Basrah Journal for Engineering Sciences, Vol. 20, No. 2, (2020), 66-73

Basrah Journal for Engineering Sciences

Journal homepage: www.bjes.edu.iq

ISSN (Online): 23118385, ISSN (Print): 18146120

OPEN ACCESS

New Design of Low Complexity Multipliers in DFT/IDFT

Majid A. Alwan [1]

[1]*Department of Electrical Engineering, College of Engineering, University of Basrah, Basrah, Iraq*

E-mail address: altimimee174@gmail.com

Received: 21 January 2020; Accepted: 26 February 2020; Published: 2 March 2020

Abstract

This paper presents a new design to implement DFT/IDFT using the two components of a sequence, which are even and odd component sequences to solve the complexity of complex multiplications and reduce the number of multipliers. The proposed two implementations reduce the number of real multipliers needed to compute the DFT. The first proposed design gives good results for N ≤ 512 as compared to conventional FFT algorithm, while the second scenario gives good results for N ≤ 1024 as compared to conventional FFT algorithm. The proposed design is performed directly from real and imaginary part equations of the DFT sequence *X* [*k*] without additional processing.

Keywords: Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Complex Multiplier, Even and Odd Sequences.

1. Introduction

There is a very important role played by Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) in many digital signal-processing applications. The DFT and IDFT are widely used in signal processing applications such as spectrum analysis of signals, OFDM system, power spectrum estimation, and linear filtering. The existence of computationally efficient algorithms for computing DFT and IDFT, Fast Fourier transform (FFT) algorithms increase the importance of using this transform in practical applications. The FFT algorithm is generally breaking (decomposing) the transform into smaller transforms. The complete process is performed by combining these smaller transform components to give the total transform. By FFT, the number of complex multiplications is reduced to $\frac{N}{2}log_2 N$ (rather than N^2 in direct DFT computing) and 4 real multiplications are needed to implement each complex multiplication $(a + jb)(c + jd) = (ac - bd) + j(bc + ad)$.

The facilities of VLSI and FPGA techniques were used in last 10 years to improve FFT performance. A 32-point FFT were designed and implemented with Canonical Sign Digit (CSD) and Dual edge trigged flip-flop to reduce the complexity of multiplication. Braun multipliers were implemented in FFT design by Anitha [2]. This design has disadvantage of binary floating-point multiplications high complexity.

A pipelined reconfigurable processor is designed by Wang [3] for implementing variable-length single-precision floating-point FFT/IFFT and DCT/IDCT computations. It is compatible with the IEEE754 standard. The number of adders is reduced by 75 % by the proposed radix-4 butterfly (RR4BF) as compared to the conventional parallel radix-4 butterfly.

A mixed-radix FFT algorithm with the single-sided binary-tree decomposition is implemented by Wei [4] to reduce the complexity of multiplications for 2^k – point FFT. For this assistance, parallel processing of the twiddle factor is generated and the dual addition and rounding floating point FP arithmetic units are improved to meet the demand for high accuracy calculation and low energy budget in execution.

Sivanandam and Kumar [5], an FFT butterfly structure is implemented utilizing the Vedic multiplier for high-speed applications. Urdhava Triyakbhyam algorithm is used to improve the Vedic multiplier efficiency. The FPGA implementation for Vedic multiplier shows that it reduced 35 % of the delay compare to the Booth multiplier for 16 × 16 multiplications.

Beyond this introductory section, there are five other sections. Section II reviews DFT in terms of Circular symmetry, The Discrete Fourier Transform principles, and the Twiddle Factors. Section III depicts the proposed implementation I and II. Section IV gives the IDFT implementation. Section V summarizes the obtained results. Finally, section VI concludes the paper.

2. Review of Sequence and DFT

2.1. Circular Symmetry

In general, if time-reversal on sequence results in an identical sequence, the sequence has even symmetry; the sequence has odd symmetry, if time-reversal changes the signs of the samples. Thus, the difference between linear and circular time-reversal has implications on the definition of symmetry for finite-length sequences. For sequences defined for all *n* (time index), symmetry is determined about the point $n = 0$. In the circular framework, symmetry is determined with respect to the circle diameter passing through the point $n = 0$. Thus, for a finite-length real-valued sequence $x[n]$, circular symmetry is defined by the conditions:

$x[n] = x[\langle -n \rangle_N]$, circular even symmetry (1)

$x[n] = -x[\langle -n \rangle_N]$, circular odd symmetry (2)

Where: $\langle n \rangle \triangleq n$ modulo N

From Fig. 1, the circular time reversal, which is known as Circular Folding, is defined by:

$$x[\langle -n \rangle] \triangleq \begin{cases} x[0] & n = 0 \\ x[N-n] & 1 \le n \le n-1 \end{cases} \tag{3}$$

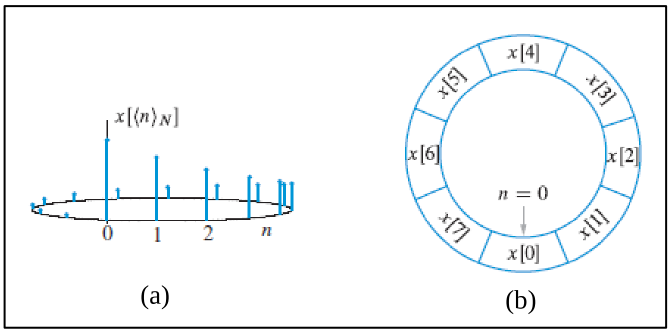


Fig. 1 Circular wrapping: (a) wrapping the sequence $x[n]$ around a cylinder with circumference N and using modulo-N addressing (b) representation of a circular buffer with modulo-N indexing.

Any N-point real sequence $x[n]$ can be decomposed into a sum of even $x^e[n]$ and odd $x^o[n]$ components as [6]:

$$x[n] = x^e[n] + x^o[n] \qquad 0 \le n \le N-1 \tag{4}$$

Where,

$$x^e[n] \triangleq \frac{x[n] + x[\langle -n \rangle_N]}{2}$$

$$= \begin{cases} x[0] & n = 0 \\ \frac{1}{2}(x[n] + x[N-n]) & 1 \le n \le n-1 \end{cases} \tag{5}$$

and

$$x^o[n] \triangleq \frac{x[n] - x[\langle -n \rangle_N]}{2}$$

$$= \begin{cases} 0 & n = 0 \\ \frac{1}{2}(x[n] - x[N-n]) & 1 \le n \le n-1 \end{cases} \tag{6}$$

Figure 2 shows the even and odd sequences of 8 – point sequence $x[n]$ using (5) and (6), respectively, and $x^e[n]$ can be written for even N :

$$x^e[n] = \left\{x[0], x^e[1], x^e[2], \dots \dots, x^e\left[\frac{N}{2} - 1\right], x\left[\frac{N}{2}\right], x^e\left[\frac{N}{2} - 1\right], \dots \dots, x^e[2], x^e[1]\right\} \tag{7}$$

and $x^o[n]$ can be written for even N :

$$x^o[n] = \left\{0, x^o[1], x^o[2], \dots \dots, x^o\left[\frac{N}{2} - 1\right], 0, -x^o\left[\frac{N}{2} - 1\right], \dots \dots, -x^o[2], -x^o[1]\right\} \tag{8}$$

In order to implement the even component of the sequence $x[n]$, need only $(\frac{N}{2} - 1)$ adders, and the same number of subtracters to generate the odd component of the sequence.

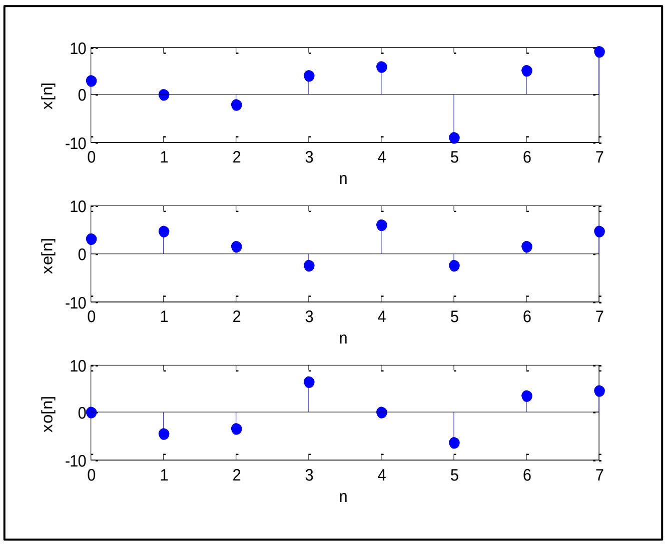


Fig. 2 The even component and odd component of the real sequence $x[n] = \{3, 0, -2, 4, 6, -9, 5, 9\}$

2.2. The Discrete Fourier Transform (DFT)

The discrete Fourier transform (DFT) of N – point is expressed as follows:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \qquad k = 0,1,\dots,N-1 \tag{9}$$

Where, k is frequency index, and W_N is called twiddle factor

$$W_N = e^{-j2\pi/N} \tag{10}$$

Equation (9) can be computed as

$$X[k] = \sum_{n=0}^{N-1} x[n] \cos\frac{2\pi kn}{N} - j \sum_{n=0}^{N-1} x[n] \sin\frac{2\pi kn}{N} \qquad k = 0,1,\dots,N-1 \tag{11}$$

Let

$$X_R[k] = \sum_{n=0}^{N-1} x[n] \cos\frac{2\pi kn}{N} \qquad k = 0,1,\dots,N-1 \tag{12}$$

$$X_I[k] = -\sum_{n=0}^{N-1} x[n] \sin\frac{2\pi kn}{N} \qquad k = 0,1,\dots,N-1 \tag{13}$$

so

$$X[k] = X_R[k] + j\, X_I[k] \qquad k = 0,1,\dots,N-1 \tag{14}$$

Where, $X_R[k]$ is the real part of $X[k]$ and $X_I[k]$ is the imaginary part of $X[k]$.

Now, if the sequence is real and even $x^e[n]$, then Eq. (13) yields $X_I[k] = 0$, and the DFT reduce to

$$X^e[k] = X_R[k]$$

$$= \sum_{n=0}^{N-1} x^e[n] \cos \frac{2\pi kn}{N} \qquad k = 0,1,\dots,N-1 \qquad (15)$$

Which is real valued and with even symmetry too.

In addition, if the sequence is real and odd $x^o[n]$, then Eq. (12) yields $X_R[k] = 0$. Hence

$$X^o[k] = jX_I[k]$$

$$= -j \sum_{n=0}^{N-1} x^o[n] \sin \frac{2\pi kn}{N} \qquad k = 0,1,\dots,N-1 \qquad (16)$$

Which is odd and purely imaginary.

In general, for real sequence $x[n]$ in Eq. (4) and by using the linear property of the DFT

$$X[k] = X^e[k] + X^o[k]$$

$$= \sum_{n=0}^{N-1} x^e[n] \cos \frac{2\pi kn}{N} - j \sum_{n=0}^{N-1} x^o[n] \sin \frac{2\pi kn}{N}$$
$$k = 0,1,\dots,N-1 \qquad (17)$$

Equation (17) is the core of our proposed implementation, that for computing the DFT of any sequence, we do not need to use any complex multiplications, but only determining the real part of $X[k]$ using the even component of $x[n]$ and the imaginary part of $X[k]$ using the odd component of $x[n]$ separately with real multiplications. And for more reduction in number of multipliers, we will test the twiddle factors W_N^{nk}.

2.3. The Twiddle Factors

The twiddle factors $W_N^{nk} = e^{-j2\pi nk/N}$ have Nth primitive root of unity and its exponent being evaluated modulo N. To compute the DFT in (17) the twiddle factor is separated to real part, which is $a_m = \cos\left(-\frac{2\pi m}{N}\right), m = 0,1,\dots,N-1$, and imaginary part $b_m = \sin\left(-\frac{2\pi m}{N}\right), m = 0,1,\dots,N-1$, as shown in Fig. 3, for $N = 8$.

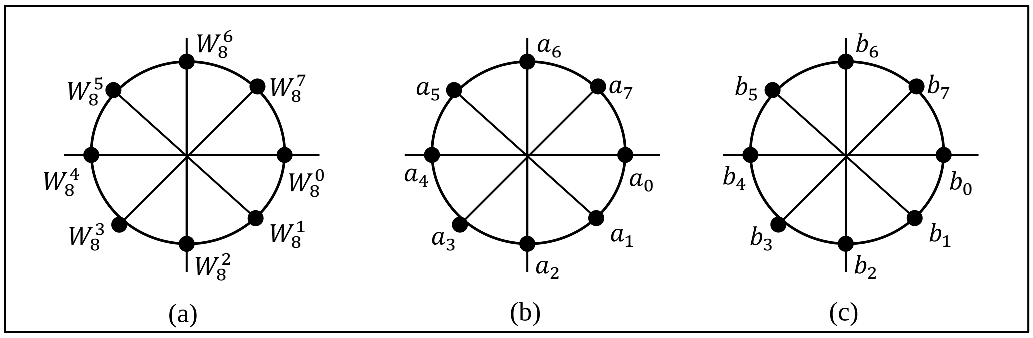


Fig. 3 (a) the twiddle factor for N = 8. (b) the real part, and (c) the imaginary part.

If N is even then the real part of the roots of the twiddle factor (a's) will be symmetric about x-axis and anti-symmetric about y-axis, while the imaginary part (b's) will be anti-symmetric about x-axis and symmetric about y-axis so for 8 – point DFT. The twiddle coefficients can be reduced to only one that is a_1 for real part and b_1 to imaginary part as shown if Fig. 4.

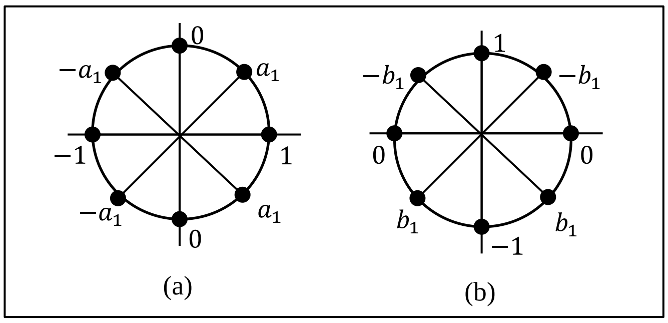


Fig. 4 (a) the real part of twiddle factor for N = 8. (b) its imaginary part

In the same way for N = 16, the number of coefficients a's is 3 and other 3 for b's as shown in Fig. 5.

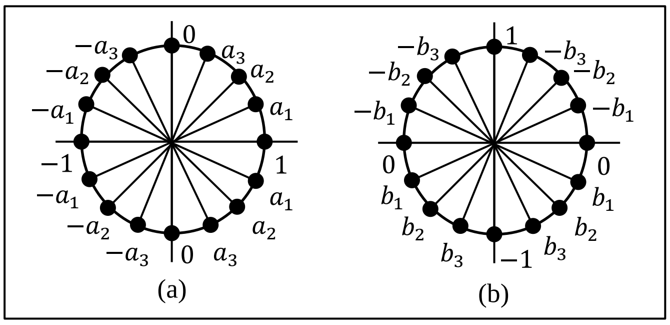


Fig. 5 (a) the real part of twiddle factor for N = 16. (b) its imaginary part.

3. The Proposed Implementation of DFT

In previous sections, we show 4 facts which are used to implement the DFT:

1. Any sequence has even and odd components.
2. The DFT of a real sequence is complex values with real even part and imaginary odd part.
3. The real part of DFT can be determined from the even component of the sequence, and imaginary part of DFT can be determined from the odd component.
4. The number of coefficients, which is needed in the twiddle matrices, is equal to $2\left(\frac{N}{4} - 1\right)$ for N as a power of 2.

3.1 The Proposed Implementation I of DFT

In order to explain our proposed implementation for 8 – point DFT, let rewrite Eqs. (9), (15) and (16) in matrix form as

$$[X] = [W][x] \tag{18}$$

$$[X_R] = [A][x^e] \tag{19}$$

$$[X_I] = [B][x^o] \tag{20}$$

Where [W], [A] and [B] are the twiddle matrix, real matrix of the twiddle, and imaginary matrix of the twiddle matrix, respectively. Thus, [W] for 8 – point is

$$[W] = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W_8^1 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\ 1 & W_8^2 & W_8^4 & W_8^6 & W_8^0 & W_8^2 & W_8^4 & W_8^6 \\ 1 & W_8^3 & W_8^6 & W_8^1 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\ 1 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 & W_8^0 & W_8^4 \\ 1 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8^1 & W_8^6 & W_8^3 \\ 1 & W_8^6 & W_8^4 & W_8^2 & W_8^0 & W_8^6 & W_8^4 & W_8^2 \\ 1 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8^1 \end{bmatrix} \tag{21}$$

To determine $[X_R]$ in Eq. (19), we use Eq. (20) and Fig. 4 to generate the matrix $[A]$

$$\begin{bmatrix} X_R[0] \\ X_R[1] \\ X_R[2] \\ X_R[3] \\ X_R[4] \\ X_R[5] \\ X_R[6] \\ X_R[7] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & a_1 & 0 & -a_1 & -1 & -a_1 & 0 & a_1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & -a_1 & 0 & a_1 & -1 & a_1 & 0 & -a_1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -a_1 & 0 & a_1 & -1 & a_1 & 0 & -a_1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & a_1 & 0 & -a_1 & -1 & -a_1 & 0 & a_1 \end{bmatrix} \begin{bmatrix} x^e[0] \\ x^e[1] \\ x^e[2] \\ x^e[3] \\ x^e[4] \\ x^e[3] \\ x^e[2] \\ x^e[1] \end{bmatrix} \tag{22}$$

Knowing that X_R is even, i.e. $X_R[7] = X_R[1]$, $X_R[6] = X_R[2]$, and $X_R[5] = X_R[3]$, so we need only 5 – point to compute

$$\begin{bmatrix} X_R[0] \\ X_R[1] \\ X_R[2] \\ X_R[3] \\ X_R[4] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & a_1 & 0 & -a_1 & -1 \\ 1 & 0 & -1 & 0 & 1 \\ 1 & -a_1 & 0 & a_1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x^e[0] \\ 2x^e[1] \\ 2x^e[2] \\ 2x^e[3] \\ x^e[4] \end{bmatrix} \tag{23}$$

For determining $[X_I]$, which is odd, i.e. $X_I[0] = 0$, $X_I[4] = 0$, $X_I[7] = -X_I[1]$, $X_I[6] = -X_I[2]$, and $X_I[5] = -X_I[3]$, so we need only 3 – point to compute from Eq. (20)

$$\begin{bmatrix} X_I[0] \\ X_I[1] \\ X_I[2] \\ X_I[3] \\ X_I[4] \\ X_I[5] \\ X_I[6] \\ X_I[7] \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & b_1 & -1 & b_1 & 0 & -b_1 & 1 & -b_1 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & b_1 & 1 & b_1 & 0 & -b_1 & -1 & -b_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -b_1 & -1 & -b_1 & 0 & b_1 & 1 & b_1 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 0 & -b_1 & 1 & -b_1 & 0 & b_1 & -1 & b_1 \end{bmatrix} \begin{bmatrix} 0 \\ x^o[1] \\ x^o[2] \\ x^o[3] \\ 0 \\ -x^o[3] \\ -x^o[2] \\ -x^o[1] \end{bmatrix}$$

$$\begin{bmatrix} X_I[1] \\ X_I[2] \\ X_I[3] \end{bmatrix} = \begin{bmatrix} b_1 & -1 & b_1 \\ -1 & 0 & 1 \\ b_1 & 1 & b_1 \end{bmatrix} \begin{bmatrix} 2x^o[1] \\ 2x^o[2] \\ 2x^o[3] \end{bmatrix} \tag{24}$$

To implement the 8 – point DFT, the Eqs. (23) and (24) be written in form;

$$X_R[0] = (x^e[0] + x^e[4]) + (2x^e[1] + 2x^e[3]) + 2x^e[2]$$

$$X_R[1] = (x^e[0] - x^e[4]) + a_1\,(2x^e[1] - 2x^e[3])$$

$$X_R[2] = (x^e[0] + x^e[4]) - 2x^e[2]$$

$$X_R[3] = (x^e[0] - x^e[4]) - a_1(2x^e[1] - 2x^e[3])$$

$$X_R[4] = (x^e[0] + x^e[4]) - (2x^e[1] + 2x^e[3]) + 2x^e[2]$$

$$X_I[1] = b_1(2x^o[3] + 2x^o[1]) - 2x^o[2]$$

$$X_I[2] = (2x^o[3] - 2x^o[1])$$

$$X_I[3] = b_1(2x^o[3] + 2x^o[1]) + 2x^o[2] \tag{25}$$

Figure 6 shows the complete implementation of 8 – point DFT, which contains three stages, in stage 1, the even and odd components of the input sequence $x[n]$ compute from Eqs. (5) and (6), respectively. The parameter $\frac{1}{2}$ in these two equations is omitted with parameter 2 in Eq. (25) for next stage. In stage 2, the Eq. (25) implements using only two real multipliers, i.e. $a_1(2x^e[1] - 2x^e[3])$ and $b_1(2x^o[3] + 2x^o[1])$. In stage 3, the outputs of stage 2 are sorted and paired to have complex addition $X[k] = X_R[k] + j\,X_I[k]$, and yields the discrete Fourier transform of the 8 – samples input.

Comparing with radix 2 FFT for N = 8 point, the number of complex multipliers $(\frac{N}{2}\log_2 N) = 12$, i.e. 48 real multipliers.

16 – Point DFT

The matrix A in 16 – point DFT can be generated using 3 parameters (a_1, a_2, and a_3) only, also matrix B by 3 parameters (b_1, b_2, and b_3), as shown in Fig. 5. The real part and the imaginary part of DFT in Eqs. (19) and (20) are simplified as:

$$\begin{bmatrix} X_R[0] \\ X_R[1] \\ X_R[2] \\ X_R[3] \\ X_R[4] \\ X_R[5] \\ X_R[6] \\ X_R[7] \\ X_R[8] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & a_1 & a_2 & a_3 & 0 & -a_3 & -a_2 & -a_1 & -1 \\ 1 & a_2 & 0 & -a_2 & -1 & -a_2 & 0 & a_2 & 1 \\ 1 & a_3 & -a_2 & -a_1 & 0 & a_1 & a_2 & -a_3 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ 1 & -a_3 & -a_2 & a_1 & 0 & -a_1 & a_2 & a_3 & -1 \\ 1 & -a_2 & 0 & a_2 & -1 & a_2 & 0 & -a_2 & 1 \\ 1 & -a_1 & a_2 & -a_3 & 0 & a_3 & -a_2 & a_1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x^e[0] \\ 2x^e[1] \\ 2x^e[2] \\ 2x^e[3] \\ 2x^e[4] \\ 2x^e[5] \\ 2x^e[6] \\ 2x^e[7] \\ x^e[8] \end{bmatrix}$$

$$\begin{bmatrix} X_I[1] \\ X_I[2] \\ X_I[3] \\ X_I[4] \\ X_I[5] \\ X_I[6] \\ X_I[7] \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & b_3 & -1 & b_3 & b_2 & b_1 \\ b_2 & -1 & b_2 & 0 & -b_2 & 1 & -b_2 \\ b_3 & b_2 & -b_1 & 1 & -b_1 & b_2 & b_3 \\ -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ b_3 & -b_2 & -b_1 & -1 & -b_1 & -b_2 & b_3 \\ b_2 & 1 & b_2 & 0 & -b_2 & -1 & -b_2 \\ b_1 & -b_2 & b_3 & 1 & b_3 & -b_2 & b_1 \end{bmatrix} \begin{bmatrix} 2x^o[1] \\ 2x^o[2] \\ 2x^o[3] \\ 2x^o[4] \\ 2x^o[5] \\ 2x^o[6] \\ 2x^o[7] \end{bmatrix} \tag{26}$$

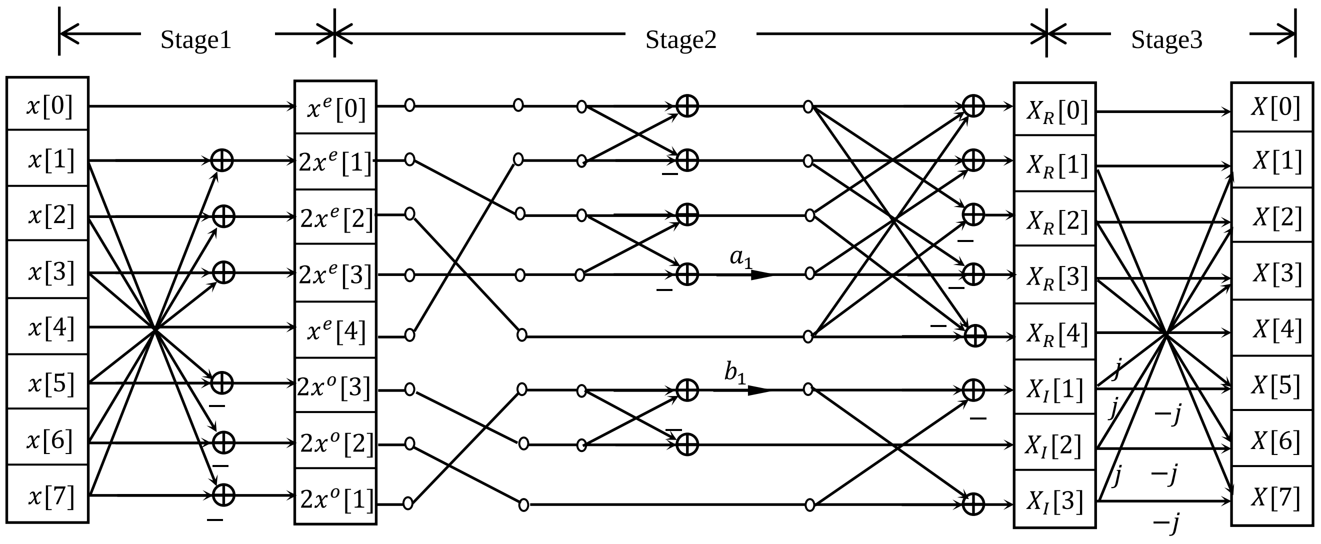

Stage1
Stage2
Stage3
$x[0]$ … $x[7]$
$x^e[0]$, $2x^e[1]$, $2x^e[2]$, $2x^e[3]$, $x^e[4]$, $2x^o[3]$, $2x^o[2]$, $2x^o[1]$
a_1
b_1
$X_R[0]$ … $X_R[4]$, $X_I[1]$, $X_I[2]$, $X_I[3]$
j
$-j$
$X[0]$ … $X[7]$

Fig. 6 the Proposed Implementation I of 8 – Point DFT.

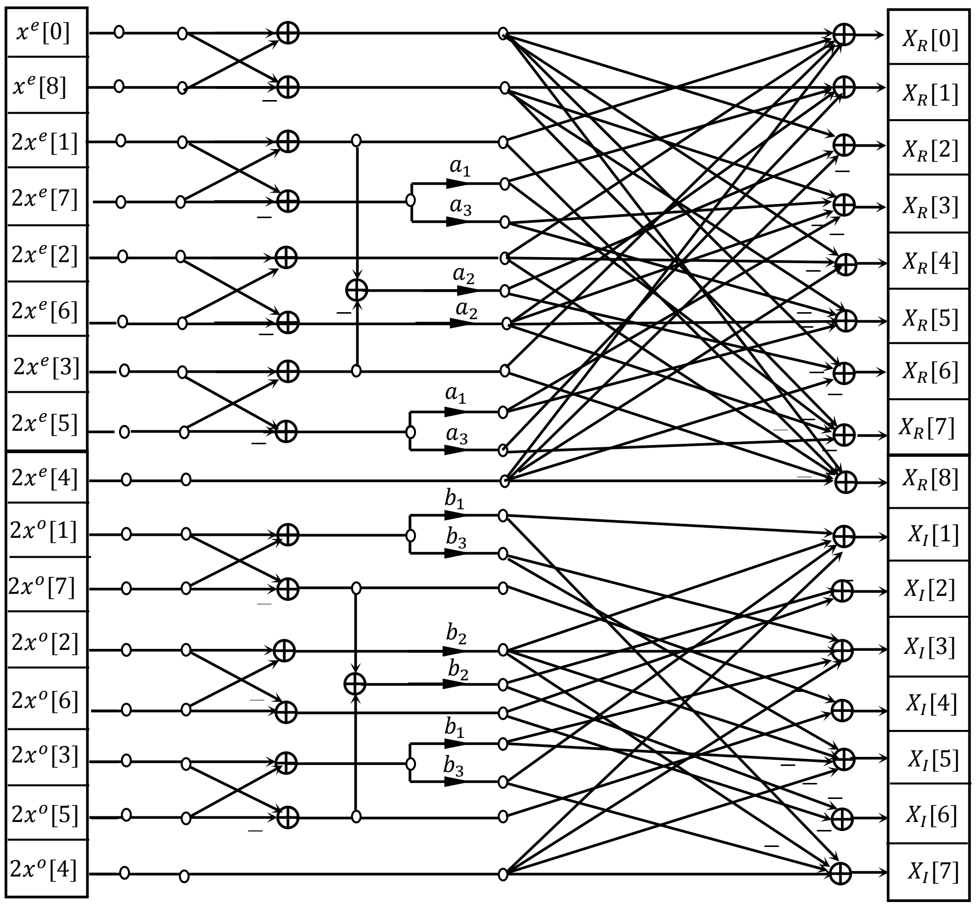

$x^e[0]$, $x^e[8]$, $2x^e[1]$, $2x^e[7]$, $2x^e[2]$, $2x^e[6]$, $2x^e[3]$, $2x^e[5]$, $2x^e[4]$
$2x^o[1]$, $2x^o[7]$, $2x^o[2]$, $2x^o[6]$, $2x^o[3]$, $2x^o[5]$, $2x^o[4]$
a_1
a_2
a_3
b_1
b_2
b_3
$X_R[0]$ … $X_R[8]$
$X_I[1]$ … $X_I[7]$

Fig. 7 the Proposed Implementation of 16 – Point DFT.

Figure 7 shows the implementation of 16 – point DFT (only at stage2); Equation (26) implements using only 12 real multipliers.

In general for N – point DFT, the real part of $X[k]$, $k = 0,1,\ldots N-1$ can be determined using $\frac{N}{2}+1$ points, that is

$$X_R[k] = x^e[0] + (-1)^k x^e\left[\frac{N}{2}\right] + \sum_{n=1}^{\frac{N}{2}-1} x^e[n]\cos\frac{2\pi kn}{N} \qquad k = 0,1,\ldots,\frac{N}{2} \tag{27}$$

$$X_R[k] = x^e[0] + x^e\left[\frac{N}{2}\right] + 2\sum_{n=1}^{\frac{N}{4}-1} a_{\langle nk\rangle_N}\left(x^e[n] + x^e\left[\frac{N}{2}-n\right]\right) + (-1)^{\frac{k}{2}} 2x^e\left[\frac{N}{4}\right] \qquad \text{for } k \text{ even} \tag{28}$$

$$X_R[k] = x^e[0] - x^e\left[\frac{N}{2}\right] + 2\sum_{n=1}^{\frac{N}{4}-1} a_{\langle nk\rangle_N}\left(x^e[n] - x^e\left[\frac{N}{2}-n\right]\right) \qquad \text{for } k \text{ odd} \tag{29}$$

Where, $a_{\langle nk\rangle_N} = \cos\frac{2\pi kn}{N}$, which are reduced to only $(\frac{N}{4}-1)$ coefficients, as we explain in previous section, i.e. $a_1, a_2, \dots,$ and $a_{\frac{N}{4}-1}$. Note that $a_0 = 1$ and $a_{\frac{N}{4}} = -1$.

To implement the real part of DFT, we need $\left(\frac{N}{8}\left(\frac{N}{4}-2\right)+\log_2 N - 2\right)$ real multiplications.

The imaginary part of DFT can be determined using $\frac{N}{2}-1$ points, that is

$$X_I[k] = \sum_{n=1}^{\frac{N}{2}-1} x^o[n]\sin\frac{2\pi kn}{N}\,, \qquad k = 1,2,\dots,\frac{N}{2}-1 \tag{30}$$

$$X_I[k] = 2\sum_{n=1}^{\frac{N}{4}-1} b_{\langle nk\rangle_N}\left(x^o[n] - x^o\left[\frac{N}{2}-n\right]\right), \quad \text{for } k \text{ even} \tag{31}$$

$$X_I[k] = 2\sum_{n=1}^{\frac{N}{4}-1} b_{\langle nk\rangle_N}\left(x^o[n] + x^o\left[\frac{N}{2}-n\right]\right) + (-1)^{\frac{k+1}{2}}2x^o\left[\frac{N}{4}\right], \quad \text{for } k \text{ odd} \tag{32}$$

Where, $b_{\langle nk\rangle_N} = \sin\frac{2\pi kn}{N}$, which are reduced to only $(\frac{N}{4}-1)$ coefficients, i.e. $b_1,\ b_2, \dots,$ and $b_{\frac{N}{4}-1}$.

Also, to implement the imaginary part of DFT, we need $\left(\frac{N}{8}\left(\frac{N}{4}-2\right)+\log_2 N-2\right)$ real multiplications, that means, to build the proposed implementation of DFT, the number of real multiplications is equal to $2\left(\frac{N}{8}\left(\frac{N}{4}-2\right)+\log_2 N-2\right)$.

3.2 The Proposed Implementation II of DFT

For more reduction in number of real multipliers, we share multipliers of real part DFT (a's) to compute the imaginary part of DFT, which also will reduce an area of implementation for efficient processors using VLSI or FPGA.

If N is a multiple of 4, the twiddle factor W_N^1 and $W_N^{\frac{N}{4}-1}$ are distributed as shown in Fig. 8.

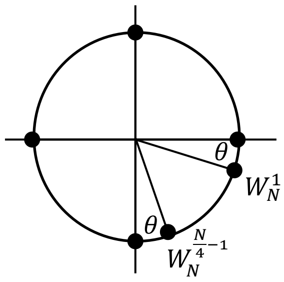


Fig. 8 The twiddle factor for N a multiple of 4

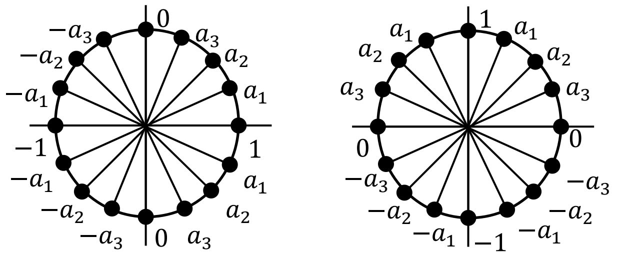
$$\sin(\frac{2\pi n}{N}) = -\cos\left(\frac{\pi}{2}-\frac{2\pi n}{N}\right) = -\cos\left(\frac{2\pi\left(\frac{N}{4}-n\right)}{N}\right), \quad n = 1,2,\dots,\frac{N}{4}-1 \tag{33}$$

$$b_1 = \sin(\frac{2\pi}{N}) = -\cos\left(\frac{2\pi\left(\frac{N}{4}-1\right)}{N}\right) = -a_{\frac{N}{4}-1}\,,$$

$$b_2 = -a_{\frac{N}{4}-2}, \quad \text{and so on,} \quad b_{\frac{N}{4}-2} = -a_2\,,$$

$$b_{\frac{N}{4}-1} = -a_1\,, \tag{34}$$

That for $N = 16$, $b_3 = -a_1$, $b_2 = -a_2$ and $b_1 = -a_3$, and the imaginary part of the twiddle factors are distributed as shown in Fig. 9 (b).



(a) (b)

Fig. 9 (a) the real part of twiddle factor for N = 16. (b) Its imaginary part with same coefficients of real part.

Equations (31) and (32) can be written as:

$$X_I[k] = -2\sum_{n=1}^{\frac{N}{4}-1} a_{\langle nk\rangle_N}\left(x^o\left[\frac{N}{4}-n\right] - x^o\left[\frac{N}{2}-(\frac{N}{4}-n)\right]\right) \quad \text{for } k \text{ even} \tag{35}$$

$$X_I[k] = -2\sum_{n=1}^{\frac{N}{4}-1} a_{\langle nk\rangle_N}\left(x^o\left[\frac{N}{4}-n\right] + x^o\left[\frac{N}{2}-(\frac{N}{4}-n)\right]\right) + (-1)^{\frac{k+1}{2}}2x^o\left[\frac{N}{4}\right] \quad \text{for } k \text{ odd} \tag{36}$$

To compute X_I [k] from Equations (28) and (29), and comparing them with Equations (35) and (36), the procedure is as follow:

- Let $x^o[0] = x^o\left[\frac{N}{2}\right] = 0$.
- The sequence from $x^o\left[\frac{N}{4}+1\right]$ to $x^o\left[\frac{N}{2}-1\right]$ enter to stage 2 of implementation as a negative value.
- The odd component sequence $x^o[n]$ is sorted from $x^o\left[\frac{N}{4}-1\right]$ to $x^o[1]$ with their pairs.
- The difference among the equations is the midterm ($x^o\left[\frac{N}{4}\right]$ or $x^e\left[\frac{N}{4}\right]$), which is added in (28) when k is even, while it appears in (35) when k is odd. This problem is solved by using de-multiplexer to select where must be added as shown in Fig. 10.

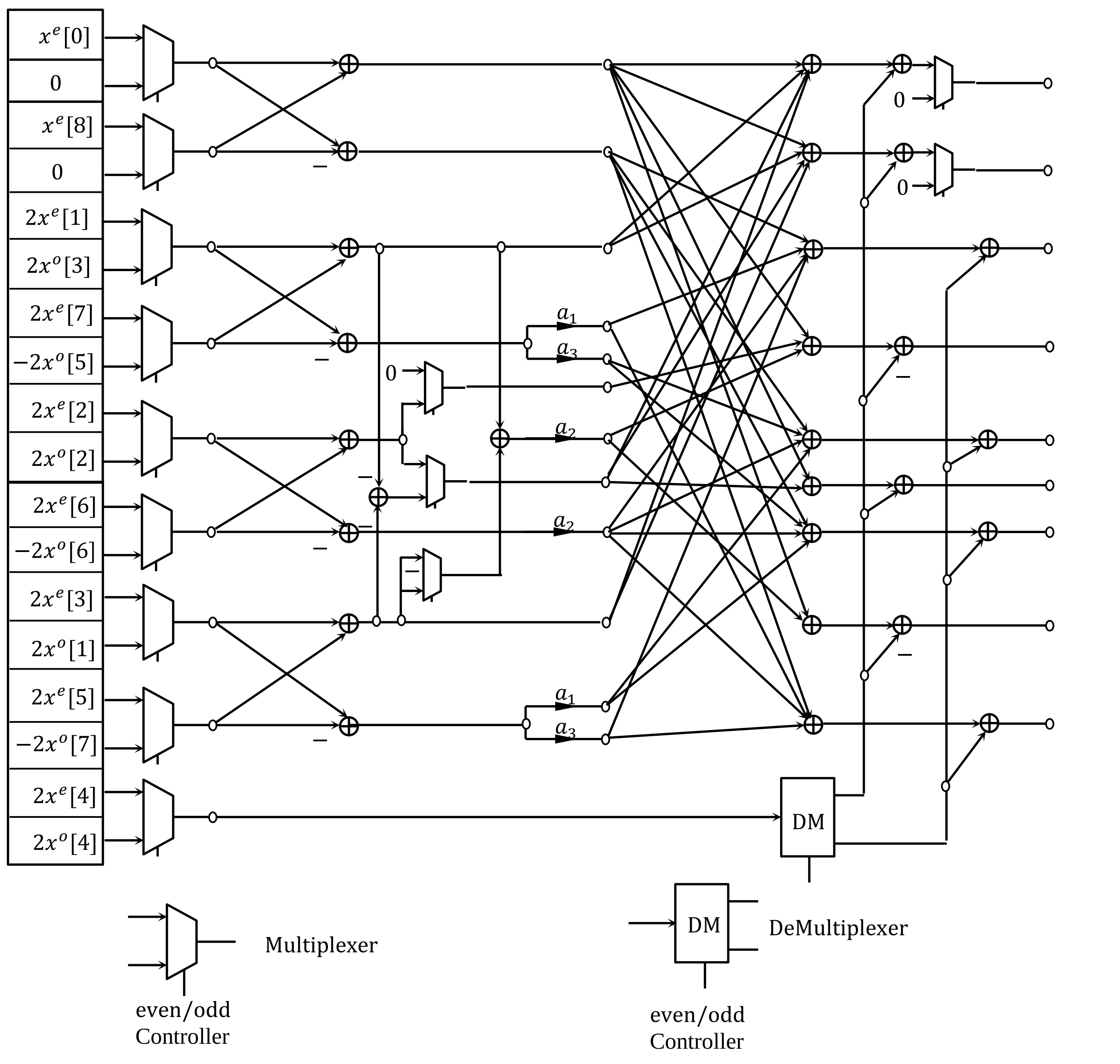


Fig. 10 The proposed implementation II for DFT.

The output of the system in Fig. 10 is either $X_R[k], k = 0,1, \dots \frac{N}{2}$, if its input $x^e[n], n = 0,1, \dots, \frac{N}{2}$, or $-X_I[k], k = 0,1, \dots \frac{N}{2}$, if its input $x^o[n], n = 0,1, \dots, \frac{N}{2}$. The selection of the inputs is done by controlling the multiplexers and de-multiplexer. To implement the second proposed implementation of the DFT, we need only $\left(\frac{N}{8}\left(\frac{N}{4} - 2\right) + \log_2 N - 2\right)$ real multiplications.

4. The Implementation of IDFT

The inverse DFT is given by Proakis and Manolakis [6]:

$$x[n] = \frac{1}{N}(DFT(X^*[k]))^* \quad (37)$$

That means, we can compute the IDFT, by using the same proposed implementation of DFT with input sequence complex conjugate of $X[k]$, which is already containing real part even component $X_R[k]$ and odd component $X_I[k]$, so the first stage in Fig. 6 is not needed in computing IDFT. The stage 3 will be changed from complex additions to real additions, while stage 2 remains as it is.

5. Results and Discussion

The proposed implementation I of DFT can be used for any N even, and proposed II is used for N a power of 4, without any zero padding, which is needed in FFT. The proposed design is sufficient for N ≤ 1024. As compared to FFT algorithm, the number of real multiplication is significantly reduced as shown in Table 1.

Table 1 Number of Real Multiplications to Compute an N-point DFT.

N	Complex Multiplications FFT	Real Multiplications FFT	Real Multiplications Proposed I	Real Multiplications Proposed II
8	12	48	2	1
16	32	128	12	6
32	80	320	54	27
64	192	768	232	116
128	448	1792	970	485
256	1024	4096	3980	1990
512	2304	9216	16124	8071
1024	5120	20480	65040	32520

6. Conclusions

This paper proposed two implementation to reduce the number of real multipliers, which are needed to compute the DFT. The new designs are sufficient for $N \leq 512$ or $N \leq 1024$ comparing with FFT algorithm as shown in Table 1. Our idea in this paper is implemented directly from real and imaginary part equations of $X\ [k]$ without processing to have cascaded, tree, or radix structure, which may be reduced further.

References

[1] R. Preyadharan, A. Tamilselvan, and M. Nithiyaa, "Modified Architecture of FFT Module using CSD Multiplier and Dual Edge Triggered Flip Flop", IEEE, 2015 International Conference on Innovations in Information, Embedded and Communication systems (ICIIECS), Coimbatore, pp. 1-4, 2015.

[2] R. Anitha, and V. Bagyaveereswaran, "Bruan's Multiplier Implementation using FPGA with Bypassing Techniques", International Journal of VLSI design and Communication Systems (VLSICS), Vol. 2, No. 3, September 2011.

[3] Mingyu Wang, Fang Wang, Shaojun Wei, and Zhaolin Li, "A pipelined area-efficient and high-speed reconfigurable processor for floating-point FFT/IFFT and DCT/IDCT computations", Elsevier, Microelectronics Journal, Vol. 47, pp. 16-30, 2016.

[4] Xing Wei, Haigang Yang, Wei Li, Zhihong Huang, Tao Yina, and Le Yub, "A reconfigurable 4-GS/s power-efficient floating-point FFT processor design and implementation based on single-sided binary-tree decomposition", Elsevier, Integration, Vol. 66, pp. 164-172, 2019.

[5] K. Sivanandam, and P. Kumar, "Design and performance analysis of reconfigurable modified Vedic multiplier with 3-1-1-2 compressor", Microprocessors and Microsystems, Vol. 65, pp. 97-106, 2019.

[6] John G. Proakis, and Dimitris G. Manolakis, "Digital Signal Processing, Principles, Algorithms, and Applications", Fourth Edition, Prentice-Hall, Inc., ISBN-13: 978-0131873742, 2007.

Biographies

Majid Abdulnabi Alwan received the B. Sc. degree in Electrical Engineering from Department of Electrical Engineering, College of Engineering, University of Basrah, Basrah, Iraq in 1981. He received the M. Sc. degree in "Control and Computer Engineering" from University of Basrah, Iraq in 1990. He also received the Ph. D. degree in "Electrical and Electronic Engineering" from University of Basrah, Iraq in 2002. He worked as a teaching assistant and an instructor in the Department of Electrical Engineering, College of Engineering, University of Basrah, Basrah, Iraq from 1985 to 1987. He rejoined the same department as an Assistant lecturer in 1990, then as a lecturer. He had held several times the post of Head of the Department of Computer Engineering, and Dean Assistant, at the College of Engineering, University of Basrah.