Hindawi Publishing Corporation
VLSI Design
Volume 2014, Article ID 690594, 13 pages
http://dx.doi.org/10.1155/2014/690594

Research Article

Radix-$2^\alpha/4^\beta$ Building Blocks for Efficient VLSI's Higher Radices Butterflies Implementation

Marwan A. Jaber and Daniel Massicotte

Laboratory of Signals and Systems Integrations, Electrical and Computer Engineering Department, Université du Québec à Trois-Rivières, QC, Canada G9A 5H7

Correspondence should be addressed to Daniel Massicotte; daniel.massicotte@uqtr.ca

Received 27 December 2013; Revised 12 March 2014; Accepted 26 March 2014; Published 13 May 2014

Academic Editor: Dionysios Reisis

This paper describes an embedded FFT processor where the higher radices butterflies maintain one complex multiplier in its critical path. Based on the concept of a radix-r fast Fourier factorization and based on the FFT parallel processing, we introduce a new concept of a radix-r Fast Fourier Transform in which the concept of the radix-r butterfly computation has been formulated as the combination of radix-$2^\alpha/4^\beta$ butterflies implemented in parallel. By doing so, the VLSI butterfly implementation for higher radices would be feasible since it maintains approximately the same complexity of the radix-2/4 butterfly which is obtained by block building of the radix-2/4 modules. The block building process is achieved by duplicating the block circuit diagram of the radix-2/4 module that is materialized by means of a feed-back network which will reuse the block circuit diagram of the radix-2/4 module.

1. Introduction

For the past decades, the main concern of the researchers was to develop a fast Fourier transform (FFT) algorithm in which the number of operations required is minimized. Since Cooley and Tukey presented their approach showing that the number of multiplications required to compute the discrete Fourier transform (DFT) of a sequence may be considerably reduced by using one of the fast Fourier transform (FFT) algorithms [1], interest has arisen both in finding applications for this powerful transform and for considering various FFT software and hardware implementations.

The DFT computational complexity increases according to the square of the transform length and thus becomes expensive for large N. Some algorithms used for efficient DFT computation, known as fast DFT computation algorithms, are based on the divide-and-conquer approach. The principle of this method is that a large problem is divided into smaller subproblems that are easier to solve. In the FFT case, dividing the work into subproblems means that the input data $x_{[n]}$ can be divided into subsets from which the DFT is computed, and then the DFT of the initial data is reconstructed from these intermediate results. Some of these methods are known as the Cooley-Tukey algorithm [1], split-radix algorithm [2], Winograd Fourier transform algorithm (WFTA) [3], and others, such as the common factor algorithms [4].

The problem with the computation of an FFT with an increasing N is associated with the straightforward computational structure, the coefficient multiplier memories' accesses, and the number of multiplications that should be performed. The overall arithmetic operations deployed in the computation of an N-point FFT decreases with increasing r as a result; the butterfly complexity increases in terms of complex arithmetic computation, parallel inputs, connectivity, and number of phases in the butterfly's critical path delay. The higher radix butterfly involves a nontrivial VLSI implementation problem (i.e., increasing butterfly critical path delay), which explains why the majority of FFT VLSI implementations are based on radix 2 or 4, due to their low butterfly complexity. The advantage of using a higher radix is that the number of multiplications and the number of stages to execute an FFT decrease [4–6].

The most recent attempts to reduce the complexity of the higher radices butterfly's critical path was achieved by the concept of a radix-r fast Fourier transform (FFT) [8, 9], in which the concept of the radix-r butterfly computation has been formulated as composed engines with identical structures and a systematic means of accessing the corresponding multiplier coefficients. This concept enables the design of butterfly processing element (BPE) with the lowest rate of complex multipliers and adders, which utilizes r or $r-1$ complex multipliers in parallel to implement each of the butterfly computations. Another strategy was based on targeting hardware oriented radix 2^{α} or 4^{β} which is an alternative way of representing higher radices by means of less complicated and simple butterflies in which they used the symmetry and periodicity of the root unity to further lower down the coefficient multiplier memories' accesses [10–20].

Based on the higher radices butterfly and the parallel FFT concepts [21, 22], we will introduce the structure of higher multiplexed 2^{α} or 4^{β} butterflies that will reduce the resources in terms of complex multiplier and adder by maintaining the same throughput and the same speed in comparison to the other proposed butterflies structures in [13–20].

This paper is organized as follows. Section 2 describes the higher radices butterfly computation and Section 3 details the FFT parallel processing. Section 4 elaborates the proposed higher radices butterflies; meanwhile Section 5 draws the performance evaluation of the proposed method and Section 6 is devoted to the conclusion.

2. Higher Radices' Butterfly Computation

The basic operation of a radix-r PE is the so-called butterfly computation in which r inputs are combined to give the r outputs via the following operation:

$$\mathbf{X} = \mathbf{B}_r \mathbf{x}_{\text{in}},$$
$$\mathbf{x}_{\text{in}} = \left[x_{(0)}, x_{(1)}, \dots, x_{(r-1)}\right]^T, \quad (1)$$
$$\mathbf{X} = \left[X_{(0)}, X_{(1)}, \dots, X_{(r-1)}\right]^T,$$

where $\mathbf{x}_{\text{in}}$ and $\mathbf{X}$ are, respectively, the butterfly's input and output vectors. $\mathbf{B}_r$ is the butterfly matrix ($\dim(\mathbf{B}_r) = r \times r$) which can be expressed as

$$\mathbf{B}_r = \mathbf{W}_N \mathbf{T}_r, \quad (2)$$

for decimation in frequency (DIF) process, and

$$\mathbf{B}_r = \mathbf{T}_r \mathbf{W}_N, \quad (3)$$

for decimation in time (DIT) process. In both cases the twiddle factor matrix, $\mathbf{W}_N$, is a diagonal matrix which is defined by $\mathbf{W}_N = \operatorname{diag}(1, w_N^{p}, w_N^{2p}, \dots, w_N^{(r-1)p})$ with $p = 0, 1, \dots, N/r^s - 1$ and $s = 0, 1, \dots, \log_r N - 1$ and $\mathbf{T}_r$ is the *adder tree* matrix within the butterfly structure expressed as [4]

$$\mathbf{T}_r = \begin{bmatrix} w_N^0 & w_N^0 & w_N^0 & \cdots & \cdots & w_N^0 \\ w_N^0 & w_N^{N/r} & w_N^{2N/r} & \cdots & \cdots & w_N^{(r-1)N/r} \\ w_N^0 & w_N^{2N/r} & w_N^{4N/r} & \vdots & \vdots & w_N^{2(r-1)N/r} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_N^0 & w_N^{(r-1)N/r} & w_N^{2(r-1)N/r} & \cdots & \cdots & w_N^{(r-1)^2 N/r} \end{bmatrix}. \quad (4)$$

As seen from (2) and (3), the adder tree, $\mathbf{T}_r$, is almost identical for the two algorithms, with the only difference being the order in which the twiddle factor and the adder tree multiplication are computed. A straightforward implementation of the adder tree is not effective for higher radices butterflies due to the added complex multipliers in the higher radices butterflies' critical path that will complicate its implementation in VLSI.

By defining the element of the lth line and the mth column in the matrix $\mathbf{T}_r$ as $[\mathbf{T}_r]_{l,m}$,

$$\left[\mathbf{T}_r\right]_{l,m} = w_N^{[\![(lmN/r)]\!]_N}, \quad (5)$$

where $l = 0, 1, \dots, r-1$, $m = 0, 1, \dots, r-1$, and $[\![x]\!]_N$ represents the operation x modulo N. By defining $\mathbf{W}_{N(m,v,s)}$ the set of the twiddle factor matrix as

$$\left[\mathbf{W}_N\right]_{l,m(v,s)} = \operatorname{diag}\left(w_{N(0,v,s)}, w_{N(1,v,s)}, \dots, w_{N(r-1,v,s)}\right), \quad (6)$$

where the index r is the FFT's radix, $v = 0, 1, \dots, V-1$ represents the number of words of size r ($V = N/r$), and $s = 0, 1, \dots, S$ is the number of stages (or iterations $S = \log_r N - 1$). Finally, the twiddle factor matrix in (2) and (3) can be expressed for the different stages of an FFT process as [7, 8]

$$\left[\mathbf{W}_N\right]_{l,m(v,s)} = \begin{cases} w_N^{[\![\lfloor v/r^s \rfloor l r^s]\!]_N} & \text{for } l = m \\ 0 & \text{elsewhere,} \end{cases} \quad (7)$$

for the DIF process and (3) would be expressed as

$$\left[\mathbf{W}_N\right]_{l,m(v,s)} = \begin{cases} w_N^{[\![\lfloor v/r^{(S-s)} \rfloor l r^{(S-s)}]\!]_N} & \text{for } l = m \\ 0 & \text{elsewhere,} \end{cases} \quad (8)$$

for the DIT process, where $l = 0, 1, \dots, r-1$ is the lth butterfly's output, $m = 0, 1, \dots, r-1$ is the mth butterfly's input, and $\lfloor x \rfloor$ represents the integer part operator of x.

As a result, the lth transform output during each stage can be illustrated as

$$\mathbf{X}_{(v,s)}[l] = \sum_{m=0}^{r-1} x_{(v,s)}[m]\, w_N^{[\![lmN/r + \lfloor v/r^s \rfloor l r^s]\!]_N}, \quad (9)$$

for the modified DIF process, and

$$\mathbf{X}_{(v,s)}[l] = \sum_{m=0}^{r-1} x_{(v,s)}[m]\, w_N^{[\![lmN/r + \lfloor v/r^{(S-s)} \rfloor m r^{(S-s)}]\!]_N}, \quad (10)$$

for the modified DIT process.

The conceptual key to the modified radix-r FFT butterfly is the formulation of the radix-r as composed engines with identical structures and a systematic means of accessing the corresponding multiplier coefficients [8, 9]. This enables the design of an engine with the lowest rate of complex multipliers and adders, which utilizes r or $r-1$ complex multipliers in parallel to implement each of the butterfly computations. There is a simple mapping from the three indices m, v, and s (FFT stage, butterfly, and element) to the addresses of the multiplier coefficients needed by using the proposed FFT address generator in [24]. For a single processor environment, this type of butterfly with r parallel multipliers would result in decreasing the time delay for the complete FFT by a factor of $O(r)$. A second aspect of the modified radix-r FFT butterfly is that they are also useful in parallel multiprocessing environments. In essence, the precedence relations between the engines in the radix-r FFT are such that the execution of r engines in parallel is feasible during each FFT stage. If each engine is executed on the modified processing element (PE), it means that each of the r parallel processors would always be executing the same instruction simultaneously, which is very desirable for SIMD implementation on some of the latest DSP cards.

Based on this concept, Kim and Sunwoo proposed a proper multiplexing scheme that reduces the usage of complex multiplier for the radix-8 butterfly from 11 to 5 [25].

3. Parallel FFT Processing

For the past decades, there were several attempts to parallelize the FFT algorithm which was mostly based on parallelizing each stage (iteration) of the FFT process [26–28]. The most successful FFT parallelization was accomplished by parallelizing the loops during each stage or iteration in the FFT process [29, 30] or by focusing on memory hierarchy utilization that is achieved by the combination of production and consumption of butterflies' results, data reuse, and FFT parallelism [31].

The definition of the DFT is represented by

$$X_{(k)} = \sum_{n=0}^{N-1} x_{(k)} w_N^{nk}, \quad k \in [0, N-1], \tag{11}$$

where $x_{(n)}$ is the input sequence, $X_{(k)}$ is the output sequence, N is the transform length, and w_N is the Nth root of unity: $w_N = e^{-j2\pi/N}$. Both $x_{(n)}$ and $X_{(k)}$ are complex valued sequences.

Let $x_{(n)}$ be the input sequence of size N and let p_r denote the degree of parallelism which is multiple of N; therefore, we can rewrite (11) by considering $k_1 = 0, 1, \ldots, V-1$, $p = 0, 1, \ldots, p_r - 1$, $q = 0, 1, \ldots, p_r - 1$, $V = N/p_r$, and $k = k_1 + qV$ as [9]

$$\begin{aligned} &X_{(k_1 + qN/p_r)} \\ &= \left[w_N^0 \sum_{n=0}^{N/p_r - 1} x_{(p_r n)} w_{N/p_r}^{n(k_1 + q_r N/p_r)} \right. \\ &\quad + w_N^{(k_1 + qN/p_r)} \sum_{n=0}^{N/p_r - 1} x_{(p_r n + 1)} w_{N/p_r}^{n(k_1 + qN/p_r)} \\ &\quad \left. + \cdots + w_N^{(p_r - 1)(k_1 + qN/p_r)} \sum_{n=0}^{N/p_r - 1} x_{(p_r n + (p_r - 1))} w_{N/p_r}^{n(k_1 + qN/p_r)} \right]. \end{aligned} \tag{12}$$

If $X_{(k)}$ is the Nth order Fourier transform $\sum_{n=0}^{N-1} x_{(n)} w_N^{nk}$, then, $X_{(0)_{(k_1)}}, X_{(1)_{(k_1)}}, \ldots$, and $X_{(p_r - 1)_{(k_1)}}$ will be the Nth$/p_r$ order Fourier transforms given, respectively, by the following expressions: $\sum_{n=0}^{V-1} x_{(p_r n)} w_V^{nv}$, $\sum_{n=0}^{V-1} x_{(p_r n+1)} w_V^{nv}, \ldots$, and $\sum_{n=0}^{V-1} x_{(p_r n + (p_r - 1))} w_V^{nv}$.

4. The Proposed Higher Radices Butterflies

Most of the FFTs' computation transforms are done within the butterfly loops. Any algorithm that reduces the number of additions and multiplications in these loops will reduce the overall computation speed. The reduction in computation is achieved by targeting trivial multiplications which have a limited speedup or by parallelizing the FFTs that have a significant speedup on the execution time of the FFT. In this section we will be limited in the elaboration of the proposed butterfly's radix-$2^\alpha/4^\beta$ (the radix-2/4 families) for the DIT FFT process. By rewriting (3) as

$$\mathbf{X} = \mathbf{W}_N \sum_{m=0}^{r-1} x_{(m)} w_N^{lmN/r} = \mathbf{W}_N \sum_{m=0}^{r-1} x_{(m)} w_r^{lm} \tag{13}$$

and by applying the concept of the parallel FFT (introduced in Section 3) on the kernel $\mathbf{B}_r$, therefore, (13) will be expressed as

$$\begin{aligned} \mathbf{X} &= \mathbf{W}_N \sum_{m=0}^{r-1} x_{(m)} w_r^{lm} \\ &= \mathbf{W}_N \left[\sum_{m=0}^{r/\alpha - 1} x_{(\alpha m)} w_r^{lm\alpha} + \cdots \right. \\ &\qquad \left. + \sum_{m=0}^{r/\alpha - 1} x_{(\alpha m + (\alpha - 1))} w_r^{l(\alpha m + (\alpha - 1))} \right] \\ &= \mathbf{W}_N \left[X_{(0)} + w_r^l X_{(1)} + \cdots + w_r^{l(\alpha - 1)} X_{(\alpha - 1)} \right] \\ &\qquad \text{for } l = 0, \ldots, \frac{r}{\alpha} - 1. \end{aligned} \tag{14}$$

It is to be noted that the notation w_x in all figures of this paper represents the set of twiddle factor associated with the butterfly input defined by $[w_0, \ldots, w_{(r-2)}] = \operatorname{diag}(w_N^p, w_N^{2p}, \ldots, w_N^{(r-1)p})$.

For the radix-4 butterfly ($r = 2$ and $\alpha = 2$), we can express (13) as

$$\begin{aligned} \mathbf{X} &= \mathbf{W}_N \left[\sum_{m=0}^{1} x_{(2m)} w_2^{lm} + w_4^l \sum_{m=0}^{1} x_{(2m+1)} w_2^{lm} \right] \\ &= \mathbf{W}_N \left[X_{(0)} + w_4^l X_{(1)} \right], \end{aligned} \tag{15}$$

and the conventional radix-2^2 (MDC-R2^2) BPE in terms of radix-2 butterfly is illustrated in Figure 1.

The use of resources could also be reduced by a feedback network and a multiplexing network where the feedback network is for feeding the ith output of the jth radix-2 adder network to the jth input of the ith butterfly and the multiplexers selectively pass the input data or the feedback, alternately, to the corresponding radix-2 adder network as

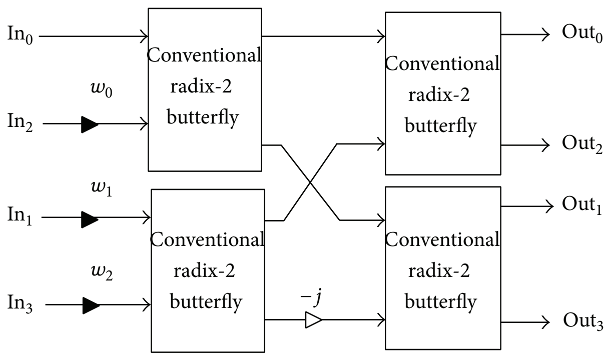

FIGURE 1: Conventional radix-2^2 (MDC-R2^2) BPE (butterfly processing element).

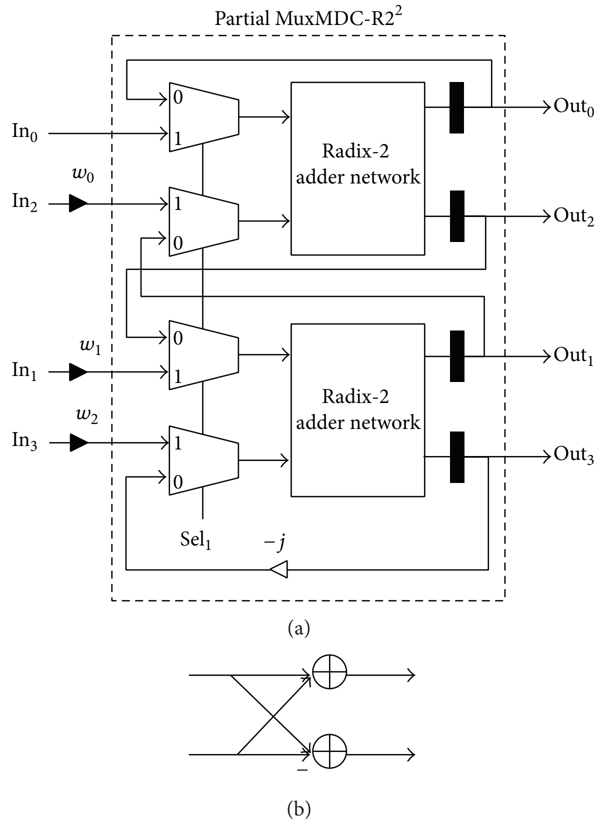

FIGURE 2: (a) Proposed multiplexed radix-2^2 (MuxMDC-R2^2) BPE and (b) block circuit diagram of the radix-2 adder network [7].

illustrated in Figure 2(a) [23]. The circuit block diagram of the radix-2 adder network is illustrated in Figure 2(b) that consists of two complex adders only.

With the rising edge of the clock cycle the inputs data are fed to the butterfly's input of the system presented in Figure 1. In order to complete the butterfly's operations within one clock cycle, the following conditions should be satisfied:

$$T_{\mathrm{CLK}} > T_{\mathrm{CM}} + 2T_{\mathrm{CA}}, \qquad \text{Throughput} = \frac{4}{T_{\mathrm{CLK}}}, \tag{16}$$

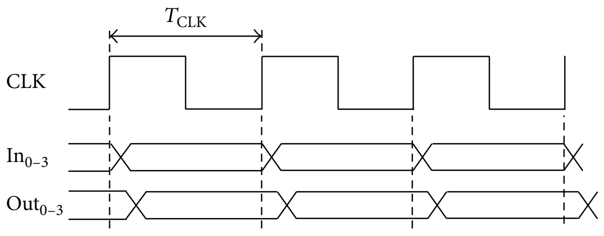

FIGURE 3: Timing block diagram of Figure 1.

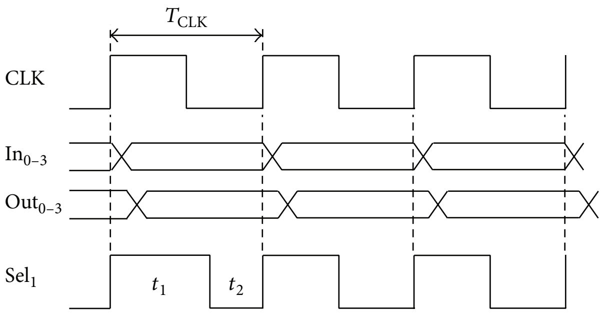

FIGURE 4: Timing block diagram of Figure 2(a).

where $T_{\mathrm{CM}}/T_{\mathrm{CA}}$ is the time required to perform one complex multiplication/addition and the timing block diagram of Figure 1 is sketched in Figure 3.

With the rising edge of the clock cycle the inputs data are fed to the butterfly's input of the system presented in Figure 2(a) and with the falling edge of the clock cycle the feedback data are fed to the butterfly's input. In order to complete the butterfly's operations within one clock cycle, the following conditions should be satisfied:

$$\begin{gathered} t_1 > T_{\mathrm{CM}} + T_{\mathrm{CA}}, \\ t_2 > T_{\mathrm{CA}}, \\ T_{\mathrm{CLK}} > (t_1 + t_2) > T_{\mathrm{CM}} + 2T_{\mathrm{CA}}, \\ \text{Throughput} = \frac{4}{T_{\mathrm{CLK}}}, \end{gathered} \tag{17}$$

and the timing block diagram of Figure 2(a) is illustrated in Figure 4.

Further block building of these modules could be achieved by duplicating the block circuit diagram of Figure 2(a) and combining them in order to obtain the radix-8 MDC-R2^3 BPE; therefore, for this case ($r = 4$ and $\alpha = 2$), (4) could be expressed as

$$\begin{aligned} \mathbf{X}_{(l)} &= \mathbf{W}_N \left[\sum_{m=0}^{3} x_{(2m)} w_4^{lm} + w_8^l \sum_{m=0}^{3} x_{(2m+1)} w_4^{lm} \right] \\ &= \mathbf{W}_N \left[X_{(0)} + w_8^l X_{(1)} \right], \end{aligned} \tag{18}$$

and the signal flow graph (SFG) of the DIT conventional MDC-R2^3 BPE butterfly is illustrated in Figure 5. The resources in the conventional MDC-R2^3 BPE could also be

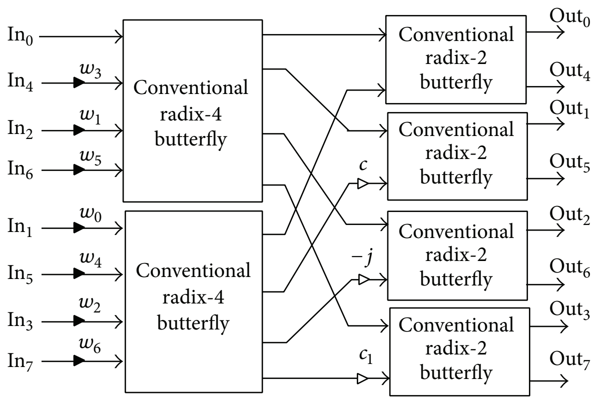
In0
In4
In2
In6
In1
In5
In3
In7
w3
w1
w5
w0
w4
w2
w6
Conventional radix-4 butterfly
Conventional radix-4 butterfly
Conventional radix-2 butterfly
c
−j
c1
Out0
Out4
Out1
Out5
Out2
Out6
Out3
Out7

FIGURE 5: Conventional MDC-R2^3 BPE.

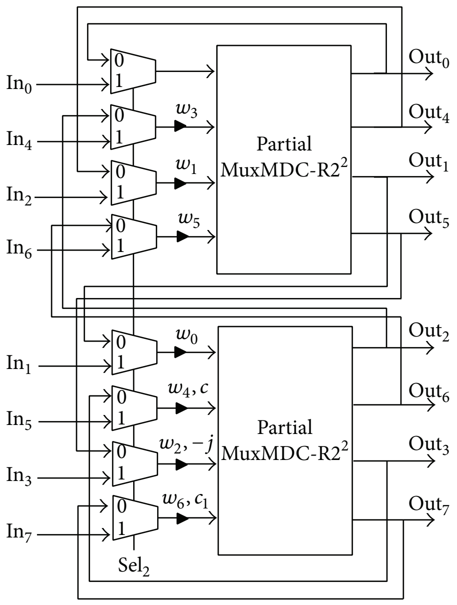

FIGURE 6: Proposed MuxMDC-R2^3 BPE based on the partial MuxMDC-R2^2.

reduced by means of the partial multiplexed radix 2^2 and a feedback network yielding to the proposed MuxMDC-R2^3 BPE structure in Figure 6.

The clock timing of Figure 5 is computed as

$$\begin{aligned} T_{\mathrm{CLK}} &> T_{\mathrm{CM}} + t_{\mathrm{pm}} + 3T_{\mathrm{CA}}, \\ \text{Throughput} &= \frac{8}{T_{\mathrm{CLK}}}, \end{aligned} \tag{19}$$

where t_{pm} is the time required to execute one complex multiplication on a constant multiplier and the clock timing of the proposed MuxMDC-R2^3 is estimated as

$$\begin{aligned} t_1 &> T_{\mathrm{CM}} + T_{\mathrm{CA}}, \\ t_2 &> T_{\mathrm{CA}}, \\ t_3 &= t_1, \\ T_{\mathrm{CLK}} &> (t_1 + t_2 + t_3) > 2T_{\mathrm{CM}} + 3T_{\mathrm{CA}}, \\ \text{Throughput} &= \frac{8}{T_{\mathrm{CLK}}}. \end{aligned} \tag{20}$$

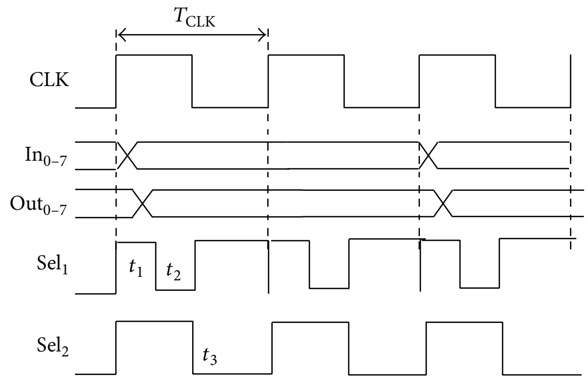

FIGURE 7: Timing block diagram of Figure 6.

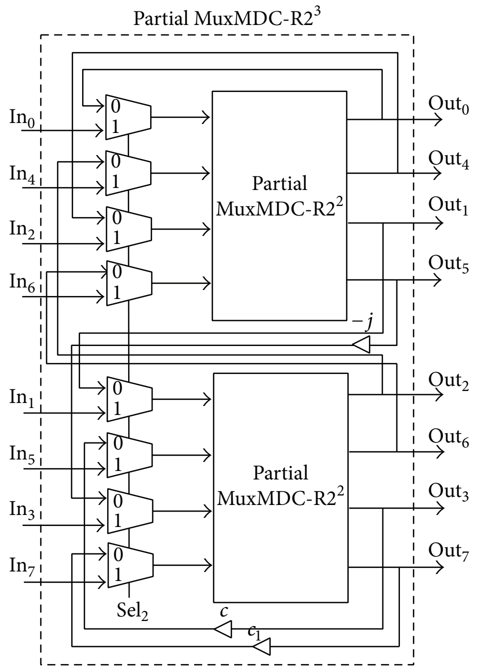

FIGURE 8: Proposed Partial MuxMDC-R2^3.

The overall timing block diagram of the proposed MuxMDC-R2^3 is sketched in Figure 7. In Figure 6, the inputs are multiplied by the twiddle factors w_i when $S_2 = 1$ and by the constant factors $-j$, c, c_1 or 1 for $S_2 = 0$.

Further block building of these modules could be achieved by combining two radix-8 butterflies with eight radix-2 butterflies in order to obtain the conventional MDC-R2^4 BPE; therefore, for this case ($r = 8$ and $\alpha = 2$), (4) could be expressed as

$$\begin{aligned} \mathbf{X}_{(l)} &= \mathbf{W}_N \left[\sum_{m=0}^{7} x_{(2m)} w_8^{lm} + w_8^l \sum_{m=0}^{7} x_{(2m+1)} w_8^{lm} \right] \\ &= \mathbf{W}_N \left[X_{(0)} + w_{16}^l X_{(1)} \right], \end{aligned} \tag{21}$$

and the signal flow graph (SFG) of the proposed DIT radix-2^4 MuxMDC-R2^4 based on the partial MuxMDC-R2^3 (Figure 8) is illustrated in Figure 9.

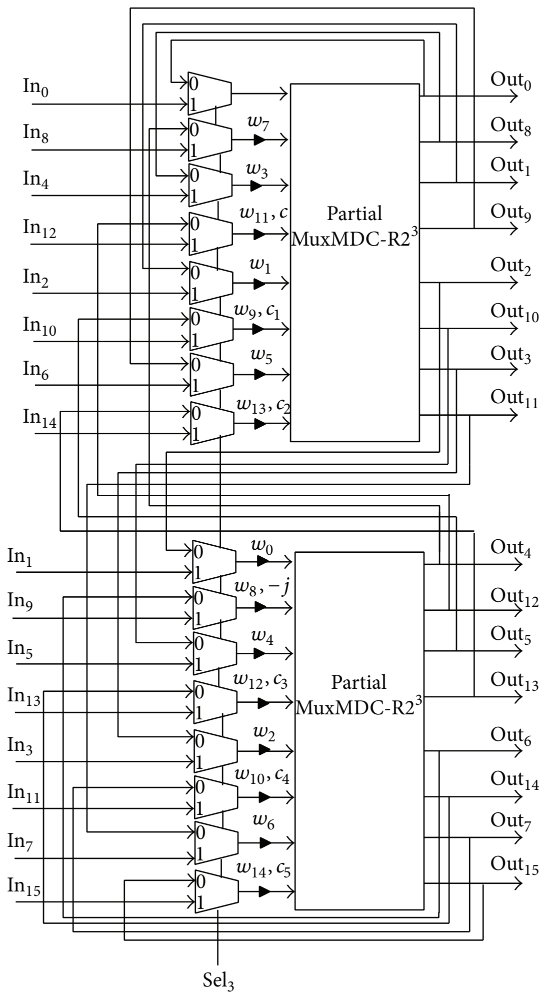

Figure 9: Proposed MuxMDC-R2^4 BPE based on the Partial MuxMDC-R2^3.

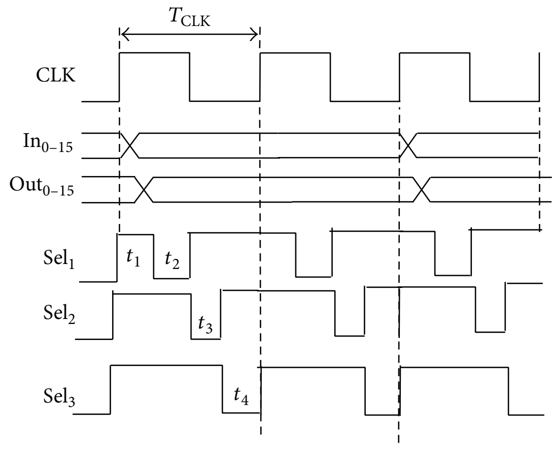

Figure 10: Timing block diagram of Figure 6.

The clock timing of the conventional MDC-R2^4 BPE is computed as

$$T_{\mathrm{CLK}} > T_{\mathrm{CM}} + 2t_{\mathrm{pm}} + 4T_{\mathrm{CA}},$$
$$\text{Throughput} = \frac{16}{T_{\mathrm{CLK}}}, \tag{22}$$

and the clock timing of the proposed MuxMDC-R2^4 is estimated as

$$t_1 > T_{\mathrm{CM}} + T_{\mathrm{CA}},$$
$$t_2 > T_{\mathrm{CA}},$$
$$t_3 = t_{\mathrm{pm}} + T_{\mathrm{CA}},$$

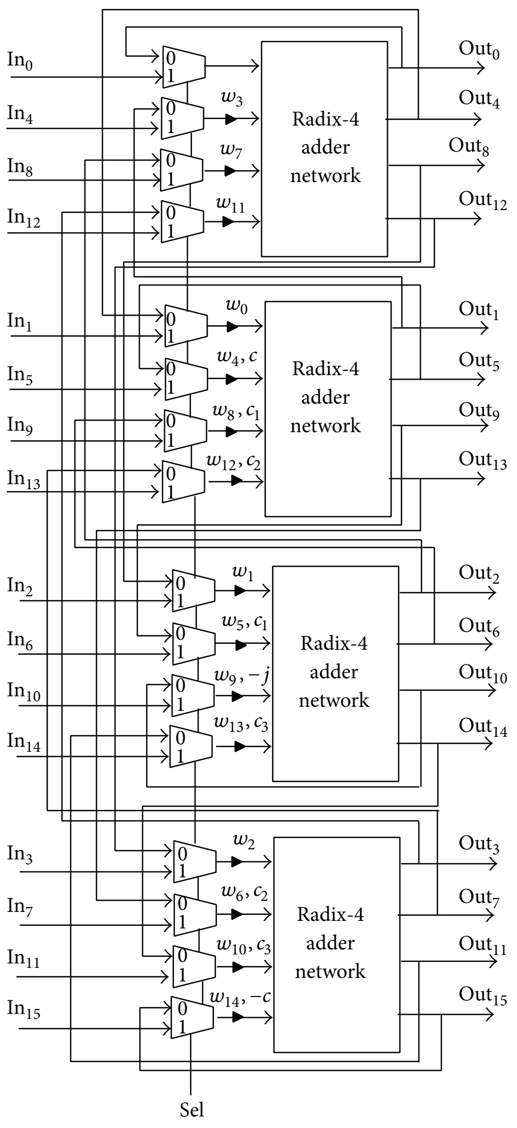

Figure 11: The proposed DIT MuxMDC-R4^2.

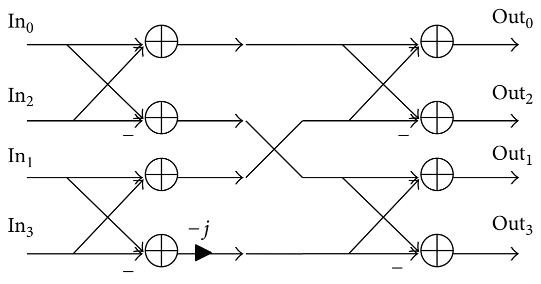

Figure 12: Block circuit diagram of the radix-4 adder network.

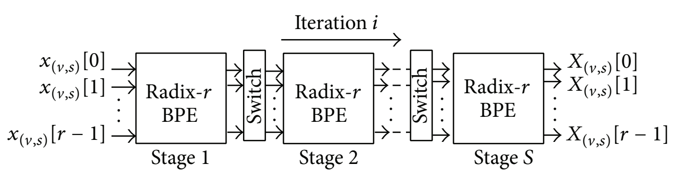

Figure 13: S stages radix-r pipelined FFT.

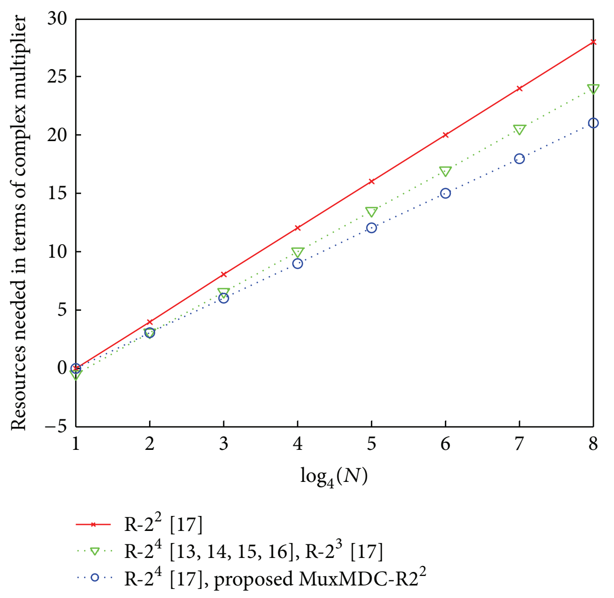

Figure 14: Comparison between the different butterflies' structures in terms of complex multiplier needed to compute the 4 parallel BPE pipelined FFTs of size N.

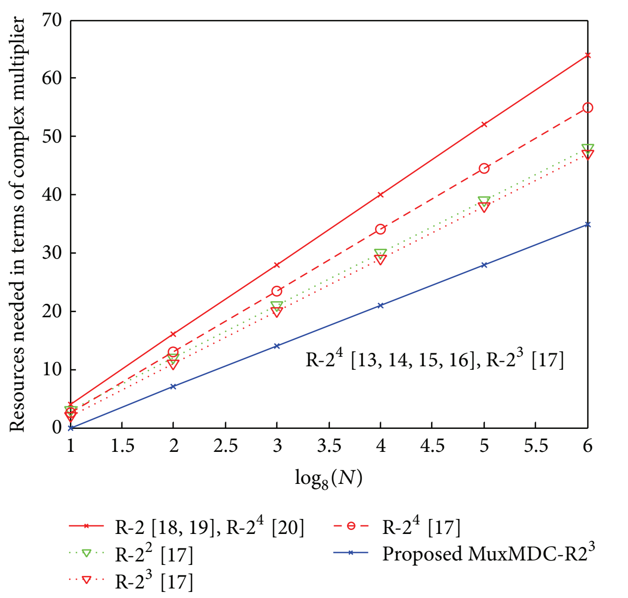

Figure 16: Comparison between the different butterflies' structures in terms of complex multiplier needed to compute the 8 parallel BPE pipelined FFTs of size N.

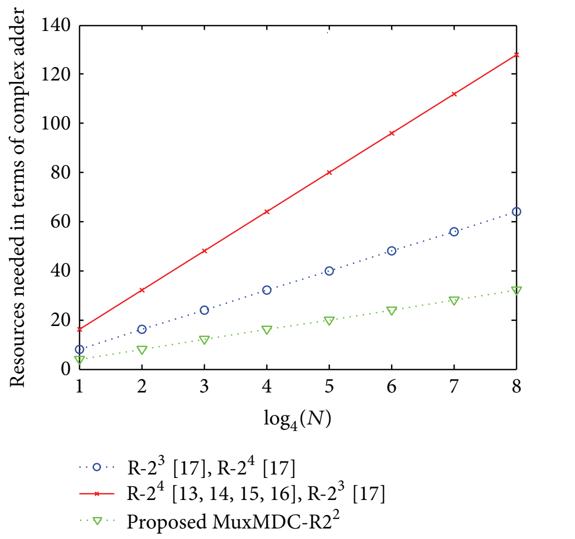

Figure 15: Comparison between the different butterflies' structures in terms of complex adder needed to compute the 4 parallel BPE pipelined FFTs of size N.

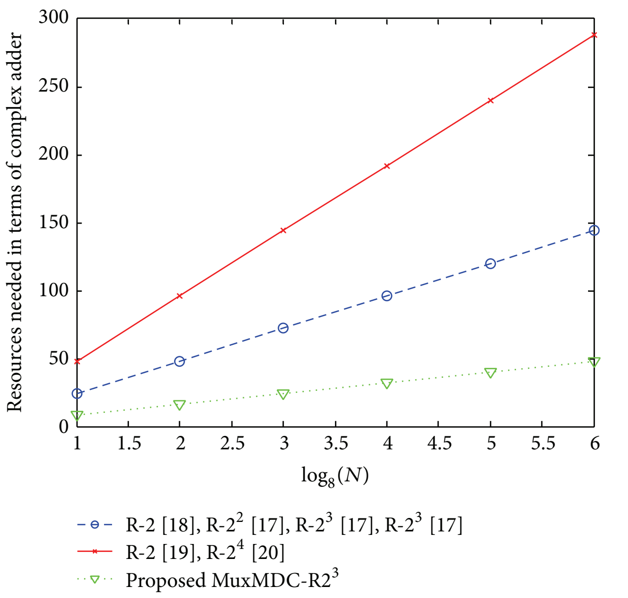

Figure 17: Comparison between the different butterflies' structures in terms of complex adder needed to compute the 8 parallel BPE pipelined FFTs of size N.

$$t_4 = t_1,$$
$$T_{\mathrm{CLK}} > (t_1 + t_2 + t_3 + t_4) > 2T_{CM} + t_{\mathrm{pm}} + 4T_{\mathrm{CA}},$$
$$\text{Throughput} = \frac{16}{T_{\mathrm{CLK}}}. \tag{23}$$

The overall timing block diagram of the proposed MuxMDC-R2^4 is sketched in Figure 10.

With the same reasoning as above, we will be limited in the elaboration of the proposed butterfly's radix-4$^\alpha$ family to the DIT FFT process.

For the radix-16 butterfly ($r = 4$ and $\alpha = 4$), we can express (4) as

$$\begin{aligned}\mathbf{X} = \mathbf{W}_N \Bigg[&\sum_{m=0}^{3} x_{(4m)} w_4^{lm} + w_{16}^{l} \sum_{m=0}^{3} x_{(4m+1)} w_4^{lm} \\ &+ w_{16}^{2l} \sum_{m=0}^{3} x_{(4m+2)} w_4^{lm} + w_{16}^{3l} \sum_{m=0}^{3} x_{(2m+1)} w_4^{lm} \Bigg] \\ = \mathbf{W}_N &\left[X_{(0)} + w_{16}^{l} X_{(1)} + w_{16}^{2l} X_{(2)} + w_{16}^{3l} X_{(3)} \right],\end{aligned} \tag{24}$$

Table 1: Resources needed to compute an FFT of size N.

Butterfly structure	Complex multiplier	Complex adder	Latency (cycles)	T (Spc)
		4 parallel BPE architectures		
R-2^4 [13], [14]	$4(\log_4 N - 1)$	$16\log_4 N$	$N/4$	4
R-2^4 [15]	$4(\log_4 N - 1)$	$16\log_4 N$	$N/4$	4
R-2^4 [16]	$4(\log_4 N - 1)$	$16\log_4 N$	$N/4$	4
R-2^2 [17]	$3(\log_4 N - 1)$	$16\log_4 N$	$N/4$	4
R-2^3 [17]	$4(\log_4 N - 1)$	$8\log_4 N$	$N/4$	4
R-2^4 [17]	$3.5(\log_4 N - 4)$	$8\log_4 N$	$N/4$	4
Proposed MuxMDC-R2^2	$3(\log_4 N - 1)$	$4\log_4 N$	$N/4$	4
		8 parallel BPE architectures		
R-2 [18]	$8(\log_4 N - 1)$	$16\log_4 N$	$N/8$	8
R-2 [19]	$8(\log_4 N - 1)$	$32\log_4 N$	$N/8$	8
R-2^4 [20]	$8(\log_4 N - 1)$	$32\log_4 N$	$N/8$	8
R-2^2 [17]	$6(\log_4 N - 1)$	$16\log_4 N$	$N/8$	8
R-2^3 [17]	$6\log_4 N - 7$	$16\log_4 N$	$N/8$	8
R-2^4 [17]	$7\log_4 N - 8$	$16\log_4 N$	$N/8$	8
Proposed MuxMDC-R2^3	$7(\log_8 N - 1)$	$8\log_8 N$	$N/8$	8
		16 parallel BPE architectures		
Proposed MuxMDC-R2^4	$17(\log_{16} N - 1)$	$16\log_{16} N$	$N/16$	16
Proposed MuxMDC-R4^2	$15(\log_{16} N - 1)$	$32\log_{16} N$	$N/16$	16

Table 2: Resources needed in terms of FA to compute an FFT of size N.

Butterfly structure	FA
4 parallel BPE architectures	
R-2^4 [13], [14]	$12n^2(\log_4 N - 1) + 20(p\log_4 N - 1) + 32p\log_4 N$
R-2^4 [15]	$12n^2(\log_4 N - 1) + 20p(\log_4 N - 1) + 32p\log_4 N$
R-2^4 [16]	$12n^2\log_4(N - 1) + 20p\log_4(N - 1) + 32p\log_4 N$
R-2^2 [17]	$9n^2(\log_4 N - 1) + 15p(\log_4 N - 1) + 32p\log_4 N$
R-2^3 [17]	$12n^2(\log_4 N - 1) + 20p(\log_4 N - 1) + 16p\log_4 N$
R-2^4 [17]	$10.5n^2(\log_4 N - 1) + 17.5p(\log_4 N - 1) + 16p\log_4 N$
Proposed MuxMDC-R2^2	$9n^2(\log_4 N - 1) + 15p(\log_4 N - 1) + 8p\log_4 N$
8 parallel BPE architectures	
R-2 [18]	$24n^2(\log_4 N - 1) + 40p(\log_4 N - 1) + 32p\log_4 N$
R-2 [19]	$24n^2(\log_4 N - 1) + 40p(\log_4 N - 1) + 64p\log_4 N$
R-2^4 [20]	$24n^2(\log_4 N - 1) + 40p(\log_4 N - 1) + 64p\log_4 N$
R-2^2 [17]	$18n^2(\log_4 N - 1) + 30p(\log_4 N - 1) + 32p\log_4 N$
R-2^3 [17]	$18n^2(\log_4 N - 1) + 30p(\log_4 N - 1) + 32p\log_4 N - 21n^2 - 35p$
R-2^4 [17]	$21n^2(\log_4 N - 1) + 35p(\log_4 N - 1) + 32p\log_4 N - 24n^2 - 40p$
Proposed MuxMDC-R2^3	$21n^2(\log_8 N - 1) + 35p(\log_8 N - 1) + 16p\log_8 N$
16 parallel BPE architectures	
Proposed MuxMDC-R2^4	$51n^2(\log_{16} N - 1) + 85p(\log_{16} N - 1) + 32p\log_{16} N$
Proposed MuxMDC-R4^2	$45n^2(\log_{16} N - 1) + 75p(\log_{16} N - 1) + 64p\log_{16} N$

and the proposed MDC-R4^2 in terms of radix-4 network is illustrated in Figure 11 where the feedback network is for feeding the ith output of the jth radix-4 network to the jth input of the ith butterfly and the switches selectively pass the input data or the feedback, alternately, to the corresponding radix-4 butterfly. The circuit block diagram of the radix-4 network is illustrated in Figure 12.

5. Performance Evaluation

FFTs are the most powerful algorithms that are used in communication systems such as OFDM. Their implementation is very attractive in fixed point due to the reduction in cost compared to the floating point implementation. One of the most powerful FFT implementations is the pipelined FFT

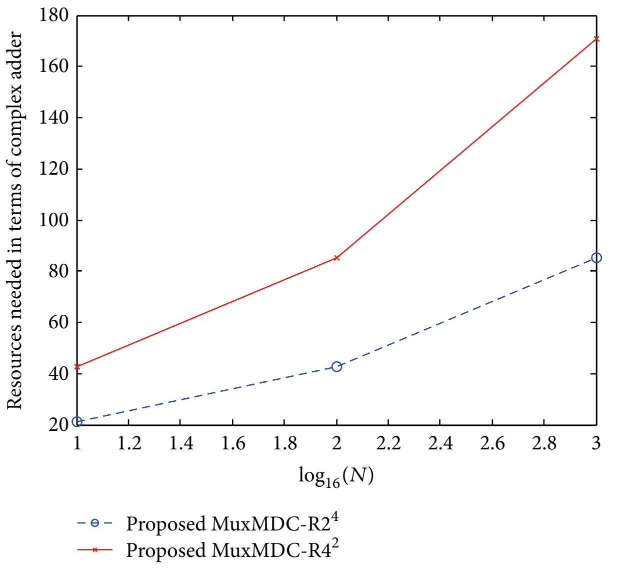

Figure 18: Comparison between the different butterflies' structures in terms of complex adder needed to compute the 16 parallel BPE pipelined FFTs of size N.

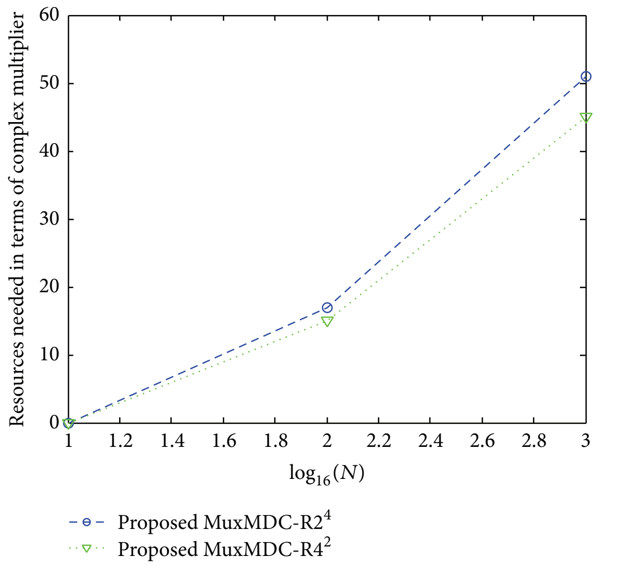

Figure 19: Comparison between the different butterflies' structures in terms of complex multiplier needed to compute the 16 parallel BPE pipelined FFTs of size N.

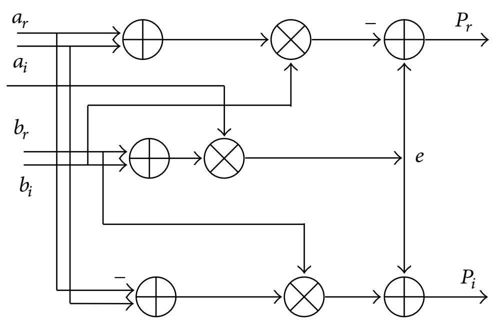

Figure 20: Complex multiplier using three real multipliers and five real adders.

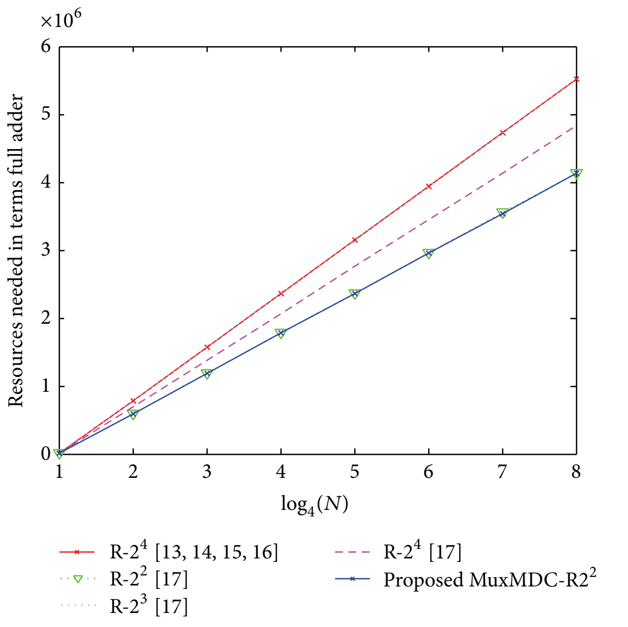

Figure 21: Comparison between the different butterflies' structures in terms of full adder needed to compute the 4 parallel pipelined FFTs of size N (multiplier on 16 bits and adder on 32 bits).

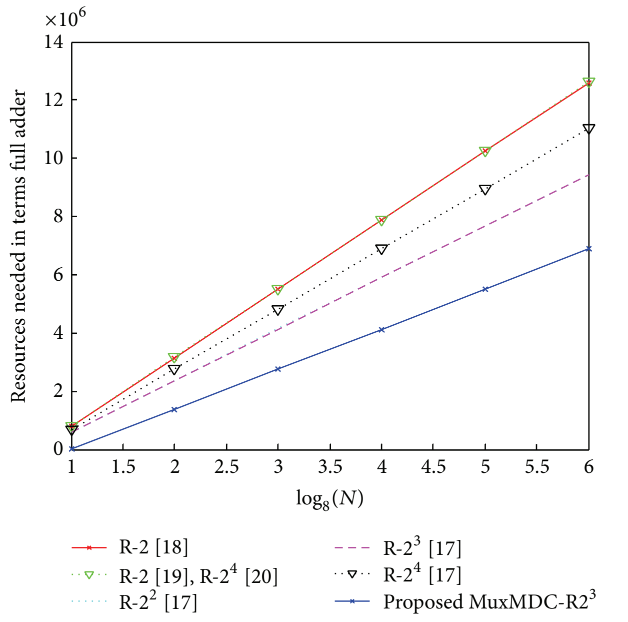

Figure 22: Comparison between the different butterflies' structures in terms of full adder needed to compute the 8 parallel pipelined FFTs of size N (multiplier on 16 bits and adder on 32 bits).

which is highly implemented in the communication systems; see Figure 13.

Since the objective of this paper is mainly concentrated on the higher radices butterflies structures, in our performance study we will be limited to the impact of the butterfly structure. Once the pipeline is filled, the butterflies will produce r output each clock cycle (throughput T in samples per cycle (Spc)). Therefore, Table 1 will draw the comparison between

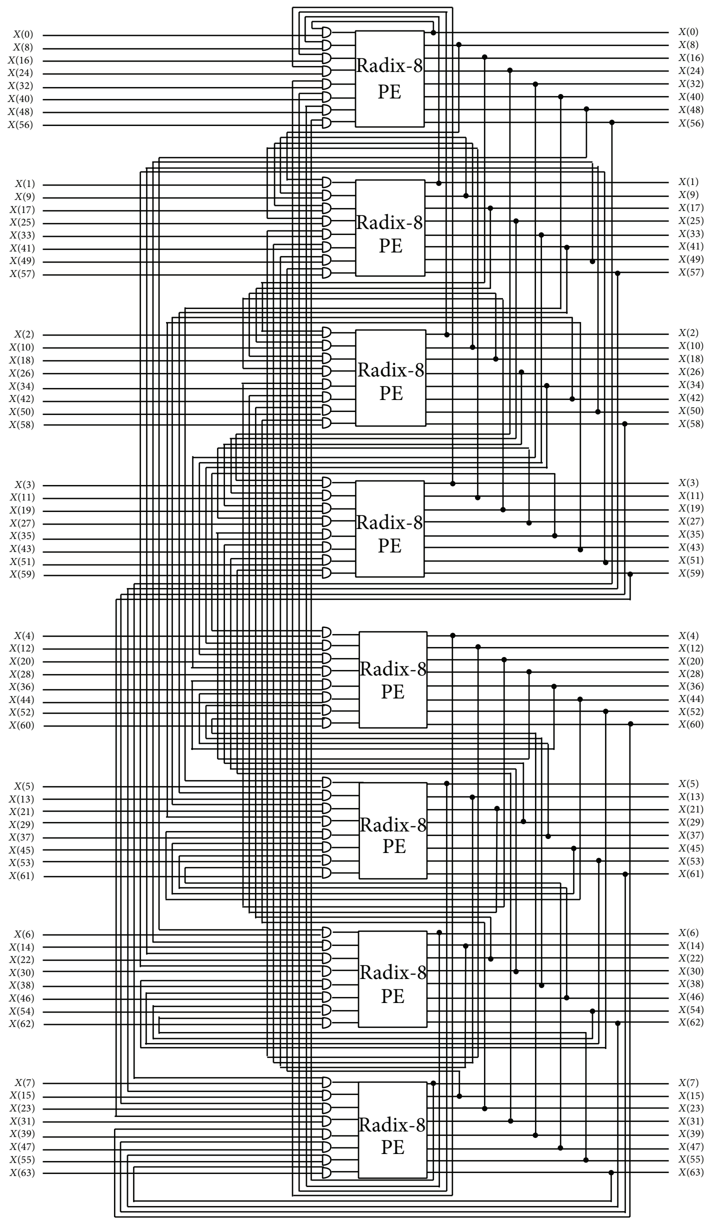

Figure 23: Two stage pipelined FFT (or array structure) with a feedback network [23].

the different butterflies' structures in terms of resources needed to compute an FFT of size N.

As shown in Figure 14, we could clearly see that the proposed MuxMDC-R2^2 for the four parallel pipelined FFTs of size N will have the same amount of complex multiplier compared to the radix 2^4 cited in [30]. Furthermore, our proposed MuxMDC-R2^2 achieves a reduction in the usage of complex multiplier by a factor that ranges between 1.1 and 1.4 compared to the other cited butterflies.

For the 4 parallel pipelined FFTs of size N, the reduction in the usage of complex adder for our proposed method MuxMDC-R2^2 ranges between 1.9 and 3.9 compared to the cited butterflies as shown in Figure 15.

For the 8 parallel pipelined FFTs of size N, the reduction factor in the usage of complex multiplier for our proposed MuxMDC-R2^3 could range from 1.3 to 2.1 compared to the cited butterflies as illustrated in Figure 16.

For the same structure, the reduction factor in the usage of complex adder for our proposed method MuxMDC-R2^3 could range from 3.0 to 5.4 compared to the cited butterflies (Figure 17).

It seems that the proposed MuxMDC-R2^4 uses less complex adders than the proposed MuxMDC-R4^2 as shown in Figure 18 where the proposed MuxMDC-R2^4 achieves a reduction in the usage of complex adder by a factor of 2 but the proposed MuxMDC-R4^2 achieves a reduction in the usage of complex multiplier by a factor of 1.1 as shown in Figure 19.

Since one complex multiplication is counted as 3 real multiplications and 5 real additions as shown in Figure 20, Table 2 will illustrate the required resources in terms of full adder (FA) that will be computed as (a) n^2 for two n-digit real multiplier and (b) p for two p-digit real adder.

For the four parallel pipelined FFTs of size N, it seems that the R-2^2 butterfly cited in [30] will have approximately the same amount of FA as the proposed MuxMDC-R2^2 according to Figure 21. Our proposed MuxMDC-R2^2 will achieve a reduction in the usage of FA by a factor that ranges between 1.17 and 1.34 (Figure 21).

With regard to the eight parallel pipelined FFTs of size N, it seems that the proposed MuxMDC-R2^3 will achieve a reduction in the usage of FA by a factor that ranges between 1.4 and 1.9 in comparison to the other cited butterflies as shown in Figure 22.

Since the implementation of higher radices by means of the radix-$2^\alpha/4^\beta$ butterfly is feasible, the optimal pipelined FFT is achieved by the two stage FFT as shown in Figure 23 where the use of complex memories between the different stages is completely eliminated and the delay required to fill up the pipeline is totally absent.

6. Conclusion

It has been shown that the higher radix FFT algorithms are advantageous for the hardware implementation, due to the reduced quantity of complex multiplications and memory access rate requirements. This paper has presented an efficient way of implementing the higher radices butterflies by means of the radix-$2^\alpha/4^\beta$ kernel where serial parallel models have been represented. The proposed optimized different structures with a scheduling scheme of complex multiplications are suitable for embedded FFT processors. Furthermore, it has been proven that the higher radices butterflies could be obtained by reusing the block circuit diagram of the radix-$2^\alpha/4^\beta$ butterfly. Based on this concept, the hardware resources needed could be reduced which is highly desirable for low power consumption FFT processors. The proposed method is suitable for large pipelined FFTs implementation where the performance gain will increase with an increasing FFTs' radix size. This structure is also appropriate for SIMD implementation on some of the latest DSP cards.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the financial support from the Natural Sciences and Engineering Research Council of Canada and from JABERTECH's Shareholders Trevor Hill from Alberta and Bassam Kabbara from Kuwait.

References

[1] W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965.

[2] P. Duhamel and H. Hollmann, "Split radix FFT algorithm," *Electronics Letters*, vol. 20, no. 1, pp. 14–16, 1984.

[3] S. Winograd, "On computing the discrete Fourier transform," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 73, no. 4, pp. 1005–1006, 1976.

[4] T. Widhe, *Efficient Implementation of FFT Processing Elements*, Linkoping Studies in Science and Technology no. 619, Linkoping University, Linköping, Sweden, 1997.

[5] T. Widhe, J. Melander, and L. Wanhammar, "Design of efficient radix-8 butterfly PEs for VLSI," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '97)*, pp. 2084–2087, June 1997.

[6] J. Melander, T. Widhe, K. Palmkvist, M. Vesterbacka, and L. Wanhammar, "An FFT processor based on the SIC architecture with asynchronous PE," in *Proceedings of the IEEE 39th Midwest Symposium on Circuits and Systems*, vol. 3, pp. 1313–1316, Ames, Iowa, USA, August 1996.

[7] M. Jaber and D. Massicotte, "The self-sorting JMFFT algorithm eliminating trivial multiplication and suitable for embedded DSP processor," in *Proceedings of the 10th IEEE International NEWAS Conference*, Montreal, Canada, June 2012.

[8] M. Jaber, "Butterfly processing element for efficient fast Fourier transform method and apparatus," US Patent No. 6, 751, 643, 2004.

[9] M. A. Jaber and D. Massicotte, "A new FFT concept for efficient VLSI implementation: part I—butterfly processing element," in *16th International Conference on Digital Signal Processing (DSP '09)*, pp. 1–6, Santorini, Greece, July 2009.

[10] Y. Wang, Y. Tang, Y. Jiang, J. Chung, S. Song, and M. Lim, "Novel memory reference reduction methods for FFT implementations on DSP processors," *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 2338–2349, 2007.

[11] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 131–134, May 1998.

[12] S. He and M. Torkelson, "New approach to pipeline FFT processor," in *Proceedings of the 10th International Parallel Processing Symposium (IPPS '96)*, pp. 766–770, April 1996.

[13] E. E. Swartzlander, W. K. W. Young, and S. J. Joseph, "A radix 4 delay commutator for fast Fourier transform processor implementation," *IEEE Journal of Solid-State Circuits*, vol. 19, no. 5, pp. 702–709, 1984.

[14] J. H. McClellan and R. J. Purdy, *Applications of Digital Signal Processing*, Applications of Digital Signal Processing to Radar, chapter 5, Prentice Hall, New York, NY, USA, 1978.

[15] H. Liu and H. Lee, "A high performance four-parallel 128/64-point radix-2^4 FFT/IFFT processor for MIMO-OFDM systems," in *Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS '08)*, pp. 834–837, Macao, China, December 2008.

[16] S.-I. Cho, K.-M. Kong, and S.-S. Choi, "Implementation of 128-point fast fourier transform processor for UWB systems," in *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC '08)*, pp. 210–213, Crete Island, Greece, August 2008.

[17] M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson, "Pipelined radix-2k feedforward FFT architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 23–32, 2013.

[18] J. A. Johnston, "Parallel pipeline fast Fourier transformer," *IEE Proceedings F: Communications, Radar and Signal Processing*, vol. 130, no. 6, pp. 564–572, 1983.

[19] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," *IEEE Transactions on Computers*, vol. 33, no. 5, pp. 414–426, 1984.

[20] S.-N. Tang, J.-W. Tsai, and T.-Y. Chang, "A 2.4-GS/s FFT processor for OFDM-based WPAN applications," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 6, pp. 451–455, 2010.

[21] M. Jaber, "Parallel multiprocessing for the fast Fourier transform with pipeline architecture," US Patent No. 6, 792, 441.

[22] M. A. Jaber and D. Massicotte, "A new FFT concept for efficient VLSI implementation: part II—parallel pipelined processing," in *Proceedings of the 16th International Conference on Digital Signal Processing (DSP 2009)*, pp. 1–5, Santorini, Greece, July 2009.

[23] M. Jaber, "Fourier transform processor," US Patent No. 7, 761, 495.

[24] M. Jaber, "Address generator for the fast Fourier transform processor," US-6, 993, 547 82 and European Patent Application Serial no: PCT/USOI /07602.

[25] E. J. Kim and M. H. Sunwoo, "High speed eight-parallel mixed-radix FFT processor for OFDM systems," in *Proceedings of the IEEE International Symposium of Circuits and Systems (ISCAS '11)*, pp. 1684–1687, Rio de Janeiro, Brazil, May 2011.

[26] P. Li and W. Dong, "Computation oriented parallel FFT algorithms on distributed computer," in *Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming (PAAP '10)*, pp. 369–373, Dalian, China, December 2010.

[27] D. Takahashi, A. Uno, and M. Yokokawa, "An implementation of parallel 1-D FFT on the K computer," in *Proceedings of the IEEE International Conference on High Performance Computing and Communication*, pp. 344–350, Liverpool, UK, June 2012.

[28] R. M. Piedra, "Parallel 1-D FFT implementation with TMS320C4x DSPs," Texas Instruments SPRA108, Digital Signal Processing Semiconductor Group, 1994.

[29] http://www.fftw.org/.

[30] V. Petrov, "MKL FFT performance—comparison of local and distributed-memory implementations," Intel Report, 2012, http://software.intel.com/en-us/node/165305?wapkw=fft.

[31] V. I. Kelefouras, G. S. Athanasiou, N. Alachiotis, H. E. Michail, A. S. Kritikakou, and C. E. Goutis, "A methodology for speeding up fast fourier transform focusing on memory architecture utilization," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 6217–6226, 2011.