International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 11 (2017) pp. 2692-2702

An Enhanced Arbitrary Radix FFT Algorithm

Fairouz Belilita[1], Nourredine Amardjia[2, 3] and Djamel Chikouche[1, 3]

[1]Department of Electronics, Faculty of Technology, University of M'sila, Algeria.
[2]Department of Electronics, Faculty of Technology, University of Setif-1, Algeria.
[3] LIS Laboratory, University of Setif-1, Algeria.

~

[2]ORCID: 0000-0002-7177-4260

Abstract:

In this paper, we propose an enhanced radix-*R* decimation-in-frequency (DIF) FFT algorithm based on the Cooley–Tukey approach coupled with a new indexing process introduced in some of the output sub-sequences resulting from the conventional decomposition of the DFT, where *R* is an arbitrary integer greater than two. The proposed algorithm significantly reduces the number of twiddle factors computations or loads from the lookup table compared to the conventional Cooley–Tukey radix-*R* DIF FFT algorithm. The gain is about 50%, which is achieved without any extra computational or structural complexity.

Keywords: Discrete Fourier Transform; FFT indexing process; Radix-*R* decimation-in-frequency FFT; Twiddle factors computations.

INTRODUCTION

Among all the discrete transforms, the discrete Fourier transform (DFT), which maps a sequence from the time domain to the frequency domain and vice versa, is the most widely utilized transform in the area of digital signal processing and related fields. Basic applications such as filtering, convolution/correlation, spectrum analysis, and more advanced ones like communications (e.g. OFDM), speech signal-processing, radar, sonar, imaging, are just a subset of the wide range of its uses. This tremendous popularity of the DFT is partly due to the fact that it can be computed efficiently using a fast Fourier transform (FFT) algorithm. The development of the Cooley–Tukey FFT algorithm (CTA) [1] has been followed by various enhancements carried out by other subsequent researchers leading, for instance, to algorithms such as the prime factor [2, 3], vector radix [2, 4], split radix [5], split vector radix [6] and Winograd Fourier transform algorithm (WFTA) [7]. These efforts were mainly devoted to design faster algorithms suitable for some available platforms that are based on general purpose processors, DSP chips or FFT dedicated processors. However, it must be highlighted that because of their simplicity and very high regularity, the algorithms resulted from the well-known Cooley-Tukey approach remain the most popular for the reason that they lead to efficient software and hardware implementations. In addition, with the remarkable advance in VLSI technology, implementing high-radix FFT algorithms on small silicon area is becoming feasible [8-11]. High-radix Cooley-Tukey FFT algorithms are desirable for the reason that they noticeably reduce the number of arithmetic operations and data transfers when compared to the radix-2 FFT algorithm. These reductions have a direct impact on decreasing the execution time and power consumption, which are very important and sought-after properties on dedicated portable or embedded FFT processors.

Furthermore, twiddle factors represent an important part in an FFT computation. They can be generated at run-time by using the CORDIC algorithm [12, 13], a polynomial-based approach [14] or a function generator based on the recursive feedback difference equation for the computation of sine and cosine functions [15]. They can also be pre-computed once and stored in a memory array and then accessed by a table lookup. In the radix-2 *N*-point FFT, there are *N*/2 different twiddle factors and a simple way to produce the indices of the appropriate twiddle factor for each butterfly operation is given in [16]. A technique requiring only *N*/4 coefficients to be stored into a lookup table is reported in [17]. Besides, due to the periodicity of trigonometric functions, a coefficient manipulation method presented in [18] needs only (*N*/8+1) coefficients to generate all the twiddle factors. These methods permit to shrink the twiddle factors lookup table memory space but they are designed only for the sole radix-2 algorithm. Techniques based on high radices and focussing on achieving improvements in terms of twiddle factor generations and accesses to the lookup table have also been proposed in [19-

22]. Still, they are only appropriate for some few particular radices.

In this paper, we propose an improved general radix-*R* decimation-in-frequency (DIF) FFT algorithm based on the approach reported in [19] and [20]. The proposed algorithm, which is valid for any radix *R* greater than two, focuses mainly on significantly reducing the number of twiddle factors computations or loads from the lookup table. This is attained by a slight difference in the indexing process of some of the output sub-sequences resulting from the usual Cooley-Tukey decomposition of the DFT. We must emphasize that this procedure does not increase any of the computational or structural complexities of the conventional algorithm. If used in hardware implementations such as the architectures presented in [10] or [23], the proposed algorithm would have an important impact on reducing the number of the twiddle factors memory modules. It should be noted that [19] and [20] discuss only the case where the radix *R* is an integral power of two, specifically the cases *R* =4, *R* =8 and *R* =16. However, it is highly desirable to consider other possible values of the radix *R* since for many FFT-based applications the length of the input sequence to be transformed by the FFT is not necessarily a power of two. Therefore, the use of the algorithms [19] and [20] in such applications necessitates a zero-padding procedure that actually consumes a larger amount of memory on FFT-based processors, and hence is not desirable since minimizing the required memory size is an efficient way for the silicon area reduction. Fortunately, the proposed algorithm, which overcomes the limitations of the algorithms in [19] and [20], is valid for any integer *R* greater than two. For instance, this property of the proposed algorithm is of great importance for FFT-based digital filter design, where the length of the impulse response is generally not a power of two.

The remainder of this paper is organized as follows. The conventional Cooley–Tukey radix-*R* DIF-FFT algorithm is thoroughly reviewed in section 2 as it is the groundwork for the proposed algorithm presented in section 3. Finally, section 4 gives a conclusion.

CONVENTIONAL COOLEY–TUKEY RADIX-*R* DIF-FFT ALGORITHM

The DFT of a data sequence $x(k)$ of size N is given by

$$X(n)=\sum_{k=0}^{N-1}x(k)W_N^{nk},\quad 0\le n\le N-1 \tag{1}$$

where $W_N=\exp(-2j\pi/N)$ and $j=\sqrt{-1}$.

If N is assumed to be an integral power of R, i.e. $N=R^r$, then we can use the radix-R DIF index map that consists of changing the input index k in (1) by

$$k+q\frac{N}{R},\quad 0\le k\le\frac{N}{R}-1,\quad 0\le q\le R-1 \tag{2}$$

The DFT given in (1) can then be expressed as

$$X(n)=\sum_{k=0}^{(N/R)-1}\left[x(k)+x\left(k+\frac{N}{R}\right)W_R^{n}+x\left(k+2\frac{N}{R}\right)W_R^{2n}+\cdots\right.$$
$$\left.\cdots+x\left(k+(R-2)\frac{N}{R}\right)W_R^{(R-2)n}+x\left(k+(R-1)\frac{N}{R}\right)W_R^{(R-1)n}\right]W_N^{nk}$$
$$n:0,1,\cdots,N-1 \tag{3}$$

As done for the input index k, we can further use the radix-R DIF index map to change the output index n in (3) by

$$Rn+p,\qquad 0\le n\le\frac{N}{R}-1,\qquad 0\le p\le R-1 \tag{4}$$

Using (4) and the fact that $W_N^{Rnk} = W_{N/R}^{nk}$, equation (3) can be rearranged as

$$X(Rn+p) = \sum_{k=0}^{(N/R)-1} \left\{ \left[x(k) + x\left(k+\frac{N}{R}\right)W_R^{p} + x\left(k+2\frac{N}{R}\right)W_R^{2p} + \cdots \right.\right.$$
$$\left.\left. \cdots + x\left(k+(R-2)\frac{N}{R}\right)W_R^{(R-2)p} + x\left(k+(R-1)\frac{N}{R}\right)W_R^{(R-1)p} \right] W_N^{pk} \right\} W_{N/R}^{nk}$$
$$0 \le n \le \frac{N}{R}-1, \qquad 0 \le p \le R-1 \tag{5}$$

A compact form of (5) can be obtained as

$$X(Rn+p) = \sum_{k=0}^{(N/R)-1} \left\{ g_p(k) W_N^{pk} \right\} W_{N/R}^{nk} \qquad 0 \le n \le \frac{N}{R}-1, \quad 0 \le p \le R-1 \tag{6}$$

where

$$g_p(k) = \sum_{q=0}^{R-1} x(k+q\frac{N}{R}) W_R^{pq}, \qquad 0 \le p \le R-1 \tag{7}$$

The matrix form of (7) is given by

$$\begin{bmatrix} g_0(k) \\ g_1(k) \\ g_2(k) \\ \vdots \\ g_{R-1}(k) \end{bmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_R^{1} & W_R^{2} & \cdots & W_R^{R-1} \\ 1 & W_R^{2} & W_R^{4} & \cdots & W_R^{2(R-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_R^{R-1} & W_R^{2(R-1)} & \cdots & W_R^{(R-1)(R-1)} \end{pmatrix} \begin{bmatrix} x(k) \\ x(k+N/R) \\ x(k+2N/R) \\ \vdots \\ x(k+(R-1)N/R) \end{bmatrix} \tag{8}$$

Equation (8) can be written in a reduced form as

$$\mathbf{G}_k = \mathbf{W}_R \mathbf{X}_k \tag{9}$$

where, as seen in (8), $\mathbf{W}_R$ is the operator matrix of the length-R DFT. The components of the input and output vectors $\mathbf{X}_k$ and $\mathbf{G}_k$ of (9) are related to the input and output sequences of (7) by $X_k(q) = x(k+q\frac{N}{B})$ and $G_k(p) = g_p(k)$, respectively.

Now, by setting

$$y_p(k) = g_p(k) W_N^{pk}, 0 \le p \le R-1 \tag{10}$$

and using (8), we get

$$\begin{bmatrix} y_0(k) \\ y_1(k) \\ y_2(k) \\ \vdots \\ y_{R-1}(k) \end{bmatrix} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & W_N^{k} & 0 & \cdots & 0 \\ 0 & 0 & W_N^{2k} & 0 & \vdots \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 & W_N^{(R-1)k} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_R^{1} & W_R^{2} & \cdots & W_R^{R-1} \\ 1 & W_R^{2} & W_R^{4} & \cdots & W_R^{2(R-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_R^{R-1} & W_R^{2(R-1)} & \cdots & W_R^{(R-1)(R-1)} \end{pmatrix} \begin{bmatrix} x(k) \\ x(k+N/R) \\ x(k+2N/R) \\ \vdots \\ x(k+(R-1)N/R) \end{bmatrix} \tag{11}$$

Equation (11) represents the matrix form of the butterfly of the conventional radix-*R* FFT algorithm based on the Cooley-Tukey approach. It can also be expressed in a compact form as

$$\mathbf{Y}_k = \mathbf{T}_k^{\text{conv}} \mathbf{W}_R \mathbf{X}_k \tag{12}$$

where the components of the output vector $\mathbf{Y}_k$ are related to the output sequence of (10) by

$$Y_k(p) = y_p(k)$$

$\mathbf{T}_k^{\text{conv}}$ is the twiddle factors diagonal matrix shown in (11), which can be expressed as

$$\mathbf{T}_k^{\text{conv}} = diag(1, W_N^k, W_N^{2k}, \cdots, W_N^{(R-2)k}, W_N^{(R-1)k}) \tag{13}$$

Finally, (5) can be rewritten as

$$X(Rn+p) = \sum_{k=0}^{(N/R)-1} y_p(k) W_{N/R}^{nk} \qquad 0 \le n \le \frac{N}{R} - 1, \quad 0 \le p \le R-1 \tag{14}$$

which is the familiar expression of an *N*/*R*-point DFT.

As a result, we notice from (5), (6) or (14) that the DFT given in (1) has been decomposed in the first stage into *R* DFTs of length (*N*/*R*), one for each value of *p* $(0 \le p \le R-1)$. The input sequence of any of these DFTs is $y_p(k) = g_p(k) W_N^{pk}$, with $g_p(k)$ given in (7). These *R* (*N*/*R*)-point DFTs together make up the initial *N*-point DFT. The decomposition process is repeated in the second stage by dividing each of these (*N*/*R*)-point DFTs into *R* (*N*/*R*^{2})-point DFTs. Each (*N*/*R*^{2}) DFT is further divided in the next stage into *R* (*N*/*R*^{3})-point DFTs, and so on, until the final decimation produces *R*-point DFTs. The entire DFT computation requires $r = log_R(N)$ stages.

PROPOSED ALGORITHM

The proposed algorithm is based on the Cooley-Tukey approach presented in the above section, but with a small difference in the indexing process of almost half of the sub DFTs. This will reduce, in the same proportion, the number of twiddle factors computations or loads from the lookup table. Two different cases can arise depending on the parity of *R*. These two cases must be studied separately because of their particularities.

Case of an odd radix R :

The first $\frac{R+1}{2}$ sub DFTs, i.e. $0 \le p \le \frac{R-1}{2}$, are computed according to (6) or (14), and rewritten here with new limits of *p*.

$$X(Rn+p) = \sum_{k=0}^{(N/R)-1} \left\{ g_p(k) W_N^{pk} \right\} W_{N/R}^{nk} = \sum_{k=0}^{(N/R)-1} y_p(k) W_{N/R}^{nk}$$

$$0 \le n \le \frac{N}{R} - 1, \quad p = 0, 1, \cdots, \frac{R-1}{2} \tag{15}$$

For the remaining sub DFTs, i.e. $\frac{R+1}{2} \le p \le R-1$, a new indexing, which takes advantage of the periodicity propriety of the twiddle factors while generating the same sub DFTs, is introduced as follows

$$X((N+Rn-p) \bmod N) = \sum_{k=0}^{(N/R)-1} \left\{ g_{-p}(k) W_N^{-pk} \right\} W_{N/R}^{nk} = \sum_{k=0}^{(N/R)-1} y_{-p}(k) W_{N/R}^{nk}$$

$$0 \le n \le \frac{N}{R} - 1 \qquad p = \frac{R-1}{2}, \frac{R-1}{2} - 1, \frac{R-1}{2} - 2, \cdots, 1 \tag{16}$$

It should be mentioned that the notation of the index *p* is kept the same in order to avoid introducing a new one.

From (15) and (16), we notice that a new twiddle factors matrix $\mathbf{T}_k^{\text{new-odd}}$, specified by

$$\mathbf{T}_k^{\text{new-odd}} = diag(1, W_N^k, W_N^{2k}, \cdots, W_N^{\frac{R-1}{2}k}, W_N^{-\frac{R-1}{2}k}, \cdots, W_N^{-2k}, W_N^{-k}) \tag{17}$$

has replaced the conventional one given in (13) and therefore, the butterfly of the conventional radix-*R* FFT given in (12) becomes

$$\mathbf{Y}_k = \mathbf{T}_k^{\text{new-odd}} \mathbf{W}_R \mathbf{X}_k \tag{18}$$

Now, using the facts that $W_N^{pk} = \cos\left(\frac{2\pi}{N} pk\right) - j \sin\left(\frac{2\pi}{N} pk\right)$ and $W_N^{-pk} = \cos\left(\frac{2\pi}{N} pk\right) + j \sin\left(\frac{2\pi}{N} pk\right)$ in the elements of $\mathbf{T}_k^{\text{new-odd}}$ given by (17), only $(R-1)$ real twiddle factors need to be computed or loaded from the lookup table during the processing of the butterfly specified by (18) of the proposed algorithm instead of the $(2(R-1))$ ones needed in the conventional butterfly given by (12). As a result, a saving of exactly 50% is realized using the proposed algorithm. Note that when the lookup table is used, similar savings are obtained in the address generations.

It should be noted that the proposed new indexing, i.e. $(N+Rn-p) \bmod N$, does not add any computational complexity compared to the conventional one, i.e. $(Rn+p)$, because

$$(N+Rn-p) \bmod N = \begin{cases} N-p, & \text{for } n=0 \\ Rn-p, & \text{elsewhere} \end{cases} \qquad 1 \le p \le \frac{R-1}{2} \tag{19}$$

For illustration considerations, we show in the following two examples the proposed radix-*R* FFT computations involved in the first stage of an *N*-point DFT ($N=R^r$).

Example 1: The proposed radix-3 FFT algorithm for an *N*-point DFT, where $N=3^r$

$$X(3n) = \sum_{k=0}^{(N/3)-1} y_0(k) W_{N/3}^{nk} \quad 0 \le n \le \frac{N}{3} - 1$$

$$X(3n+1)=\sum_{k=0}^{(N/3)-1} y_1(k)W_{N/3}^{nk} \quad 0\le n\le \frac{N}{3}-1$$

$$X((N+3n-1)\bmod N)=\sum_{k=0}^{(N/3)-1} y_{-1}(k)W_{N/3}^{nk} \quad 0\le n\le \frac{N}{3}-1$$

where

$$\begin{bmatrix} y_0(k) \\ y_1(k) \\ y_{-1}(k) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & W_N^k & 0 \\ 0 & 0 & W_N^{-k} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 1 & W_3^1 & W_3^2 \\ 1 & W_3^2 & W_3^1 \end{pmatrix} \begin{bmatrix} x(k) \\ x(k+N/3) \\ x(k+2N/3) \end{bmatrix}$$

Example 2: The proposed radix-5 FFT algorithm for an N-point DFT, where $N=5^r$

$$X(5n)=\sum_{k=0}^{(N/5)-1} y_0(k)W_{N/5}^{nk} \quad 0\le n\le \frac{N}{5}-1$$

$$X(5n+1)=\sum_{k=0}^{(N/5)-1} y_1(k)W_{N/5}^{nk} \quad 0\le n\le \frac{N}{5}-1$$

$$X(5n+2)=\sum_{k=0}^{(N/5)-1} y_2(k)W_{N/5}^{nk} \quad 0\le n\le \frac{N}{5}-1$$

$$X((N+5n-2)\bmod N)=\sum_{k=0}^{(N/5)-1} y_{-2}(k)W_{N/5}^{nk} \quad 0\le n\le \frac{N}{5}-1$$

$$X((N+5n-1)\bmod N)=\sum_{k=0}^{(N/5)-1} y_{-1}(k)W_{N/5}^{nk} \quad 0\le n\le \frac{N}{5}-1$$

where

$$\begin{bmatrix} y_0(k) \\ y_1(k) \\ y_2(k) \\ y_{-2}(k) \\ y_{-1}(k) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & W_N^k & 0 & 0 & 0 \\ 0 & 0 & W_N^{2k} & 0 & 0 \\ 0 & 0 & 0 & W_N^{-2k} & 0 \\ 0 & 0 & 0 & 0 & W_N^{-k} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W_5^1 & W_5^2 & W_5^3 & W_5^4 \\ 1 & W_5^2 & W_5^4 & W_5^1 & W_5^3 \\ 1 & W_5^3 & W_5^1 & W_5^4 & W_5^2 \\ 1 & W_5^4 & W_5^3 & W_5^2 & W_5^1 \end{pmatrix} \begin{bmatrix} x(k) \\ x(k+N/5) \\ x(k+2N/5) \\ x(k+3N/5) \\ x(k+4N/5) \end{bmatrix}$$

In order to see the exact amount of the reductions realized in terms of real twiddle factors computations or loads from the lookup table by the proposed algorithm compared to the CTA, we show in table 1 various N-size DFTs processed by the two algorithms with different odd radices (R=3, R=5 and R=7). Radices 3 and 5 have been frequently used in the design of FFT kernels or FFT processors cores [24 - 26]. For large sizes N, the savings are significant. The memory size of the lookup table is reduced to half in FFT dedicated processors.

Table 1: Number of real twiddle factors computations or loads from the lookup table in the CTA and the proposed algorithm (case of odd radices).

	Radix-*R* CTA			Proposed radix-*R* algorithm		
N \ *R*	3	5	7	3	5	7
9	8			4		
25		32			16	
27	56			28		
49			72			36
81	272			136		
125		352			176	
243	1136			568		
343			1080			540
625		2752			1376	
2401			11664			5832
3125		18752			9376	
16807			110448			55224

Case of an even radix R :

For an even radix, the procedure is the same as for the case of odd radix illustrated in the previous subsection, except for the limits of *p*. The first $(\frac{R}{2}+1)$ sub DFTs corresponding to $0 \le p \le \frac{R}{2}$ are computed according to (6) or (14), and rewritten here with new limits of *p*.

$$X(Rn+p) = \sum_{k=0}^{(N/R)-1} \left\{ g_p(k) W_N^{pk} \right\} W_{N/R}^{nk} = \sum_{k=0}^{(N/R)-1} y_p(k) W_{N/R}^{nk}$$

$$0 \le n \le \frac{N}{R}-1, \quad p = 0,1,\cdots,\frac{R}{2} \tag{20}$$

The remaining sub DFTs (i.e. $\frac{R}{2}+1 \le p \le R-1$) are calculated with a new indexing as follows

$$X((N+Rn-p) \bmod N) = \sum_{k=0}^{(N/R)-1} \left\{ g_{-p}(k) W_N^{-pk} \right\} W_{N/R}^{nk} = \sum_{k=0}^{(N/R)-1} y_{-p}(k) W_{N/R}^{nk}$$

$$0 \le n \le \frac{N}{R}-1 \qquad p = \frac{R}{2}-1, \frac{R}{2}-2, \cdots, 1 \tag{21}$$

The new twiddle factors matrix ($\mathbf{T}_k^{\text{new-even}}$) is then given by

$$\mathbf{T}_k^{\text{new-even}} = diag(1, W_N^k, W_N^{2k}, \cdots, W_N^{\left(\frac{R}{2}-1\right)k}, W_N^{\left(\frac{R}{2}\right)k}, W_N^{-\left(\frac{R}{2}-1\right)k}, \cdots, W_N^{-2k}, W_N^{-k}) \quad (22)$$

and the expression for the butterfly of the proposed algorithm for an even R is

$$\mathbf{Y}_k = \mathbf{T}_k^{\text{new-even}} \mathbf{W}_R \mathbf{X}_k \quad (23)$$

with $W_N^{pk} = \cos\left(\frac{2\pi}{N} pk\right) - j\sin\left(\frac{2\pi}{N} pk\right)$ and $W_N^{-pk} = \cos\left(\frac{2\pi}{N} pk\right) + j\sin\left(\frac{2\pi}{N} pk\right)$, a glance to the structure of $\mathbf{T}_k^{\text{new-even}}$ shows that only R real twiddle factors need to be evaluated or loaded from the lookup table during the processing of the butterfly specified by (23) of the proposed algorithm instead of the $\left(2\left(R-1\right)\right)$ ones needed in the butterfly of the conventional CTA FFT algorithm. That implies a saving of precisely $\left(100\left(1-\frac{R}{2\left(R-1\right)}\right)\right)\%$ is achieved by the former compared to the latter. This saving tends to 50% for higher radix FFTs.

Let us now illustrate, in the following example, the computations involved in the first stage of the proposed radix-6 FFT algorithm for an N-point DFT, where $N=6^r$.

Example 3: The proposed radix-6 FFT algorithm for an N-point DFT, where $N=6^r$

$$X(6n) = \sum_{k=0}^{(N/6)-1} y_0(k) W_{N/6}^{nk} \quad 0 \le n \le \frac{N}{6} - 1$$

$$X(6n+1) = \sum_{k=0}^{(N/6)-1} y_1(k) W_{N/6}^{nk} \quad 0 \le n \le \frac{N}{6} - 1$$

$$X(6n+2) = \sum_{k=0}^{(N/6)-1} y_2(k) W_{N/6}^{nk} \quad 0 \le n \le \frac{N}{6} - 1$$

$$X(6n+3) = \sum_{k=0}^{(N/6)-1} y_3(k) W_{N/6}^{nk} \quad 0 \le n \le \frac{N}{6} - 1$$

$$X((N+6n-2) \bmod N) = \sum_{k=0}^{(N/6)-1} y_{-2}(k) W_{N/6}^{nk} \quad 0 \le n \le \frac{N}{6} - 1$$

$$X((N+6n-1) \bmod N) = \sum_{k=0}^{(N/6)-1} y_{-1}(k) W_{N/6}^{nk} \quad 0 \le n \le \frac{N}{6} - 1$$

where

$$\begin{bmatrix} y_0(k) \\ y_1(k) \\ y_2(k) \\ y_3(k) \\ y_{-2}(k) \\ y_{-1}(k) \end{bmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & W_N^k & 0 & 0 & 0 & 0 \\ 0 & 0 & W_N^{2k} & 0 & 0 & 0 \\ 0 & 0 & 0 & W_N^{3k} & 0 & 0 \\ 0 & 0 & 0 & 0 & W_N^{-2k} & 0 \\ 0 & 0 & 0 & 0 & 0 & W_N^{-k} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W_6^1 & W_6^2 & W_6^3 & W_6^4 & W_6^5 \\ 1 & W_6^2 & W_6^4 & 1 & W_6^2 & W_6^4 \\ 1 & W_6^3 & 1 & W_6^3 & 1 & W_6^3 \\ 1 & W_6^4 & W_6^2 & 1 & W_6^4 & W_6^2 \\ 1 & W_6^5 & W_6^4 & W_6^3 & W_6^2 & W_6^1 \end{pmatrix} \begin{bmatrix} x(k) \\ x(k+N/6) \\ x(k+2N/6) \\ x(k+3N/6) \\ x(k+4N/6) \\ x(k+5N/6) \end{bmatrix}$$

As for the odd radices, no extra computational complexity will occur in the generation of the new indexes since

$$(N+Rn-p) \bmod N = \begin{cases} N-p, & \text{for } n=0 \\ Rn-p, & \text{elsewhere} \end{cases} \quad 1 \le p \le \frac{R}{2}-1 \qquad (24)$$

Table 2 presents the number of real twiddle factors computations or loads from the lookup table required by the proposed and CTA algorithms for frequently used high even radices R=4, 6, 8 and 16. It is clear from this table that the advantage of using the proposed algorithm over the CTA is very remarkable. In FFT dedicated processors, the memory size of the lookup table is drastically reduced and is cut to almost half for R=16.

Table 2: Number of real twiddle factors computations or loads from the lookup table in the CTA and the proposed algorithm (case of even radices).

	Radix-R CTA				Proposed radix-R algorithm			
N \ R	4	6	8	16	4	6	8	16
16	18				12			
36		50				30		
64	162		98		108		56	
216		650				390		
256	1026			450	684			240
512			1666				952	
1024	5634				3756			
1296		6050				3630		
4096	28674		20482	14850	19116		11704	7920
7776		49250				29550		
16384	139266				92844			
32768			221186				126392	
46656		373250				223950		
65536	655362			360450	436908			192240

CONCLUSION

In this paper, an improved radix-R DIF FFT algorithm has been proposed. It has been obtained by introducing a new indexing process for nearly half of the output sub-sequences resulting from the conventional decomposition of the DFT. Without adding any computational or structural complexity, the proposed algorithm, which can be used with any arbitrary radix R, reduces significantly (up to 50%) the number of twiddle factors evaluations or accesses to the lookup table and therefore, a similar reduction is obtained in the address generations compared to the conventional radix-R DIF FFT algorithm. An analogous radix-R decimation-in-time (DIT) FFT algorithm, which will offer the same reductions in the number of twiddle factors evaluations or accesses to the lookup table, can as well be derived following derivations similar to the ones introduced in this paper.

REFERENCES

[1] Cooley, J.W., and Tukey, J.W., 1965, "An algorithm for machine computation of complex Fourier series," Math. Comp. 19 (90), pp. 297–301.

[2] Duhamel, P., and Vetterli, M., 1990, "Fast Fourier transforms: A tutorial review and a state of the art," Signal Processing 19(4), pp. 259–299.

[3] Jiang, L.-X., Liu, C.-Y., and Zhang, P., 2012,"A novel overall in-place in-order prime factor FFT algorithm," 5th International Congress on Image and Signal Processing (CISP), pp. 1500–1503.

[4] Harris, D.B., and McClellan, J.H., 1977, "Vector-radix fast Fourier transform, "Proceedings of the IEEE International Conference on Acoustics, Speech, Signal Processing, Hartford, CT, pp. 548–551.

[5] Duhamel, P., 1986, "Implementing of split-radix FFT algorithms for complex, real, and real-symmetric data," IEEE Transactions on Acoustics, Speech and Signal Processing 34(2), pp. 285–295.

[6] Pei, S.C., and Chen, W.Y., 2004,"Split vector-radix-2/8 2-D fast Fourier transform," IEEE Signal Processing Letters 11(5), pp. 459–462.

[7] Winograd, S., 1978, "On computing the discrete Fourier transform", Math. Comp. 32, pp.175–199.

[8] Chauhan, T., and Karwal, V., 2013, "DFT implementation aspects and techniques suitable for VLSI implementation: A survey," International Conference on Signal Processing and Communication, pp. 330–334.

[9] Jaber, M.A., and Massicotte, D., 2009,"A new FFT concept for efficient VLSI implementation: Part I - Butterfly processing element," 16th International Conference on Digital Signal Processing, pp. 1–6.

[10] Jaber, M.A., and Massicotte, D., 2009,"A new FFT concept for efficient VLSI implementation: Part II - Parallel Pipelined Processing, " 16th International Conference on Digital Signal Processing, pp. 1–5.

[11] Jaber, M.A., Massicotte, D., and Achouri, Y., 2011,"A higher radix FFT FPGA implementation suitable for OFDM systems,"18th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp. 744–747.

[12] Choudhary, P., and Karmakar, A., 2011,"CORDIC based implementation of Fast Fourier Transform," 2nd International Conference on Computer and Communication Technology (ICCCT), pp. 550–555.

[13] Zhou, J., 2013, "A new method to generate twiddle factor using CORDIC based radix-4 FFT butterfly," International Conference on Communications, Circuits and Systems (ICCCAS), vol. 2, pp. 505–508.

[14] Eltawil, A. M., and Daneshrad, B., 2002, "Piece-wise parabolic interpolation for direct digital frequency synthesis," Proceedings of the IEEE 2002 Custom Integrated Circuits Conference, pp. 401–404.

[15] Chi, J.C., and Chen, S.G., 2004, "An efficient FFT twiddle factor generator," 12th European Signal Processing Conference, pp. 1533–1536.

[16] Cohen, D., 1976, "Simplified control of FFT hardware," IEEE Transactions on Acoustics, Speech, and Signal Processing 24 (6), pp. 577–579.

[17] Ma, Y., and Wanhammar, L., 2000, "A hardware efficient control of memory addressing for high performance FFT processors," IEEE Transactions on Signal Processing 48(3), pp. 917–921.

[18] Hasan, M., and Arslan, T., 2002, "FFT coefficient memory reduction technique for OFDM applications," IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 1, pp. 1085–1088.

[19] Bouguezel, S., Ahmad, M.O., and Swamy, M.N.S., 2004,"Improved Radix-4 and Radix-8 FFT algorithms," International Symposium on Circuits and Systems, vol. 3, pp. 561–564.

[20] Bouguezel, S., Ahmad, M.O., and Swamy, M.N.S., 2004, "An improved Radix-16 FFT algorithm, " Canadian Conference on Electrical and Computer Engineering vol. 2, pp. 1089–1092.

[21] Bouguezel, S., Ahmad, M.O., and Swamy, M.N.S., 2006, "An Alternate Approach for Developing Higher Radix FFT Algorithms," IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp. 227–230.

[22] Qureshi, F., Gustafsson, O., 2009, "Analysis of twiddle factor memory complexity of radix-2^i pipelined FFTs," Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers, pp. 217–220.

[23] Chang, Y.-N., and Parhi, K.K., 2003, "An efficient pipelined FFT architecture, "IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 50(6), pp. 322–325.

[24] Goedecker, S., 1997, "Fast Radix 2, 3, 4, and 5 Kernels for Fast Fourier Transformations on Computers with Overlapping Multiply-Add Instructions," SIAM J. Sci. Comput. 18(6), pp. 1605–1611.

[25] Lofgren, J., and Nilsson, P., 2011, "On hardware implementation of radix 3 and radix 5 FFT kernels for LTE systems," NORCHIP, pp. 1–4.

[26] Wang, G., Yin, B., Cho, I., Cavallaro, J.R., Bhattacharyya, S. , and Takala, J., 2014, "Efficient architecture mapping of FFT/IFFT for cognitive radio networks," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3933–3937.