Open access • Proceedings Article • DOI:10.1109/APSIPA.2013.6694148

The split-radix fast Fourier transforms with radix-4 butterfly units

Sian-Jheng Lin, Wei-Ho Chung

Institutions: Center for Information Technology

Published on: 01 Oct 2013 - Asia-Pacific Signal and Information Processing Association Annual Summit and Conference

Topics: Split-radix FFT algorithm, Prime-factor FFT algorithm, Mixed radix, Cooley–Tukey FFT algorithm and Twiddle factor

The Split-Radix Fast Fourier Transforms with Radix-4 Butterfly Units

Sian-Jheng Lin* and Wei-Ho Chung*
*Research Center for Information Technology, Academia Sinica, Taipei City, Taiwan, R.O.C.
E-mail: {sjlin; whc}@citi.sinica.edu.tw

***Abstract*—We present a split radix fast Fourier transform (FFT) algorithm consisting of radix-4 butterflies. The major advantages of the proposed algorithm include: i). The proposed algorithm consists of mixed radix butterflies, whose structure is more regular than the conventional split radix algorithm. ii). The proposed algorithm is asymptomatically equal computation amount to the split radix algorithm, and is fewer operations than the radix-4 algorithms. iii). The proposed algorithm is in the conjugate-pair version, which requires less memory access than the conventional FFT algorithms.**

I. Introduction

The fast Fourier transform (FFT) algorithm has been widely adopted in digital signal processing and multimedia applications. With the widespread utilization of FFT, many techniques have been proposed to speed up the FFT algorithm in recent years. The major metrics to measure the performance of FFT structure include the arithmetic complexity, the recursive structure of FFT algorithm, and the overhead for memory access. Based on Cooley-Tukey algorithm [1], the radix-2 algorithm is proposed for the input sequence with length 2^i. To further reduce the cost of multiplying twiddle factors, a number of FFT algorithms, such as mixed-radix and split-radix [2]-[7], have been proposed. The mixed-radix 4 and split-radix 2/4 are two well-known algorithms for the input sequence with length 4^i. The radix-4 algorithm is constructed based on 4-point butterfly units. The 4-point butterfly consists of four 2-point butterflies without any multipliers. Hence the radix-4 takes fewer operations than the ordinary radix-2 does. Furthermore, the split-radix 2/4 algorithm requires fewer operations than radix-4, but its irregular recursive structure hinders the efficiency on general-purpose computers. Table I summarizes the arithmetic complexities of those algorithms and the proposed algorithms. Based on the two perspectives described above, this work proposes to design a new FFT algorithm with combining the advantages of radix-4 and split-radix. The proposed FFT algorithm is built from radix-4 butterflies, and it has the same asymptotic complexity as split radix algorithm. Based on the conjugate-pair split-radix [6] and mixed-radix [8], the proposed FFT algorithm is formulated as the conjugate-pair version to reduce the overhead of loading twiddle factors. Furthermore, we also propose a modified FFT algorithm taking the same operation counts as the radix-2/4, but requiring a special 4-point butterfly in the recursive flow graph. The rest of this paper is organized as follows. Section II introduces the proposed FFT algorithm. Section III gives the operation counts of the proposed algorithm. Section IV introduces a 4-point special butterfly to reduce the arithmetic complexity. Section V discusses related issues, such as the overhead of loading twiddle factors, and the implementation of the proposed algorithm. Section V summarizes this work.

TABLE I
The arithmetic complexities of Radix-2, -4, 2/4 FFTs and the proposed algorithm

FFT algorithms	Operation counts
Radix-2	$5N \lg N - 10N + 16$
Radix-4	$4.25N \lg N - 43/6N + 32/3$
Radix-2/4	$4N \lg N - 6N + 8$
Proposed alg.	$4N \lg N - (88/15)N - 0.58 * (-1)^{\log_4 N} + 20/3$
Modified alg.	$4N \lg N - 6N + 8$

II. Proposed split-radix algorithm with radix-4 butterflies

Given a complex data sequence $x(k), k = 0, 1, \ldots, N-1$ with length N, the discrete Fourier transform (DFT) is defined as

$$X(k) = \sum_{i=0}^{N-1} x(i) W_N^{ik}, \text{ for } k = 0, 1, \ldots, N-1, \qquad (1)$$

where $W_N = exp(j2\pi/N)$, and $j = \sqrt{-1}$. Throughout this paper, we define that a complex multiplication operation requires four real multiplications and two real additions. Furthermore, it is noted that multiplying by $\{W_4^0, W_4^1, W_4^2, W_4^3\}$ is treated as free of calculation, and multiplying by $\{W_8^1, W_8^3, W_8^5, W_8^7\}$ only requires two multiplications and two additions. In the following, we firstly introduce the modified radix-4 FFT proposed by Bouguezel et al. [8], and then introduce the proposed algorithm derived from [8]. Assuming N being the power of four, then (1) can be reformulated as

$$\begin{aligned} X(k) &= \sum_{l=0}^{N/4-1} \sum_{i=0}^{3} x(N/4 \times i + l) W_N^{(N/4 \times i + l)k} \\ &= \sum_{l=0}^{N/4-1} W_N^{lk} \sum_{i=0}^{3} x(N/4 \times i + l) W_4^{lk}. \end{aligned} \qquad (2)$$

We substitute (2) with $k = 4p + q$ for $q = 0, 1, 2, 3$ to obtain

$$\begin{aligned} X(4p+q) &= \sum_{l=0}^{N/4-1} W_N^{l(4p+q)} \sum_{i=0}^{3} x(N/4 \times i + l) W_4^{l(4p+q)} \\ &= \sum_{l=0}^{N/4-1} W_N^{l(4p+q)} \sum_{i=0}^{3} x(N/4 \times i + l) W_4^{lq}. \end{aligned} \tag{3}$$

To emphasize the recursive structure, we define

$$\tilde{y}_q^2(l) = \sum_{i=0}^{3} x(N/4 \times i + l) W_4^{lq}. \tag{4}$$

Then (3) can be expressed as

$$\begin{aligned} X(4p+q) &= \sum_{l=0}^{N/4-1} W_N^{l(4p+q)} \tilde{y}_q^2(l) \\ &= \begin{cases} \sum_{l=0}^{N/4-1} W_{N/4}^{lp} (W_N^{lq} \tilde{y}_q^2(l)) & \texttt{if } q \in \{0,1,2\}; \\ \sum_{l=0}^{N/4-1} W_{N/4}^{l(p+1)} (W_N^{-l} \tilde{y}_q^2(l)) & \texttt{if } q \in \{3\}; \end{cases} \\ &= \begin{cases} \sum_{l=0}^{N/4-1} W_{N/4}^{lp} y_q^2(l) & \texttt{if } q \in \{0,1,2\}; \\ \sum_{l=0}^{N/4-1} W_{N/4}^{l(p+1)} y_q^2(l) & \texttt{if } q \in \{3\}; \end{cases} \end{aligned} \tag{5}$$

where

$$y_q^2(l) = \begin{cases} W_N^{lq} \tilde{y}_q^2(l) & \texttt{if } q \in \{0,1,2\}; \\ W_N^{-l} \tilde{y}_q^2(l) & \texttt{if } q \in \{3\}. \end{cases} \tag{6}$$

For $q = 3$, the standard radix 4 uses the twiddle factor W_N^{3l}, as opposed to W_N^{-l} shown in (6). This variant calculates the same DFT outputs with a different order. As the W_N^{-l} can be obtained by reversing the sign of imaginary part of W_N^l, the loading cost can be saved.

By (4), the matrix form of $\{\tilde{y}_0^2(l), \tilde{y}_1^2(l), \tilde{y}_2^2(l), \tilde{y}_3^2(l)\}$ is expressed as

$$\begin{bmatrix} \tilde{y}_0^2(l) \\ \tilde{y}_1^2(l) \\ \tilde{y}_2^2(l) \\ \tilde{y}_3^2(l) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} x(l) \\ x(N/4+l) \\ x(2N/4+l) \\ x(3N/4+l) \end{bmatrix} \tag{7}$$

The (7) is computed via the 4-point butterfly shown in Fig. 1(a). Since the multiplication by j is free, the 4-point butterfly only requires eight times of complex additions. As shown in [7], the radix-4 algorithm takes $N \lg N$ complex additions and $3/8N \lg N - N + 1$ complex multiplications.

To further reduce the complexities of above radix-4 algorithm, the term $X(4p+2)$ of (5) is decomposed into

$$\begin{aligned} X(4p+2) &= \sum_{l=0}^{N/4-1} W_N^{l(4p+2)} \tilde{y}_2^2(l) \\ &= \sum_{k=0}^{N/16-1} \sum_{l=0}^{3} W_N^{(N/16 \times l + k)(4p+2)} \tilde{y}_2^2(N/16 \times l + k) \\ &= \sum_{k=0}^{N/16-1} W_{N/2}^{k(2p+1)} \sum_{l=0}^{3} W_8^{l(2p+1)} \tilde{y}_2^2(N/16 \times l + k) \end{aligned} \tag{8}$$

We substitute (8) with $p = 4p_1 + q_1$, $q_1 \in \{0, 1, 2, 3\}$ to obtain

$$\begin{aligned} & X(16p_1 + 4q_1 + 2) \\ &= \sum_{k=0}^{N/16-1} W_{N/2}^{k(8p_1+2q_1+1)} \sum_{l=0}^{3} W_8^{l(8p_1+2q_1+1)} \tilde{y}_2^2(N/16 \times l + k) \\ &= \sum_{k=0}^{N/16-1} W_{N/2}^{k(8p_1+2q_1+1)} \sum_{l=0}^{3} (W_8^l \tilde{y}_2^2(N/16 \times l + k)) W_4^{lq_1}, \end{aligned} \tag{9}$$

To emphasize the recursive structure in (9), we define

$$\tilde{y}_{q_1}^4(k) = \sum_{l=0}^{3} (W_8^l \tilde{y}_2^2(N/16 \times l + k)) W_4^{lq_1} \tag{10}$$

Then (9) is formulated as

$$\begin{aligned} X(16p_1 + 4q_1 + 2) &= \sum_{k=0}^{N/16-1} W_{N/2}^{k(8p_1+2q_1+1)} \tilde{y}_{q_1}^4(k) \\ &= \begin{cases} \sum_{k=0}^{N/16-1} W_{N/16}^{kp_1} [W_{N/2}^{k(2q_1+1)} \tilde{y}_{q_1}^4(k)] & \texttt{if } q_1 \in \{0,1\}; \\ \sum_{k=0}^{N/16-1} W_{N/16}^{k(p_1+1)} [W_{N/2}^{k(2q_1-7)} \tilde{y}_{q_1}^4(k)] & \texttt{if } q_1 \in \{2,3\}; \end{cases} \\ &= \begin{cases} \sum_{k=0}^{N/16-1} W_{N/16}^{kp_1} y_{q_1}^4(k) & \texttt{if } q_1 \in \{0,1\}; \\ \sum_{k=0}^{N/16-1} W_{N/16}^{k(p_1+1)} y_{q_1}^4(k) & \texttt{if } q_1 \in \{2,3\}; \end{cases} \end{aligned} \tag{11}$$

where

$$y_{q_1}^4(k) = \begin{cases} W_{N/2}^{k(2q_1+1)} \tilde{y}_{q_1}^4(k) & \texttt{if } q_1 \in \{0,1\}; \\ W_{N/2}^{k(2q_1-7)} \tilde{y}_{q_1}^4(k) & \texttt{if } q_1 \in \{2,3\}. \end{cases} \tag{12}$$

For $q_1 \in \{2, 3\}$ in (9), the $y_{q_1}^4(k)$ is multiplied by the twiddle factors $W_{N/2}^{-3k}$ and $W_{N/2}^{-k}$, rather than $W_{N/2}^{5k}$ and $W_{N/2}^{7k}$ adopted in conventional split radix. The $W_{N/2}^{-3k}$ and $W_{N/2}^{-k}$ are the complex conjugates of $W_{N/2}^{3k}$ and $W_{N/2}^{k}$, so the values can be easily obtained without computations or table lookup. In (10), multiplying by $W_8^l = (1 \pm j)/\sqrt{2}$ only requires two multiplications and two additions, so the proposed algorithm has fewer operations than radix-4. In summary, Figure 2 illustrates the general flow graph of the proposed FFT. The 16-point input is given as $\{x(k + iN/16)\}_{i=0}^{15}$, and the output are the 12 points $\{y_q^2(k + iN/16) | q \in \{0, 1, 3\}, i \in \{0, 1, 2, 3\}\}$ defined by (6), and 4 points $\{y_{q_1}^4(k)\}_{q_1=0}^{3}$ defined by (12). There exists a special case $k = 0$. In this case, for $q \in \{1, 3\}$ in (6), we have several specific twiddle factors: for $l = 0$, the twiddle factor is one; and for $l = 2N/16$, the twiddle factor is $W_8^{\pm 1}$. For $q_1 \in \{0, 1, 2, 3\}$ in (12), all the twiddle factors are turned into one. Evidently, there are six twiddle factors turned into 1 and two twiddle factors turned into $W_8^{\pm 1}$, so we can save several operations at $k = 0$. In summary, the proposed algorithm is constructed from a general flow graph for $k \in \{1, 2, \ldots, N/16 - 1\}$, a special flow graph for $k = 0$, and the 4-point butterfly.

An important issue is the order of the output values. The order is different from the conventional radix-4 algorithms. For example, for q=3 in (5) and $q_1 \in \{2, 3\}$ in (11), the order of output is shifted cyclically by one. Table II gives an algorithm to calculate the location of each $X(k)$.

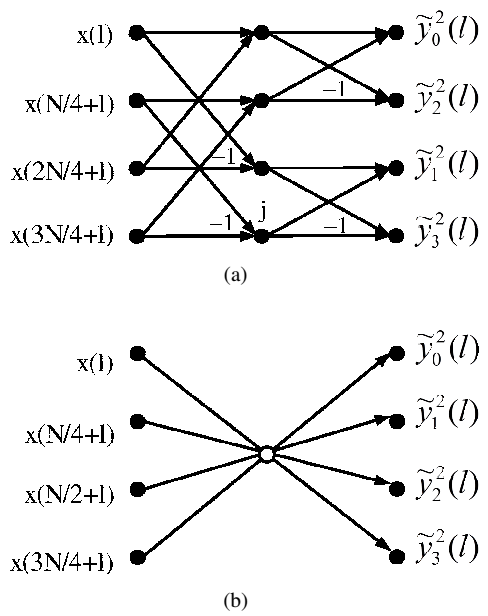

Fig. 1. Radix-4 butterfly. (a). The radix-4 butterfly constructed with radix-2 butterflies. (b). A simplified representation.

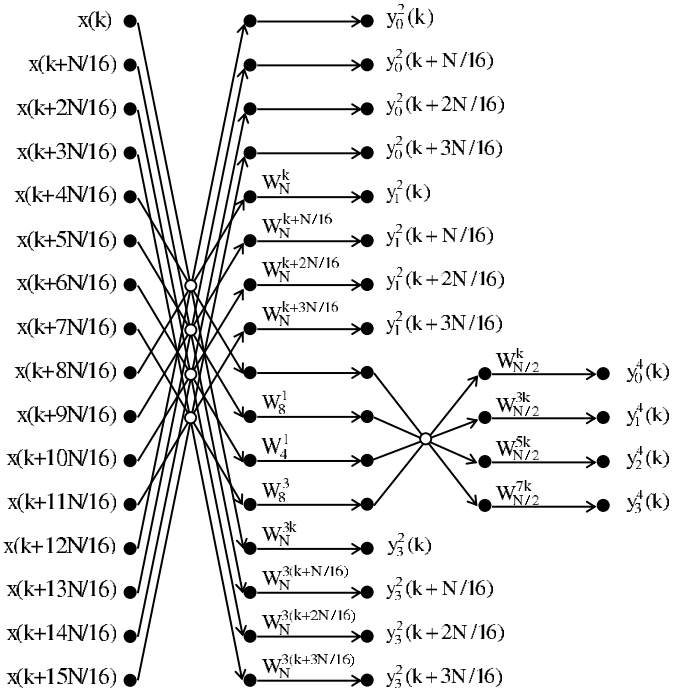

Fig. 2. General flow graph of the proposed split-radix algorithm based on radix-4 butterflies.

TABLE II
THE ALGORITHM OF CALCULATING THE POSITION OF X(K) IN THE OUTPUT SEQUENCE.

Algorithm 1: Calculate the position of X(k) in the output sequence.

Input: an integer k ∈[0, N).

Output: an integer d indexing the position of X(k).

d←0
For i=0 to ($\log_4$N−1) **Then**
  r←k (mod 4)
  k←⌊k/4⌋
  d←4d+r
  If r=3 **Then** k←k+1 **Endif**
  If r=2 and i<($\log_4$N−1) **Then**
    r←k (mod 4)
    k←⌊k/4⌋
    d←4d+r
    If r= 2 or 3 **Then** k←k+1 **Endif**
    i←i+1
  Endif
Endfor
Return d

III. OPERATION COUNTS OF THE PROPOSED ALGORITHM

It can be shown that the general flow graph depicted in Fig. II takes 52 real multiplications and 108 real additions, and the special flow graph for $k = 0$ takes 24 real multiplications and 96 real additions. Let $M(N)$ and $A(N)$ respectively denote the amount of used multiplications and additions in the proposed N-point algorithm. Given a sequence with N points, the first stage of the proposed algorithm decomposes the N-point DFT into three $N/4$-point DFTs and four $N/16$-point DFTs as shown in Fig. II. The first stage employs $(N/16 - 1)$ general flow graphs for $k = 1, 2, \ldots, (N/16 - 1)$, and a special flow graph for $k = 0$. Thus, the recursive function is given by

$$M(N) = 3M(N/4) + 4M(N/16) + (N/16 - 1)52 + 24; \quad M(1) = M(4) = 0. \tag{13}$$

$$A(N) = 3A(N/4) + 4A(N/16) + (N/16 - 1)108 + 96; \quad A(1) = 0; A(4) = 16. \tag{14}$$

The solutions of the recurrence are shown to be

$$M(N) = 1.3N \lg N - (296/75)N - 0.72(-1)^{\log_4 N} + 14/3. \tag{15}$$

$$A(N) = 2.7N \lg N - 1.92N - 0.08(-1)^{\log_4 N} + 2. \tag{16}$$

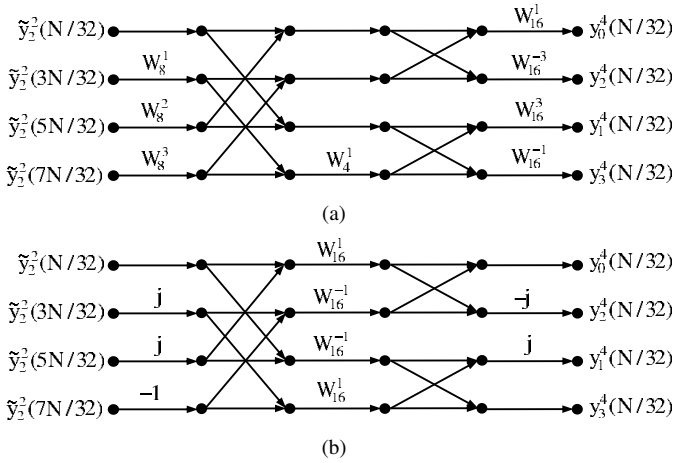

Fig. 3. The sub-flow graph for the case $k = N/32$. (a). The original butterfly. (b). The modified version.

Thus, the total of operations is

$$\begin{aligned} &T(N) = M(N) + A(N) \\ =&4N\lg N - (88/15)N - 0.8(-1)^{\log_4 N} + 20/3. \end{aligned} \tag{17}$$

As listed in Table I, the proposed algorithm takes the same asymptotic complexity as the radix-2/4. Notably, the [9] presents a FFT with fewer arithmetic operations (about $34/9 \times N \lg N$). However, the butterfly structure of [9] is more complex, and the output have accuracy problem (see Sec. V of [9]).

IV. The customized 4-point butterfly for k=N/32

For $k = N/32$, the operation amount can be reduced further by using a special butterfly. In this case, Fig. 3(a) depicts the butterfly for (12) at $k = N/32$, and Fig. 3(b) depicts the modified butterfly. It can be verified that the customized butterfly outputs the same values as the original butterfly does. As shown in Fig. 3(a)(b), the modified butterfly saves two complex multiplications. Thus, the number of operations of the modified algorithm is

$$\begin{aligned} &\hat{M}(N) = 3\hat{M}(N/4) + 4\hat{M}(N/16) + 3.25N - 32; \\ &M(4) = 0; M(16) = 24. \end{aligned} \tag{18}$$

$$\begin{aligned} &\hat{A}(N) = 3\hat{A}(N/4) + 4\hat{A}(N/16) + 6.75N - 16; \\ &A(4) = 16; A(16) = 144. \end{aligned} \tag{19}$$

It is noted that the case $k = N/32$ occurs when $N \geq 32$, so the recurrences (18)(19) start from $N = 4$ and $N = 16$. The solutions are

$$\hat{M}(N) = 1.3N\lg N - (301/75)N - 0.32(-1)^{\log_4 N} + 5.33. \tag{20}$$

$$\hat{A}(N) = 2.7N\lg N - (149/75)N + 0.32(-1)^{\log_4 N} + 2.67. \tag{21}$$

Then the arithmetic complexity is

$$\hat{T}(N) = \hat{M}(N) + \hat{A}(N) = 4N\lg N - 6N + 8, \tag{22}$$

which is the same as radix-2/4 listed in Table I. Consequently, the number of operations of the modified version is exactly the same as the radix-2/4 algorithm, but it requires a special butterfly for the case $k = N/32$. In summary, the customized butterfly saves about $(2/15 \times N)$ operations.

V. Discussions

A. Number of times of loading twiddle factors

In counting the times of loading twiddle factors by table lookup, it is assumed that the constant twiddle factors are available in the registers of processors during the FFT calculation, so we only count the number of non-constant twiddle factors used in the flow graph. A complex number consists of a real part and an imaginary part, so loading a twiddle factor requires two times of table lookup. When a complex number is loaded to a register, its conjugate can be easily obtained through reversing the sign of imaginary value, so the loading of conjugate value is free. Based on the above assumptions, it can be shown that the general flow graph requires 12 inevitable times of table lookup. For the special flow graph $k = 0$, all the twiddle factors are constants, so the table lookup is not needed. Thus, we have the following recursive function:

$$\begin{aligned} &S(N) = 3S(N/4) + 4S(N/16) + (N/16 - 1) \times 12 \\ &S(1) = S(4) = 0. \end{aligned} \tag{23}$$

The solutions is

$$S(N) = 0.3N\lg N - 1.28N - 0.72(-1)^{\log_4 N} + 2. \tag{24}$$

B. Benefits of proposed FFT algorithms in parallel computing

The proposed algorithm has another advantage in parallel computing. Let the G denote a flow graph constructed by the proposed N-point algorithm. As the 4-point butterfly has no multiplication operation in its middle stage (see Fig. 1(a)), the G does not require multiplications in its odd stages. Thus, in hardware design, the G does not need the clock cycles for multiplications in odd stages, and the computing time can be reduced further. In contrast, the radix-2/4 has multiplication operations in all stages, except for the first and the last stages, so the radix-2/4 cannot save the clock cycles for the multiplications.

VI. Conclusions

This paper presents a split-radix FFT algorithm based on radix-4 butterfly units. The arithmetic complexity of the proposed algorithm achieves the same asymptotic complexity as split radix algorithm. We also propose a modified algorithm to exactly achieve the same number of operations as the radix-2/4, but it should employ a special 4-point butterfly. The proposed algorithm can be efficiently implemented upon elaborate 4-point butterfly units. The theoretical perspective is also discussed to show that the proposed algorithm explores the relationships between radix-4 and radix-2/4 algorithm.

REFERENCES

[1] James W.Cooley, and John W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Mathematics of Computation, vol. 19, no. 90, pp. 297301, Apr. 1965.

[2] R. Yavne, "An economical method for calculating the discrete Fourier transform," *in Proc. AFIPS Fall Joint Computer Conf.*, vol. 33, pp. 115125, 1968.

[3] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," *Electronics Letters*, vol. 20, no. 1, pp. 1416, 1984.

[4] D. Takahashi, An extended split-radix FFT algorithm, *IEEE Signal Processing Letters*, vol. 8, no. 5, pp. 145147, May 2001.

[5] S. Bouguezel, M.O. Ahmad and M.N.S. Swamy, A new radix-2/8 FFT algorithm for length-q2m DFTs, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 9, pp. 17231732, Sept. 2004.

[6] I. Kamar, Conjugate pair fast Fourier transform, *Electronics Letters*, vol. 25, no. 5, pp. 324325, Mar. 1989.

[7] T. Mateer, Fast Fourier transform algorithms with applications, Ph.D. dissertation, Clemson University, Clemson, SC, 2008.

[8] S. Bouguezel, M. O. Ahmad, and M.N.S. Swamy, Improved radix-4 and radix-8 FFT algorithms, *in Proceedings of the 2004 International Symposium on Circuits and Systems*, vol. 3, pp. (III-561)(III-564), 2004.

[9] S.G. Johnson and M. Frigo, "A modified split-radix FFT with fewer arithmetic operations," *IEEE Trans. Signal Processing*, vol. 55, no. 1, pp.111-119, 2007.