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: Dionyssios Reisis

Copyright © 2014 M. A. Jaber and D. Massicotte. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 \( 2^a \) or \( 4^b \) 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 \( 2^a \) or \( 4^b \) 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:

\[
X = B_r x_m,
\]

\[
x_m = [x_{(0)}, x_{(1)}, \ldots, x_{(r-1)}]^T,
\]

\[
X = [x_{(0)}, x_{(1)}, \ldots, x_{(r-1)}]^T,
\]

where \( x_m \) and \( X \) are, respectively, the butterfly’s input and output vectors. \( B_r \) is the butterfly matrix (\( \text{dim}(B_r) = r \times r \)) which can be expressed as

\[
B_r = W_N T_r,
\]

for decimation in frequency (DIF) process, and

\[
B_r = T_r W_N,
\]

for decimation in time (DIT) process. In both cases the twiddle factor matrix, \( W_N \), is a diagonal matrix which is defined by \( W_N = \text{diag}(1, w_N^0, w_N^2, \ldots, w_N^{(r-1)p}) \) with \( p = 0, 1, \ldots, N/r - 1 \) and \( s = 0, 1, \ldots, \lfloor \log N \rfloor - 1 \) and \( T_r \) is the adder tree matrix within the butterfly structure expressed as

\[
T_r = \begin{bmatrix}
        w_N^0 & w_N^0 & w_N^0 & \cdots & w_N^0 \\
        w_N^0 & w_N^{2N/r} & w_N^{2N/r} & \cdots & w_N^{(r-1)N/r} \\
        w_N^{2N/r} & w_N^{2N/r} & w_N^{2N/r} & \cdots & w_N^{(r-1)N/r} \\
        \vdots & \vdots & \vdots & \ddots & \vdots \\
        w_N^{(r-1)N/r} & w_N^{(r-1)N/r} & w_N^{(r-1)N/r} & \cdots & w_N^0
\end{bmatrix}.
\]

As seen from (2) and (3), the adder tree, \( T_r \), 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 \( l \)th line and the \( m \)th column in the matrix \( T_r \) as \( [T_r]_{l,m} \),

\[
[T_r]_{l,m} = w_N^{lmN/r},
\]

where \( l = 0, 1, \ldots, r-1 \) and \( m = 0, 1, \ldots, r-1 \), and \( [x]_N \) represents the operation \( x \) modulo \( N \). By defining \( W_N(m, v, s) \) the set of the twiddle factor matrix as

\[
W_N(m, v, s) = \text{diag}(w_N^{(mN/r)(v/s)}, w_N^{(mN/r)(v/s+1)}, \ldots, w_N^{(mN/r)(v/s+(r-1)v/s)}),
\]

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

\[
W_N(l, s) = \begin{cases}
    w_N^{l/v/r^s} & \text{for } l = m \\
    0 & \text{elsewhere}
\end{cases}
\]

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

\[
W_N(l, s) = \begin{cases}
    w_N^{l/v^{(s-a)}r^{(s-a)}} & \text{for } l = m \\
    0 & \text{elsewhere}
\end{cases}
\]

for the DIT process, where \( l = 0, 1, \ldots, r-1 \) is the \( l \)th butterfly’s output, \( m = 0, 1, \ldots, r-1 \) is the \( m \)th butterfly’s input, and \( [x] \) represents the integer part operator of \( x \).

As a result, the \( l \)th transform output during each stage can be illustrated as

\[
X_{(v,s)[l]} = \sum_{m=0}^{r-1} X_{(v,s)[m]} w_N^{lmN/r + l/v/r^s}.
\]

for the modified DIF process, and

\[
X_{(v,s)[l]} = \sum_{m=0}^{r-1} X_{(v,s)[m]} w_N^{lmN/r + l/v^{(s-a)}r^{(s-a)}}.
\]

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 parallelizing the loops during each stage or iteration in the most successful FFT parallelization was accomplished by each stage (iteration) of the FFT process [26–28]. The degree of parallelism which is multiple of parallelism [31] would result in decreasing the time delay for the FFT butterfly is that they are also useful

& 

\[ X(k) = \sum_{n=0}^{N-1} x(n) w_N^{nk}, \quad k \in [0, N-1], \] 

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

Let \( x_m \) 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]

\[
\begin{align*}
X(k_1+qV/p_r) &= W_N^{k_1} \sum_{n=0}^{N/p_r-1} x(n)w_N^{nk_1+qN/p_r} + W_N^{k_1+qN/p_r} \sum_{n=0}^{N/p_r-1} x(n)w_N^{nk_1+qN/p_r} + \cdots + W_N^{k_1+(p_r-1)qN/p_r} \sum_{n=0}^{N/p_r-1} x(n)w_N^{nk_1+qN/p_r}.
\end{align*}
\]

If \( X(k) \) is the \( N \)th order Fourier transform \( \sum_{n=0}^{N-1} x(n)w_N^{nk} \), then, \( X(0), X(1), \ldots, X(p_r-1) \) will be the \( N/p_r \)

\[
\begin{align*}
X &= W_N^{r-1} \sum_{m=0}^{r-1} x(m)u_N^{\alpha mN/r} = W_N^{r-1} \sum_{m=0}^{r-1} x(m)u_r^m & (13)
\end{align*}
\]

and by applying the concept of the parallel FFT (introduced in Section 3) on the kernel \( B_r \), therefore, (13) will be expressed as

\[
\begin{align*}
X &= W_N \sum_{m=0}^{r-1} x(m)u_r^m \\
&= W_N \left[ \sum_{m=0}^{r/\alpha-1} x(m)u_r^m + \cdots + \sum_{m=0}^{r/\alpha-1} x(m)u_r^m \right] \\
&= W_N \left[ X(0) + u_r^1 X(1) + \cdots + u_r^{(r-1)/\alpha} X^{(1)} \right] \\
&= W_N \left[ X(0) + u_r^1 X(1) \right] \\
for l = 0, \ldots, r - 1.
\end{align*}
\]

It is to be noted that the notation \( u_r \) in all figures of this paper represents the set of twiddle factor associated with the butterfly input defined by \( [u_0, \ldots, u_{r-2}] = \text{diag}(u_N, u_N^{2p}, \ldots, u_N^{(r-1)p}) \).

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

\[
\begin{align*}
X &= W_N \left[ \sum_{m=0}^{1} x(2m)u_2^m + u_1^1 \sum_{m=0}^{1} x(2m+1)u_2^m \right] \\
&= W_N \left[ X(0) + u_1^1 X(1) \right],
\end{align*}
\]

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

\[
\begin{align*}
\end{align*}
\]
VLSI Design

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

![Conventional radix-2 butterfly block diagram](image)

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

![Block circuit diagram of radix-2 adder network](image)

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:

\[ T_{\text{CLK}} > T_{\text{CM}} + T_{\text{CA}} \]

Throughput = \( \frac{4}{T_{\text{CLK}}} \), \hspace{1cm} (16)

where \( T_{\text{CM}}/T_{\text{CA}} \) 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:

\[ t_1 > T_{\text{CM}} + T_{\text{CA}} \]

\[ t_2 > T_{\text{CA}} \]

\[ T_{\text{CLK}} > (t_1 + t_2) > T_{\text{CM}} + 2T_{\text{CA}} \], \hspace{1cm} (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

\[ X(l) = W_N \left[ \sum_{m=0}^{3} x_{2m} w_4^{4m} + w_8 \sum_{m=0}^{3} x_{(2m+1)} w_4^{4m} \right] \]

\[ = W_N \left[ X(0) + w_4^4 X(1) \right], \]

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
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 a = 2), (4) could be expressed as

\[ X_m = W_N \left[ \sum_{m=0}^{7} x_{(2m)} w_{m}^{m} + w_{8}^{l} \sum_{m=0}^{7} x_{(2m+1)} w_{m}^{m} \right] 
= W_N \left[ X_{(0)} + w_{16}^{l} X_{(1)} \right], \]

and the signal flow graph (SFG) of the proposed DIT radix-2^4 MuxMDC-R2^3 based on the partial MuxMDC-R2^2 (Figure 8) is illustrated in Figure 9.
The clock timing of the conventional MDC-R$^4$ BPE is computed as

\[ T_{\text{CLK}} > T_{\text{CM}} + 2t_{\text{pm}} + 4T_{\text{CA}}, \]

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

\[ t_1 > T_{\text{CM}} + T_{\text{CA}}, \]

\[ t_2 > T_{\text{CA}}, \]

\[ t_3 = t_{\text{pm}} + T_{\text{CA}}, \]

Throughput = \( \frac{16}{T_{\text{CLK}}} \). (22)
Figure 11: The proposed DIT MuxMDC-R4.

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

Figure 13: $S$ stages radix-$r$ pipelined FFT.
Resources needed in terms of complex multiplier

\[
R_{-22}^{17}, R_{-24}^{17}, \text{proposed MuxMDC-R}^{22}
\]

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

Resources needed in terms of complex adder

\[
R_{-2}^{18}, R_{-2}^{19}, R_{-2}^{20}, R_{-2}^{24}^{17}, R_{-2}^{3}^{17}, \text{Proposed MuxMDC-R}^{23}
\]

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

The overall timing block diagram of the proposed MuxMDC-R\(2^3\) is sketched in Figure 10.

With the same reasoning as above, we will be limited in the elaboration of the proposed butterfly’s radix-\(4^0\) family to the DIT FFT process.

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

\[
X = W_N \left[ \sum_{m=0}^{3} x_{(4m)} w_{4}^{lm} + \sum_{m=0}^{3} x_{(4m+1)} w_{4}^{lm} + \sum_{m=0}^{3} x_{(4m+2)} w_{4}^{lm} + \sum_{m=0}^{3} x_{(4m+3)} w_{4}^{lm} \right]
\]

\[
= W_N \left[ X_{(0)} + w_{16}^l X_{(1)} + w_{16}^{3l} X_{(2)} + w_{16}^{3l} X_{(3)} \right],
\]

(24)

Figure 16: Comparison between the different butterflies’ structures in terms of complex multiplier needed to compute the 8 parallel BPE pipelined FFTs of size \(N\).
and the proposed MDC-R4 in terms of radix-4 network is illustrated in Figure 11 where the feedback network is for feeding the i th output of the j th radix-4 network to the j th input of the i th 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

FFT s 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.

<table>
<thead>
<tr>
<th>Butterfly structure</th>
<th>Complex multiplier</th>
<th>Complex adder</th>
<th>Latency (cycles)</th>
<th>T (Spc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 parallel BPE architectures</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R-2(^4) [13], [14]</td>
<td>4(log(_N) - 1)</td>
<td>16log(_N)</td>
<td>N/4</td>
<td>4</td>
</tr>
<tr>
<td>R-2(^4) [15]</td>
<td>4(log(_N) - 1)</td>
<td>16log(_N)</td>
<td>N/4</td>
<td>4</td>
</tr>
<tr>
<td>R-2(^4) [16]</td>
<td>4(log(_N) - 1)</td>
<td>16log(_N)</td>
<td>N/4</td>
<td>4</td>
</tr>
<tr>
<td>R-2(^3) [17]</td>
<td>3(log(_N) - 1)</td>
<td>16log(_N)</td>
<td>N/4</td>
<td>4</td>
</tr>
<tr>
<td>R-2(^3) [17]</td>
<td>4(log(_N) - 1)</td>
<td>8log(_N)</td>
<td>N/4</td>
<td>4</td>
</tr>
<tr>
<td>R-2(^2) [17]</td>
<td>3.5(log(_N) - 4)</td>
<td>8log(_N)</td>
<td>N/4</td>
<td>4</td>
</tr>
<tr>
<td>Proposed MuxMDC-R2(^2)</td>
<td>3(log(_N) - 1)</td>
<td>4log(_N)</td>
<td>N/4</td>
<td>4</td>
</tr>
</tbody>
</table>

| 8 parallel BPE architectures |
| R-2 [18] | 8(log\(_N\) - 1) | 16log\(_N\) | N/8 | 8 |
| R-2 [19] | 8(log\(_N\) - 1) | 32log\(_N\) | N/8 | 8 |
| R-2 [20] | 8(log\(_N\) - 1) | 32log\(_N\) | N/8 | 8 |
| R-2\(^3\) [17] | 6(log\(_N\) - 1) | 16log\(_N\) | N/8 | 8 |
| R-2\(^3\) [17] | 6log\(_N\) - 7 | 16log\(_N\) | N/8 | 8 |
| R-2\(^2\) [17] | 7log\(_N\) - 8 | 16log\(_N\) | N/8 | 8 |
| Proposed MuxMDC-R2\(^2\) | 7(log\(_N\) - 1) | 8log\(_N\) | N/8 | 8 |

| 16 parallel BPE architectures |
| Proposed MuxMDC-R2\(^4\) | 17(log\(_N\) - 1) | 16log\(_N\) | N/16 | 16 |
| Proposed MuxMDC-R4 \(^2\) | 15(log\(_N\) - 1) | 32log\(_N\) | N/16 | 16 |

<table>
<thead>
<tr>
<th>Butterfly structure</th>
<th>FA</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 parallel BPE architectures</td>
<td></td>
</tr>
<tr>
<td>R-2(^4) [13], [14]</td>
<td>12n(^2) (log(_N) - 1) + 20 (plog(_N) - 1) + 32plog(_N)</td>
</tr>
<tr>
<td>R-2(^4) [15]</td>
<td>12n(^2) (log(_N) - 1) + 20p (log(_N) - 1) + 32plog(_N)</td>
</tr>
<tr>
<td>R-2(^4) [16]</td>
<td>12n(^2) (log(_N) - 1) + 20p (log(_N) - 1) + 32plog(_N)</td>
</tr>
<tr>
<td>R-2(^3) [17]</td>
<td>9n(^2) (log(_N) - 1) + 15p (log(_N) - 1) + 32plog(_N)</td>
</tr>
<tr>
<td>R-2(^3) [17]</td>
<td>12r(^2) (log(_N) - 1) + 20p (log(_N) - 1) + 16plog(_N)</td>
</tr>
<tr>
<td>R-2(^2) [17]</td>
<td>10.5n(^2) (log(_N) - 1) + 17.5p (log(_N) - 1) + 16plog(_N)</td>
</tr>
<tr>
<td>Proposed MuxMDC-R2(^2)</td>
<td>9n(^2) (log(_N) - 1) + 15p (log(_N) - 1) + 8plog(_N)</td>
</tr>
</tbody>
</table>

| 8 parallel BPE architectures |
| R-2 [18] | 24n\(^2\) (log\(_N\) - 1) + 40p (log\(_N\) - 1) + 32plog\(_N\) |
| R-2 [19] | 24n\(^2\) (log\(_N\) - 1) + 40p (log\(_N\) - 1) + 64plog\(_N\) |
| R-2\(^4\) [20] | 24n\(^2\) (log\(_N\) - 1) + 40p (log\(_N\) - 1) + 64plog\(_N\) |
| R-2\(^3\) [17] | 18n\(^2\) (log\(_N\) - 1) + 30p (log\(_N\) - 1) + 32plog\(_N\) |
| R-2\(^3\) [17] | 18n\(^2\) (log\(_N\) - 1) + 30p (log\(_N\) - 1) + 32plog\(_N\) - 21n\(^2\) - 35p |
| R-2\(^2\) [17] | 21n\(^2\) (log\(_N\) - 1) + 35p (log\(_N\) - 1) + 32plog\(_N\) - 24n\(^2\) - 40p |
| Proposed MuxMDC-R2\(^2\) | 21n\(^2\) (log\(_N\) - 1) + 35p (log\(_N\) - 1) + 16plog\(_N\) |

| 16 parallel BPE architectures |
| Proposed MuxMDC-R2\(^4\) | 51n\(^2\) (log\(_N\) - 1) + 85p (log\(_N\) - 1) + 32plog\(_N\) |
| Proposed MuxMDC-R4\(^2\) | 45n\(^2\) (log\(_N\) - 1) + 75p (log\(_N\) - 1) + 64plog\(_N\) |
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
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\textsuperscript{2} for the four parallel pipelined FFTs of size $N$ will have the same amount of complex multiplier compared to the radix 2\textsuperscript{4} cited in [30]. Furthermore, our proposed MuxMDC-R2\textsuperscript{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\textsuperscript{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\textsuperscript{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\textsuperscript{3} could range from 3.0 to 5.4 compared to the cited butterflies (Figure 17).

It seems that the proposed MuxMDC-R2\textsuperscript{4} uses less complex adders than the proposed MuxMDC-R4\textsuperscript{2} as shown in Figure 18 where the proposed MuxMDC-R2\textsuperscript{4} achieves a reduction in the usage of complex adder by a factor of 2 but the proposed MuxMDC-R4\textsuperscript{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 according to Figure 21. Our proposed MuxMDC-R2 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 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-$2^n/4^p$ 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-$2^n/4^p$ 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-$2^n/4^p$ 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


Submit your manuscripts at
http://www.hindawi.com