall that the DFT is defined by,

$$A(k) = \sum_{i=0}^{N-1} a(i) W_N^{ik}, \quad 0 \le k \le N-1 \qquad (7)$$

where, W denotes the Nth root of unity, with its exponent evaluated modulo N, and $W_N^k = \exp(-j2\pi/N)$. Then for N divisible by 4, the DIF of the length-N DFT given by Eq.(1) can provide the first four length-N/4 DFT as,

$$A(k) = A(4k) + A(4k + 1) + A(4k + 2) + A(4k + 3) \qquad (8)$$

where the decomposed N/4 -point DFT sequences are defined as:

$$A(4k) = DFT_{\frac{N}{4}}\{ g_0(n)\},$$

$$A(4k+1) = DFT_{\frac{N}{4}}\{ g_1(n)\},$$

$$A(4k+2) = DFT_{\frac{N}{4}}\{ g_2(n)\}, and$$

$$A(4k+3) = DFT_{\frac{N}{4}}\{ g_3(n)\},$$

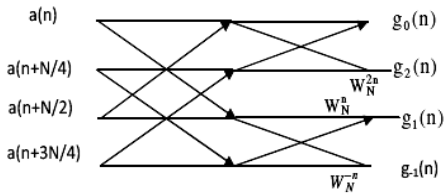

Fig.1. The new Radix-4 DIF FFT butterflies

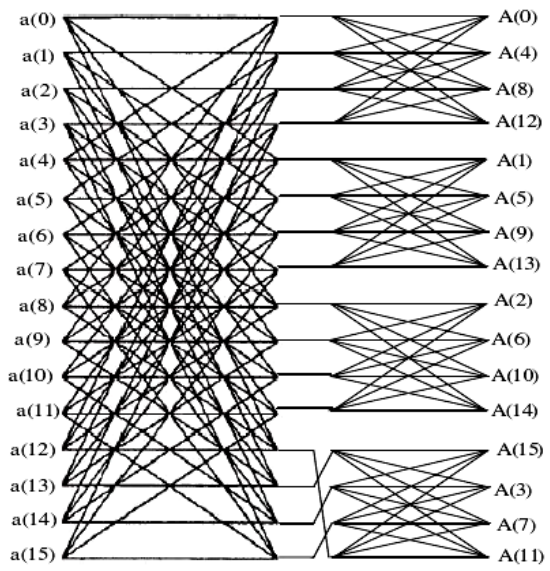

Fig.2. In Place computation diagram of the new Radix-4 DIF FFT algorithm for N=16

The input sequences $g_0(n)$ - $g_3(n)$ in matrix form can be expressed as:

$$\begin{bmatrix} g_0(n) \\ g_1(n) \\ g_2(n) \\ g_3(n) \end{bmatrix} = F_{4,n} \begin{bmatrix} a(n) \\ a(n+N/4) \\ a(n+N/2) \\ a(n+3N/4) \end{bmatrix} \qquad (9)$$

where,

$$F_{4,n} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ W_N^n & 1 & -j & -1 & j \\ W_N^{2n} & 1 & -1 & 1 & -1 \\ W_N^{3n} & 1 & j & -1 & -j \end{bmatrix} \qquad (10)$$

and n = 0 … (N/4-1). Eq.(8) can be modified by circular shift to $A(4k+3)$ to get Conjugate pair radix-4 FFT as,

$$A(k) = A(4k) + A(4k + 1) + A(4k + 2) + A((4k - 1))_N \qquad (11)$$

where, $A((4k-1))_N$ can be obtained as,

$$A((4k-1))_N = \begin{cases} N-1, & k = 0 \\ 4k-1, & \text{elsewhere} \end{cases} \qquad (12)$$

Now Eq.(10) becomes,

$$F_{4,n} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ W_N^n & 1 & -j & -1 & j \\ W_N^{2n} & 1 & -1 & 1 & -1 \\ W_N^{-n} & 1 & j & -1 & -j \end{bmatrix} \qquad (13)$$

2.1 IMPLEMENTATION ISSUES OF CONJUGATE PAIR RADIX-4 FFT ALGORITHM

The butterfly diagram for the computation of Eq.(11) can be represented by the two stages of computation is shown in Fig.1, and the complete in place computation diagram for the same algorithm for $N = 16$ is shown in Fig.2. These diagrams are used not only to compute the arithmetic complexity but also used for the implementation of algorithm. It is observed that the procedure is as that of standard FFT with an additional circular shift operation on last $N/4$-point output at every stage.

2.2 COMPUTATION COMPLEXITY OF ABOVE ALGORITHM

To determine the arithmetic cost of the radix-4 FFT algorithm given in [8], observed that in the two stages of butterfly computation (sub problem) shown in Fig.1 that three complex multiplications are required. Since the size of each sub problem is $N/4$, so $3N/4$ complex multiplications are performed during the first stage of butterfly computation. The total number of complex additions in the two stages will be $8N/4$. Thus, $3N/4$ complex multiplications and $8N/4$ complex additions are required to implement the butterfly computation shown in Fig.1.

Recall that the arithmetic cost of computer algorithm is measured by the number of real arithmetic operations, and that one complex addition incurs 2 real addition, and one complex multiplication incurs 3 real additions and 3 real multiplications [1]. Accordingly $9N/4$ real multiplications and $25N/4$ real additions are required per step. Thus, one step of this radix-4 DIF FFT algorithm requires $17N/2$ flops in total.

The cost of the radix-4 FFT algorithm given in [8] with the exclusion of trivial multiplication be represented by the following recurrence:

$$T(N) = \begin{cases} 4T(\frac{N}{4}) + 17\frac{N}{2} - 32, & if\ N = 4^n > 4 \\ 16, & if\ N = 4 \end{cases} \quad (14)$$

Solving by repeated substitution we get,

$$T(N) = 4\frac{1}{4} N \log_2 N - \frac{43}{6} N + \frac{32}{3} \quad (15)$$

3. MODIFIED CONJUGATE PAIR RADIX-4 FFT

We now present a modification to the conjugate pair, Radix-4 FFT algorithm. This is proved to be better also in terms of arithmetic complexity. The key to reduce the number of flop count is the observation of Eq.(13), that is the multiplier of both $g_1(n)$, and $g_3(n)$ are complex conjugate of each other, which can be rewritten as:

$$F_{4,n} = T_{N,n} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \quad (16)$$

where,

$$T_{N,n} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & W_N^n & 0 & 0 \\ 0 & 0 & W_N^{2n} & 0 \\ 0 & 0 & 0 & W_N^{-n} \end{bmatrix} \quad (17)$$

This means that we can rescale the size N/4 sub-transform by any factor $\frac{1}{S_{N/4}, n}$ desired, and absorb the scale factor into $[W_N^n\ S_{N/4},\ n]$ at no cost. We need to find a rescaling that will save some flops in the sub-transform. Consider a sub-transform of size N that we want to rescale by some factor $\frac{1}{S_{N/4}, n}$ for each of the output $g_i(n)$ where, $i = 0...N/4-1$. If $S_{N,n} = S_{N,n+N/4} = \cos(2\pi n/N)$ for $n \le N/8$, then

$$T_{N/4,n} = \begin{bmatrix} \frac{1}{S_{N/4}, n} & 0 & 0 & 0 \\ 0 & P_{N,n} & 0 & 0 \\ 0 & 0 & \frac{W_N^{2n}}{S_{N/4}, n} & 0 \\ 0 & 0 & 0 & P^*_{N,n} \end{bmatrix} \quad (18)$$

where, $P_{N,n}=1 - jtan(2\pi n/N)$. Here direct scaling does not give any saving however if we adopt the procedure as in [7], that is push the rescaling factor (required for compensation of scaling) down into the recursive computation of $g_i(n)$, we can save the extra multiplications by combining scaled $g_i(n)$ with twiddle factors inside the $N/2$ transform as shown in Fig.2.

Consider scaling/rescaling factor $S_{N,n}=cos(2\pi n/N)$, then $S_{N,n+N/2}= -S_{N,n}$ where, $n = 0...(N/4)-1$. For $N = 16$ $S_{N,0} = -S_{N,2}$ and $S_{N,1} = - S_{N,3}$. Hence the rescaling can be done on $N/2$ size transform instead of $N/4$, due to their symmetry condition to save multiplications. Note that for this, rescaling factor has to be pushed down through two levels of recursion.

To avoid the possible ill-conditioned problem the following dynamic scaling can be used:

$$S_{N,n} = \begin{cases} 1, & for\ N \le 4 \\ \cos(2\pi n/N), & for\ \ n \le N/8 \\ \sin(2\pi n/N), & otherwise \end{cases} \quad (19)$$

3.1 COMPUTATION COMPLEXITY OF MODIFIED ALGORITHM

Due to pushing the rescaling factor down by one step, required for the compensation of scaling as of [7], we can save 2 real multiplication per step, and the total flop count of the proposed algorithm will become $T(N) = 16\frac{1}{2} N$, and the cost of

the algorithm with the exclusion of trivial multiplication be represented by the following recurrence :

$$T(N)=\begin{cases}4T(\frac{N}{4})+16\frac{N}{2}-32, \text{ if } N=4^n>4\\ 16, \qquad\qquad \text{if } N=4\end{cases} \tag{20}$$

Solving Eq.(20) by repeated substitution we have,

$$T(N)=4N\log_2 N-\frac{13}{6}N+\frac{32}{3} \tag{21}$$

Resulting in a saving of almost 6%, as compare to standard Radix-4 FFT.

4. NEW CONJUGATE PAIR RADIX-8 FFT

In this section an improved conjugate pair Radix-8 FFT algorithm is given. As in the above Radix-4 FFT algorithm, here also the algorithm starts with its conjugate pair representation. The Eq.(7) for Radix-8 DIF FFT algorithm can be represented as:

$$A(k)=A(8k)+A(8k+1)+A(8k+2)+A(8k+3)+ A(8k+4)+A(8k+5)+A(8k+6)A(8k+7) \tag{22}$$

where the decomposed $N/8$-point DFT sequences are defined as:

$$A(8k)=DFT_{\frac{N}{8}}\{g_0(n)\},$$
$$A(8k+1)=DFT_{\frac{N}{8}}\{g_1(n)\},$$
$$A(8k+2)=DFT_{\frac{N}{8}}\{g_2(n)\},$$
$$\vdots$$
$$A(8k+7)=DFT_{\frac{N}{8}}\{g_7(n)\}$$

The input sequences $g_0(n)$ – $g_7(n)$ in matrix form can be expressed as,

$$g_i = P_{N,n}\,F_{8,n}\,a \tag{23}$$

where, $g_i=\begin{bmatrix}g_0(n)\\ g_1(n)\\ g_2(n)\\ g_3(n)\end{bmatrix}$,

$$P_{N,n}=\begin{bmatrix}1&0&0&0&0&0&0&0\\ 0&W_N^n&0&0&0&0&0&0\\ 0&0&W_N^{2n}&0&0&0&0&0\\ 0&0&0&W_N^{3n}&0&0&0&0\\ 0&0&0&0&W_N^{4n}&0&0&0\\ 0&0&0&0&0&W_N^{5n}&0&0\\ 0&0&0&0&0&0&W_N^{6n}&0\\ 0&0&0&0&0&0&0&W_N^{7n}\end{bmatrix} \tag{24}$$

$$F_{8,n}=\begin{bmatrix}1&1&1&1&1&1&1&1\\ 1&-j&-1&j&1&-j&-1&j\\ 1&-1&1&-1&1&-1&1&-1\\ 1&j&-1&-j&1&j&-1&-j\\ 1&\alpha&-j&-\alpha^*&-1&-\alpha&j&\alpha^*\\ 1&-\alpha^*&j&\alpha&-1&\alpha^*&-j&-\alpha\\ 1&-\alpha&-j&\alpha^*&-1&\alpha&j&\alpha^*\\ 1&\alpha^*&j&-\alpha&-1&-\alpha^*&-j&\alpha\end{bmatrix} \tag{25}$$

where, $\alpha=\sqrt{2}(1-j)$ and $a=\begin{bmatrix}a(n)\\ a(n+N/8)\\ a(n+N/4)\\ \dots\\ a(n+7N/8)\end{bmatrix}$, n = 0...(N/8)-1.

Eq.(22) can be modified by circular shift to $A(8k+5)$ … $A(8k+7)$ to get conjugate pair radix-8 FFT as,

$$A(k)=A(8k)+\dots A(8k+4)+A((8k-3))_N +A((8k-2))_N+A((8k-1))_N \tag{26}$$

where, $A((8k - x))_N$ can be obtained as,

$$A((8k-x))_N=\begin{cases}N-1, & k=0\\ 8k-x, & \text{elsewhere}\end{cases} \tag{27}$$

Now Eq.(24) becomes

$$P_{N,n}=\begin{bmatrix}1&0&0&0&0&0&0&0\\ 0&W_N^n&0&0&0&0&0&0\\ 0&0&W_N^{2n}&0&0&0&0&0\\ 0&0&0&W_N^{3n}&0&0&0&0\\ 0&0&0&0&W_N^{4n}&0&0&0\\ 0&0&0&0&0&W_N^{-3n}&0&0\\ 0&0&0&0&0&0&W_N^{-2n}&0\\ 0&0&0&0&0&0&0&W_N^{-n}\end{bmatrix} \tag{28}$$

In the implementation of this algorithm after observing Eq.(28), we can say that we need to perform additional circular shift operation on last N /8-point sequence at each stage. The cost of computation of conjugate pair Radix-8 FFT [8] is similar to that of standard Radix-8 [1], however the number of twiddle factor evaluation will be fewer than that of standard radix-8 FFT.

4.1 MODIFIED CONJUGATE PAIR RADIX-8 FFT ALGORITHM

In this subsection a method to modify the above algorithm to fetch efficiency in arithmetic complexity is given. Like radix-4 here also the key to reduce the number of flop counts is the

observation of Eq.(28), where the multiplier of $g_0(n)$ … $g_3(n)$ are complex conjugate of $g_5(n)$ ... $g_7(n)$, this again means that rescaled and claim saving of 2 real multiplication and 2 real additions in each step.

4.2 COMPUTATION COMPLEXITY OF MODIFIED CONJUGATE PAIR RADIX-8 FFT

In this case due to scaling 7 complex multiplications (see Eq.(24)) will transform to 6 partial complex multiplications(each partial complex multiplication take 2 real multiplications and 2 real additions) and 2 complex multiplications. However during rescaling one partial complex multiplication is required. This results in saving of 2 real additions in each step, also due to pushing the rescaling factor as in above radix-4 algorithm 2 real multiplication saving is added, and since the total flop count of standard radix- 8 FFT algorithm is $T(N) = 98(N/8)$, [1]. Hence now the total flop count of proposed radix-8 FFT algorithm will become $T(N) = 94(N/8)$. Finally the arithmetic cost of this algorithm is approximately represented as:

$$T(N) = \begin{cases} 8T(\frac{N}{8}) + 11\frac{3N}{4} - c, \text{ if } N = 8^n > 8, c > 0, \\ d, \qquad\qquad \text{if } N = 8, \text{and } d > 0 \end{cases} \tag{29}$$

Solving Eq.(29) by repeated substitution we have,

$$T(N) \approx 3\frac{11}{12} N \log_2 N \tag{30}$$

Resulting in a saving of almost 5%, as compare to standard Radix-8 FFT.

5. NEW CONJUGATE PAIR SPLIT RADIX FFT

In this section an improved complex conjugate based split radix FFT is given. Here also the algorithm starts with its conjugate pair representation. Rewriting Eq.(7) for this case:

$$A(k) = A(2k) + A(4k + 1) + A(4k + 3) \tag{31}$$

where the decomposed $N/8$-point DFT sequences are defined as:

$$A(2k) = DFT_{\frac{N}{2}}\{g_0(n)\},$$
$$A(4k+1) = DFT_{\frac{N}{4}}\{g_1(n)\},$$
$$A(4k+3) = DFT_{\frac{N}{4}}\{g_3(n)\}$$

The input sequences $g_0(n)$, $g_1(n)$ and $g_3(n)$ in matrix form can be expressed as,

$$g_0(n) = \left[a(n) + a(n + N/2)\right], n = 0 \ldots \frac{N}{2} - 1 \tag{32}$$

and

$$\begin{bmatrix} g_1(n) \\ g_3(n) \end{bmatrix} = \hat{F}_{4,n} \begin{bmatrix} a(n) \\ a(n+N/4) \\ a(n+N/2) \\ a(n+3N/4) \end{bmatrix}, n = 0 .. \frac{N}{4} - 1 \tag{33}$$

where,
$$\hat{F}_{4,n} = \begin{bmatrix} W_N^n - jW_N^n & -W_N^n & jW_N^n \\ W_N^{3n} & jW_N^{3n} & -W_N^{3n} - jW_N^{3n} \end{bmatrix} \tag{34}$$

Eq.(31) can be modified by circular shift to $A(4k + 3)$ to get conjugate pair split radix FFT as,

$$A(k) = A(2k) + A(4k + 1) + A(4k - 1) \tag{35}$$

where, $A((4k - 1))_N$ can be obtained as

$$A((4k-1))_N = \begin{cases} N-1, \ k = 0 \\ 4k-1, \text{ elsewhere} \end{cases} \tag{36}$$

Now Eq.(34) becomes

$$\hat{F}_{4,n} = \begin{bmatrix} W_N^n - jW_N^n & -W_N^n & jW_N^n \\ W_N^{-n} & jW_N^{-n} & -W_N^{-n} - jW_N^{-n} \end{bmatrix} \tag{37}$$

The butterfly diagram for the computation of Eq.(35) is shown in Fig.3. It is observed that it is similar to that of standard split radix FFT with an additional circular shift operation on last $N/4$-point output at every stage. The computation cost of conjugate pair split radix FFt is similar to that of standard split radix FFT, however like earlier here aslo the evaluation of twiddle factor is comparatively less.

5.1 MODIFIED CONJUGATE PAIR SPLIT RADIX FFT

Like earlier here in this subsection the method for modification to get improvement in the computational efficiency is discussed. Due to the availability of complex conjugate pair in Eq.(37), saving of two real multiplications per steps is obtained by rescaling like earlier.

5.2 COMPUTATION COMPLEXITY OF MODIFIED CONJUGATE PAIR SPLIT RADIX FFT

As earlier due to pushing the rescaling factor by one step in each stage of the FFT computation, saving of two real multiplications is achieved, and since the flop count of standard split radix FFT algorithm is $6N - 16$, [1].

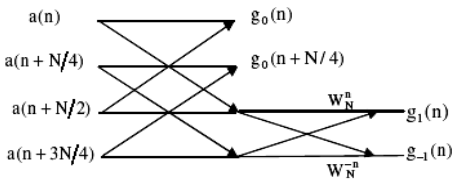

Fig.3. The new Split Radix DIF FFT butterflies

Hence now, the total flop count of the proposed algorithm will become $5.5N - 16$. Finally the cost of this split radix FFT algorithm (in terms of nontrivial flops) can be represented by the following recurrence:

$$T(N) = \begin{cases} T(\frac{N}{2}) + 2T(\frac{N}{4}) + 5.5N - 16, \text{ if } N = 4^n > 8, \\ 16, \qquad\qquad \text{if } N = 4, \text{and} \\ 4, \qquad\qquad \text{if } N = 2 \end{cases} \tag{38}$$

Solving Eq.(38) by repeated substitution we have

$$T(N) = 3\frac{7}{9} N \log_2 N - 6N + 8 \quad (39)$$

Resulting in a saving of almost 5.5%, as compare to standard Split radix FFT.

6. CONCLUSION

In this paper alternate form of Radix-4/8, and split radix FFT algorithms with their mathematical modeling is presented. The arithmetic complexity of the proposed algorithm is proved to be significantly less than the standard FFT algorithms. The approached of these algorithms is similar to that of Johnson and Frigo, however it is of practical importance as we are covering symmetrical algorithms, in addition to split radix FFT algorithm. Also in the proposed algorithms the inherited feature is the reduction of number of twiddle factors, which is another useful feature in the implementation of FFT architecture.

REFERENCES

[1] E. Chu and A. George, "*Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*," CRC press, USA, 2000.

[2] Y.N. Chang and K.K. Parhi,"An efficient pipelined FFT architecture,"*IEEE Transactions on Circuits System II* , Vol. 50, No. 6, pp. 322-325, 2003.

[3] I. Kamar and Y. Elcherif, "Conjugate pair fast Fourier transform", *Electronics Letters*, Vol. 25, No. 5, pp. 324-325, 1989.

[4] R. A. Gopinath, "Conjugate pair fast Fourier transform," *Electronics Letters*, Vol. 25, No. 16, pp. 1082- 1084, 1989.

[5] A.M. Krot and H. B. Minervina, "Comment: Conjugate pair fast Fourier transform," *Electronics Letters*, Vol. 28, No. 12, pp.1143-1144, 1992.

[6] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, "An Efficient Locally Pipelined FFT Processor," *IEEE Transactions on circuits and Systems-II: Express Briefs*, Vol. 53, No. 7, pp. 585-589, 2006.

[7] S. G. Johnson and M. Frigo, "A modified Split-radix FFT with fewer arithmetic operations," *IEEE Transactions on Signal Processing*, Vol. 55, No.1, pp. 111-119, 2007.

[8] S. Bouguezel, M.O. Ahmad and M.N.S Swamy, "Improved Radix-4 and Radix-8 FFT algorithms," *in Proceedings of IEEE, on International Symposium on Circuits and Systems*, Vol. 3, pp. III-561-4, 2004.

[9] Q. Li, N. Wang, B. Shi, and C. Zheng, "Extendible look-up table of twiddle factors and radix-8 based Fourier transform," *IEEE Transactions on Acoustics Speech, Signal Processing*, Vol. 82, No. 4, pp. 643-648, 2002.

[10] Qadeer, S. and Khan, M.Z.A., "Fixed Point error analysis of Radix-4 and Radix-8 FFT algorithms," *IEEE International Conference on Multimedia, Signal Processing and Communication Technologies*, pp. 32-35, 2011.