FPGA Implementation for the Multiplexed and Pipelined Building Blocks of Higher Radix-2^k FFT

Daniel Massicotte, Marwan A. Jaber, Chokri Neili and Messaoud Ahmed Ouameur

Université du Québec à Trois-Rivières, Electrical and Computer Engineering Department
Laboratoire des signaux et systèmes intégrés
{daniel.massicotte, marwan.jaber, chokri.neili, messaoud.ahmed.ouameur}@uqtr.ca

***Abstract* – Fast Fourier transform (FFT) is one of the fundamental processing block used in many signal processing applications (i.e. for orthogonal frequency division multiplexing in wireless telecommunication). Therefore, every proposal to reduced latency, resources or accuracy errors of FFT implementation counts. This paper proposes the implementation of the butterfly processing elements (BPE) where the concept of the radix-*r* butterfly computation has been formulated as the combination of *α* radix-2 butterflies implemented in parallel. An efficient FFT implementation is feasible using our proposed multiplexed and pipelined BPE. Compared to a state-of-the-art reference based on pipelined and parallel structure FFTs, and FPGA based implementation reveals that the maximum throughput is improved by a factor of 1.3 for a 256-point FFT and reach a throughput of 2680 MSps on Virtex-7. The analysis extends to touch on key performance measurements metrics such as throughput, latency and resource utilization.**

I. Introduction

It is a trivial fact that fast Fourier transform (FFT) is at the heart of the newly proposed fifth generation (5G) system wherein OFDM numerology is supported. The support of higher MIMO orders and flexible numerology render the need of an efficient FFT implementation of prime importance. The overall arithmetical operations deployed in the computation of an *N*-point FFT decreases with increasing radix-*r*. However, the butterfly complexity increases in terms of complex arithmetical computation, parallel inputs, connectivity, and number of phases in the butterfly's critical path delay. The higher radix butterfly involves a non-trivial very large scale implementation (VLSI) problem (i.e. increasing butterfly critical path delay), which explains why the majority of FFT implementations are based on radix-2 or 4, due to their low butterfly complexity. The advantage of using a higher radix is summarized by decreasing the number of multiplications and stages to execute an FFT [3]. Fewer attempts to reduce the computational load have failed, due to the added multipliers in the butterfly's critical path for higher radices [3]. As the main contribution of this paper, we study the FPGA based implementation complexity and performance of our method [1]. Based on the higher radices butterfly and the parallel FFT concept, our approach in [1] introduced structures of higher *multiplexed* 2^{α} or 4^{β} butterflies that 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 butterfly structures. We target the Multi-path Delay Commutator (MDC) pipeline architecture that provides low latency and well-balanced Butterfly Processing Element (BPE). Building upon the method in [1], the contributions of the current paper are: (i) Revisit the proposed multiplexed and pipelined BPE upon which (ii) and FPGA implementation is suggested (iii) to thereafter discuss and analyze the performance in terms of throughput, latency and resource utilization in comparison to a state-of-the-art reference design in [2].

The paper is organized as follows: Section 2 revisits the higher radices' butterfly computations. Section 3 the structure of the proposed multiplex BPE. FPGA implementation and performance evaluation are done in Section 4. Finally, Section 5 reports the conclusions.

II. Higher Radices' Butterfly Computation

The basic operation of a radix-*r* processing element (PE) is the so-called butterfly computation in which *r* inputs are combined to give the *r* outputs via the operation [1][7]:

$$\begin{aligned} &\mathbf{X} = \mathbf{B}_r \mathbf{x}, \\ &\mathbf{x} = [x_{(0)}, x_{(1)}, \cdots, x_{(r-1)}]^T, \\ &\mathbf{X} = \left[X_{(0)},\ X_{(1)},\ \ldots,\ X_{(r\text{-}1)} \right]^T, \end{aligned} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{X}$ are the butterfly's input and output vectors respectively, $\mathbf{B}_r$ is the $r \times r$ butterfly matrix which can be expressed as

$$\mathbf{B}_r = \mathbf{W}_N \mathbf{T}_r, \tag{2}$$

for decimation in frequency (DIF) process, where $\mathbf{W}_N$ is a diagonal matrix which is defined by $\mathbf{W}_N = diag\left(1, w_N^p, w_N^{2p}, \ldots, w_N^{(r-1)p}\right)$ with $p = 0,1,\ldots,N/r^s - 1$, $s = 0,1,\ldots,\log_r N - 1$ and $\mathbf{T}_r$ is the *adder-tree* matrix within the butterfly structure [3]. A straightforward implementation of the adder-tree is not efficient for higher radices butterflies due to the added complex multipliers in the higher radices butterflies' critical path [1].

By defining the element of the l^{th} line and the m^{th} column in the matrix $\mathbf{T}_r$ as $\left[T_r\right]_{l,m}$:

$$\left[\mathbf{T}_r\right]_{l,m} = w_N^{\left[\!\left[(lmN/r)\right]\!\right]_N}, \tag{2}$$

where l=0,1,…,r-1, m=0,1,…,r-1 and $\left[\!\left[\mathrm{x}\right]\!\right]_N$ represents the operation x modulo N.

$\mathbf{W}_{N\,(m,v,s)}$ is the set of the twiddle factor matrix defined as

$$\left[\mathbf{W}_N\right]_{l,m(v,s)} = \mathrm{diag}\left(w_{N\,(0,v,s)}, w_{N\,(1,v,s)}, \ldots, w_{N\,(r-1,v,s)}\right) \quad (4)$$

where the indices r is the FFT's radix, $v = 0,1,\ldots,V-1$ represent the number of words of size r ($V = N/r$) and $s = 0,1,\ldots,S$ is the number of stages (or iterations $S = \log_r N - 1$). Finally, Eq. (2) can be expressed for the different stages in an FFT process as [5]:

$$\left[\mathbf{W}_N\right]_{l,m(v,s)} = \begin{cases} w_N^{\left\langle \left\lfloor v/r^s \right\rfloor l\, r^s \right\rangle_N} & \text{for } l = m \\ 0 & \text{elsewhere} \end{cases} \quad (5)$$

for the DIF process, where l=0,1,…,r–1 is the l^{th} butterfly's output, m=0,1,…,r–1 is the m^{th} butterfly's input and $\lfloor \mathrm{x} \rfloor$ represents the integer part operator of x.

As a result, the l^{th} transform output during each stage can be illustrated as:

$$\mathbf{X}_{(v,s)}[l] = \sum_{m=0}^{r-1} x_{(v,s)}[m]\, w_N^{\left\langle lmN/r + \left\lfloor v/r^s \right\rfloor l\, r^s \right\rangle_N} \quad (6)$$

The conceptual key to the modified radix–r FFT butterfly is the formulation of the radix–r as composed engines with identical structures and the use of a systematic means of accessing the corresponding multiplier coefficients [1]. 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 [5]. For a single processor environment, this type of butterfly with r parallel multipliers would result in decrease in 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. The precedent 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 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.

III. The Radix-2^k Building Blocks and the Pipelined radix-2^k feedforward FFTs architectures

One of the most powerful FFT implementation is the pipelined FFT which is highly implemented in the communication systems such as the Single-path Delay Feedback structure, the Single-path Delay Commutator and the Multi-Path Delay Commutator which will be the target in our performance evaluation. The conventional radix-4 butterfly in terms of radix-2 butterflies is illustrated in Fig. 1.a [6]. In Fig. 1.b shows the multiplexed radix-2^2 (MuxMDC-R2^2) BPE one can clearly see that we used the half of radix-2 butterflies. The use of resources is reduced by multiplexing the radix-2 butterfly. The radix-4 butterfly can be represented in terms of two radix-2 butterflies where the feedback is for feeding outputs of the radix-2 butterflies to the inputs of the radix-2 butterflies having the same label, respectively, and the switches selectively pass

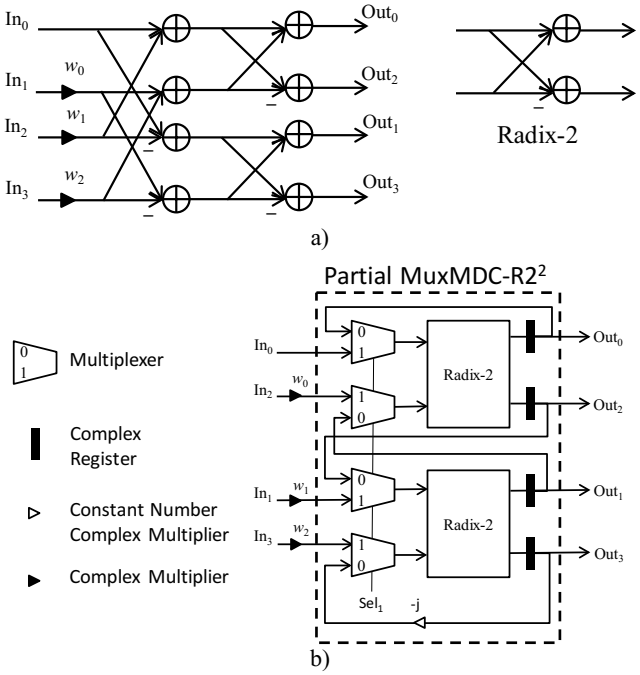

Fig. 1. (a) Signal flow graph of the proposed DIT radix-4 butterfly [1] and (b) Proposed multiplexed radix-2^2 (MuxMDC-R2^2) BPE

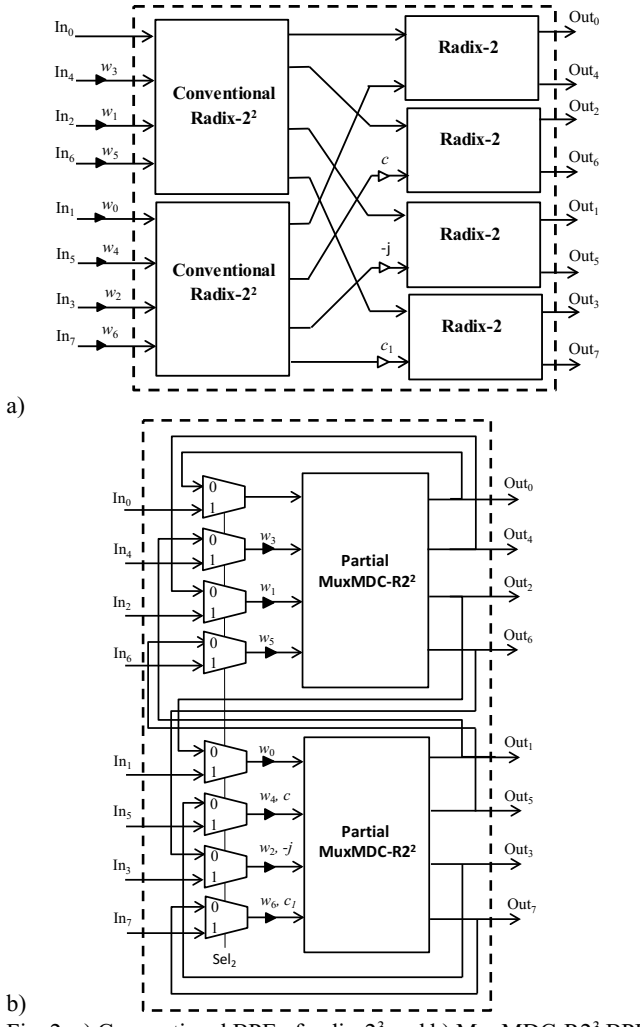

Fig. 2. a) Conventional BPE of radix-2^3 and b) MuxMDC-R2^3 BPE

the input data or the feedback, alternately, to the corresponding radix-2 butterfly. By doing so, the number of complex adders could be reduced from eight to four. For higher radix-*r*, the number of complex multipliers has been reduced [1].

Fig. 2 shows the conventional BPE of radix-2^3 and the MuxMDC-R2^3 BPE with control signals in reference [1]. We can observe only one complex value multiplier in the critical path of the MuxMDC-R2^3 BPE compared with two complex value multipliers in the conventional radix-2^3.

Table 1 reveals the resources needed for the conventional and proposed structure shown in Fig. 1 and 2. We observed that the proposed MuxMDC-R2^3 used 22% and 54% less than conventional number of real multipliers and adders, respectively. Nevertheless, need multiplexers and registers.

IV. FPGA IMPLEMENTATION AND PERFORMANCE EVALUATION

The performance analysis is based on the choice of two crucial elements, which are the complex multiplier and the commutator or multiplexing network. In our implementation, we used a complex value multiplier with four real multipliers and 2 real adders, and in order to maximize the throughput, we pipelined structure with one register in the output of each real operator. Fig. 3 depicts the BPE architecture of the pipelined MuxMDC-R2^2 [1] where one can notice that the critical path is composed of the delay in one real multiplier considering this delay superior to adder or multiplexer. It is worth mentioning that all FFT structures are implemented in fixed point with 16-bit word-length for the input data and twiddle factors.

As for the switching network or commutator, we use the proposed structure in [4]. Furthermore, our comparative study, between the two structures cited in [1] and [2], is based on different criteria such as operational frequency, delay, latency, throughput, resource utilization (e.g. DSP48 block multiplier and logic slices), FFT cost and a metric for overall normalized performance evaluation that reveals the throughput in Mega samples per second per slice (MS/s/Slice). In addition to that our comparative study will be based on three optimization goals; namely timing performance, area reduction and balanced optimization. Due to space limitations only timing performance optimization results are shown. The design environment is the Integrated Synthesis Environment "ISE" from Xilinx software which is used to synthesize and evaluate the designed VHDL code. All FFT executions have been compared with FFT floating point to assure a low error.

Table 1. Resources used for conventional and proposed MuxMDC radix-2^2 and radix-2^3 in number of real multipliers, adders, multiplexers and register.

BPE	Mult.	Add.	Mux	Reg.
Conv. R2^2	12	32	0	0
Conv. R2^3	36	66	0	0
MuxMDC-R2^2	12	14	8	8
MuxMDC-R2^3	28	30	16	16

Table 2 reveals the comparison results using the conventional BPE and the MuxMDC-R2^2 that are performed on the Xilinx's FPGA Virtex-7 (xc7vx330t-ffg1157). By examining Table 2, we can see that the use of the proposed MuxMDC-R2^2 (Fig. 1b) allows to have better performance (MS/s/Slice) compared to conventional BPE coupled with a decrease in the execution time of the FFT. The gain in speed of the proposed architecture in Fig. 1b is due to its multiplexed structure which allows the reuse of the block adders. By doing so, the arithmetical operations within this structure can be executed in a single clock cycle leading to a decrease in the latency metric compared to the conventional structure of BPE.

By focusing on the results for the different synthesis strategies, we have noticed that the major difference is in the number of

Table 2. Radix-2^2 FFTs' implementation comparison on Virtex-7 by using MuxMDC-R2^2 BPE (white area) and Conventional BPE (shaded area) (Strategy: Timing Performance)

BPE	**FFT size**	**Slices**	**DSP48E**	**Delay (ns)**	**Frequency (MHz)**	**Cycles**	**Latency (μs)**	**Execution time**		**Throughput (MS/s)**	**Cost FFT**	**Performance (MS/s/Slice)**
								Cycles	Time (ns)			
Conv.	16	291	12	2.37	422	11	26	4	9	1688	690	5.80
MuxMDC-R2^2	16	319	12	1.49	670	9	13	4	6	2680	478	8.4
Conv.	64	479	24	2.37	422	30	71	16	38	1688	1135	3.52
MuxMDC-R2^2	64	561	24	1.49	671	26	39	16	24	2684	836	4.78
Conv.	256	738	36	2.37	420	85	202	64	152	1680	1757	2.28
MuxMDC-R2^2	256	869	36	1.49	670	79	118	64	96	2680	1297	3.08

Table 3. Performance comparison between the cited references [1] and [2] on Virtex-5 and Virtex-7 (Strategy: Balanced Performance).

	Ref. Architecture [2]				**Proposed Architecture [1]**											
	R2^2MDC (Virtex-5)				**MuxMDC-R2^2 (Virtex-5)**						**MuxMDC-R2^2 (Virtex-7)**					
Size FFT	Slice	Freq. (MHz)	Throu-ghput (MS/s)	MS/s/ Slice	Slice	Freq. (MHz)	Gain in Freq.	Throu-ghput (MS/s)	MS/s /Slice	**Gain**	Slice	Freq (MHz)	Gain in Freq.	Throu-ghput (MS/s)	MS/s /Slice	**Gain**
16	386	458	1831	4.7	383	506	1.11	2028	5.29	1.12	325	673	1.47	2692	8.28	1.76
64	695	384	1536	2.21	749	507	1.32	2028	2.70	1.22	596	673	1.75	2692	4.51	2.04
256	1024	389	1554	1.51	1150	506	1.30	2028	1.76	1.16	852	673	1.73	2692	3.16	2.09

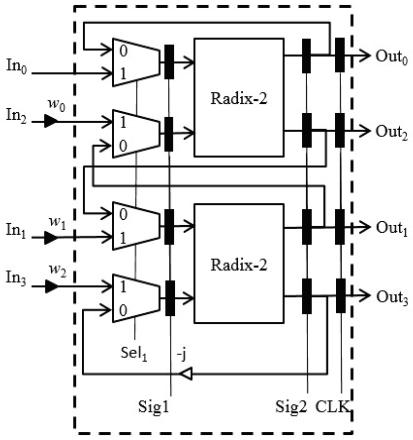

Fig. 3. Pipelined MuxMDC-R2^2 BPE

slices. The number of slices decreases to its minimum in the 'Area reduction' strategy and increases to its maximum for the 'Timing Performance' strategy, but with negligible variation. In fact, the best compromise between the number of slices and frequency can be achieved using the 'balanced' strategy which always gives a maximum frequency with a small increase in the use of slices. Table 3 shows the performance comparison with the cited reference [2] on Virtex-5 (xc5vsx240t-2ff1738) and Virtex-7 for the proposed MuxMDC-R2^2 structure. Our comparative study against is based on the results obtained by choosing the 'balanced' strategy as shown in Table 3 from which we will be concluding the following:

1. The Performance function (MS/s/slice) of our proposed architecture is always higher than the one in [2] which allows us to improve our MS/S/Slice performance by a factor of 1.16 for an FFT of size 256. Furthermore, this improvement is higher on Virtex-7 and could reach 2.09 in terms of performance for the same FFT size.
2. The operational frequency of the proposed architecture is always higher than the one in [2] where the gain in frequency increases by a factor of 1.30 for an FFT of size 256. This gain is even higher on Virtex-7 and can reach 1.73 for the same FFT size.
3. In order to minimize the usage of slices, the method in [2] used an un-pipelined FFT in its structure. Despite the fact we added a lot registers in pipelined FFT structure, we can observe the following two points: (i) the use of slices on Virtex-5 is slightly equivalent to structure in [2] but it is even better on Virtex-7, and (ii) contrarily to the structure in [2], the throughput is independent of the FFT size.

The critical path for the conventional BPE of radix-2^3 presents two complex multipliers and 3 complex adders. On the other hand, as shown in Fig. 4a, the critical path for the MuxMDC-R2^3 BPE has only one complex multiplier, one complex adder and 2 multiplexers. To increase the throughput, we can pipelined this data path as shown in Fig. 3, showing only one multiplexer or one adder in the critical path (Fig 4b). Table 4 compares the implementation of the pipelined MuxMDC-R2^3 versus the conventional Radix-2^3 BPE on the Xilinx's FPGA Virtex-5 where high frequencies can be reached

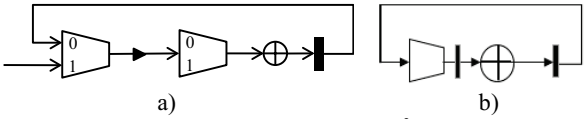

Fig. 4. Critical path for a) the MuxMDC-R2^3 BPE (Fig.2.b) and the pipelined MuxMDC-R2^2 BPE (Fig. 4)

Table 4. Comparison result between the conventional radix-2^3 butterfly and the proposed multiplexed higher radix-2^3 (MuxMDC-R2^3) on the Xilinx's FPGA Virtext-5

BPE	Slices	Frequency (MHz)	DSP48E	Slices Reduction (%)	Gain (in Frequency)
Conventional	430	263	36	-	-
MuxMDC-R2^3	597	506	28	+28%	1.93

with a slight increase in the number of slices and a decrease in the use of DSP48E.

V. Conclusion

The FFT is one of the fundamental processing block used in many signal processing applications and the advent of 5G has an example come with both spectral and energy efficiencies as equally weighted design criteria. This paper implements the structure presented in our paper [1] to further investigate the potential gains against well-cited reference such as [2]. Throughput gains of 1.3 faster for our proposed MuxMDC-R2^2 on FPGA Virtex-5 compared with [2] and similar in terms of slices. This trend is maintained when comparing the normalized throughput to the total number of slices to demonstrate that a substantial increase in the overall throughput is achieved with a minimal increase in the utilized resources.

Acknowledgements

This work has been funded by Natural Sciences and Engineering Research Council of Canada.

References

[1] M Jaber and D. Massicotte "Radix-2$^{\alpha}$/4$^{\beta}$ Building Blocks for Efficient VLSI's Higher Radices Butterflies Implementation," VLSI Design, vol. 2014, Article ID 690594, 13 pages, 2014.

[2] 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.

[3] T. Widhe, "Efficient Implementation of FFT Processing Elements" Linkoping studies in Science and Technology, Thesis No. 619, Linkoping University, Sweden, June 1997.

[4] Jung. Y, Yoon. H, Kim. J, «New Efficient FFT Algorithm Pipeline Implementation Results OFDM/DMT Applications», IEEE Transaction on Consumer Electronics, Vol. 49, No. 1, pp.14-20, February 2003.

[5] M. Jaber, D. Massicotte, "A Novel Approach for FFT Data reordering", International Symposium on Circuits and Systems, Paris, 2010, pp. 1615-1618.

[6] H. M. Saleh and E. Swartzlander, "FFT Implementation with Fused Floating-Point Operations", IEEE Transactions on Computers, No. 2, Vol. 61, February 2012, pp. 284 – 288.

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