Multiplierless Processing Element for Non-Power-of-Two FFTs

Fahad Qureshi, Anastasia Volkova, Thibault Hilaire, Jarmo Takala

Multiplierless Processing Element for Non-Power-of-Two FFTs

Fahad Qureshi, Anastasia Volkova, Thibault Hilaire, and Jarmo Takala

December 21, 2017

Abstract

This paper presents hardware-efficient building blocks for non-power-of-two Fast Fourier transform (FFT) algorithms. A reconfigurable unified multiplierless mixed radix-2/3/4/5 FFT design is proposed. In addition, standalone designs for the computation of the multiplierless radix-3 and radix-5 processing elements are illustrated. These architectures are based on Wingorad Fourier transform algorithm, which uses constant multipliers instead of general complex-valued multipliers. In this paper we propose to further reduce the hardware complexity by replacing the constant multipliers by shift-and-add structures. In addition, we show that the unified architecture is achieved by adding only a small number of multiplexers.

The implementation results for field-programmable gate arrays (FPGAs) with Look-Up Table based logic are provided. In all of them, the proposed designs achieve significant reduction in area (around 30%) with respect to state of the art.

1 Introduction

In today's digital signal processing (DSP) systems, there is often need to transform a signal between time and frequency domains. Fast Fourier transform (FFT) has become one of the most important tools in DSP and a number of algorithms have been proposed for efficient and fast computation of FFT [3, 5, 22, 24].

FFT gained significant attention in communications systems because it is an important operation in orthogonal frequency division multiplexing (OFDM) systems. OFDM is a leading modulation technique, which adequately broaden channel usage and abridge inter-symbol interference along with inter-carrier interference generated by multi path effects. It is mainly used in digital audio broadcasting (DAB), digital video broadcasting-terrestrial (DVB-T) and digital video broadcasting-handheld (DVB-H). In certain OFDM-based communication applications such as 3GPP LTE, FFT is employed to spread the transmitted data over a number of subcarriers.

Traditionally FFTs of size of a power-of-two have been exploited due to simplicity; the design of non-power-of-two FFT processors is challenging as both data management and data processing are irregular [1, 6, 15, 19, 21]. However, there are certain applications, where non-power-of-two FFT sizes are required. For instance, 3GPP LTE

demands various FFT sizes to be supported, e.g., 128, 256, 512, 1024, 1536, 2048, and $12-1296$ [1, 2, 18, 23, 26, 27].

In general, FFT architectures can be classified as memory-based architectures and pipelined architectures [7, 10, 13, 14]. Usually, memory-based FFT architectures provide small area but suffer from a long latency. There are two important challenges during their design: a) conflict-free memory access and b) the design of cores for computation of butterflies (small point DFTs or radices). The design of the cores for a single radix is a relatively simple and well studied topic. However, the design of mixed-radix architectures has some challenges. Usually mixed-radix architectures are used for hardware cost reduction purposes, which is achieved by reusing the adders and multipliers [1, 21, 26]. Pipelined architectures, in turn, are popular in real-time applications, which require high throughput. However, these architectures have larger hardware cost. In a similar fashion, the design of non-power-of-two pipelined architectures is a challenging task.

In this paper, we propose multiplierless processing elements for computing butterfly operations for non-power-of-two FFTs. These processing elements can be used used in both pipelined and memory-based architectures. The proposed processing elements do not require expensive general complex-valued multipliers as they are replaced with constant multipliers realized with inexpensive shift-and-add circuits. The independent architectures of radix-3 and radix-5 can be used in pipelined architectures. The unified reconfigurable processing element can be used in memory based FFT architectures. This paper is extension of [20] where preliminary results were presented. Additionally, we present the implementation on FPGA and compare the proposed architectures to state of the art.

The paper is organized as follows. In section 2, we briefly review the background of the individual architectures of radix-2/3/4/5 FFTs and on the Fixed-Point arithmetic. In section 3 we present the multiplierless architectures for individual radix-3 and radix-5 architectures and for the recondigurable multi-radix FFT. Then, section 4 presents a new modeling for the error propagation through FFTs. In section 5, we compare the hardware cost of the architectures. Finally section 6 concludes the paper.

2 Related Work

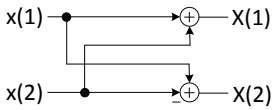
FFT algorithms are based on the approach called divide and conquer. In this approach, the N-point DFT is mapped into several sub-DFTs in such a way that it satisfies the following condition:

$$\sum_{i=1}^{L} Cost\ (\text{sub-DFT}_{N(i)}) + \sum Cost\ (interconnections) \leq Cost\ (\text{DFT}_N), \qquad (1)$$

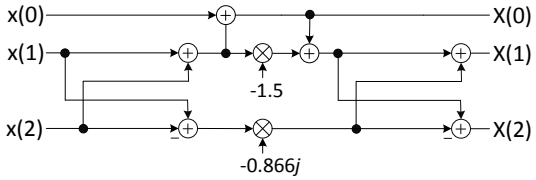
where L is the number of sub-DFTs, Cost (DFT) is the number of operations, Cost (interconnection) is the cost of twiddle factor multiplications and index mapping and DFT_N is the N-point DFT. The power of divide and conquer approach is to apply decomposition recursively, until the sub-DFTs are sufficiently small. This results in reduction of the order of complexity of FFT algorithm. Small DFT's are called the radix-r, where r is the size of small DFT. Generally, the architecture of radix consists of equal

number of hardware components and arithmetic operations, which is also considered as the isomorphic mapping of signal flow graph on the hardware.

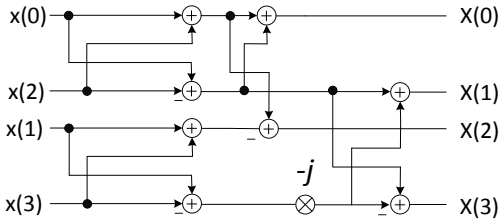
In the following, we discuss the radices of 2, 3, 4, and 5, which can be considered as the most practical cases.



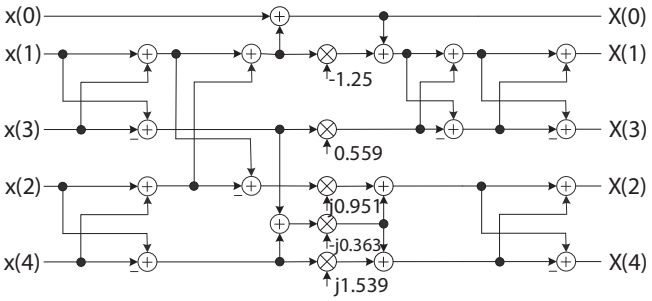
(a) Radix-2



(b) Radix-3



(c) Radix-4



(d) Radix-5

Figure 1: Signal Flow Graphs of radix-2/3/4/5.

2.1 Radix-2/3/4/5 FFTs

The radix-2 FFT can be expressed as;

$$\begin{bmatrix} X(0) \\ X(1) \end{bmatrix} = F_2 \cdot \begin{bmatrix} x(0) \\ x(1) \end{bmatrix}, \text{ and } F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{2}$$

where F_2 is the transform matrix for 2-point DFT and $x(\cdot)$ and $X(\cdot)$ represent the input and output sequences, respectively.

In a similar fashion, the basic building blocks for radix-3, radix-4, and radix-5 FFTs are the corresponding DFTs, i.e.:

$$F_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5 - \frac{\sqrt{3}}{2} j & -0.5 + \frac{\sqrt{3}}{2} j \\ 1 & -0.5 + \frac{\sqrt{3}}{2} j & -0.5 - \frac{\sqrt{3}}{2} j \end{bmatrix}; \tag{3}$$

$$F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix}; \tag{4}$$

$$F_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & \alpha - \beta j & -\gamma - \delta j & -\gamma - \delta j & \alpha + \beta j \\ 1 & -\gamma - \delta j & \alpha + \beta j & \alpha - \beta j & -\gamma + \delta j \\ 1 & -\gamma + \delta j & \alpha - \beta j & \alpha + \beta j & -\gamma - \delta j \\ 1 & \alpha + \beta j & -\gamma + \delta j & -\gamma - \delta j & \alpha - \beta j \end{bmatrix}, \tag{5}$$

with $\alpha = \cos(\frac{2\pi}{5}) = \frac{\sqrt{5}-1}{4}$, $\beta = \sin(\frac{2\pi}{5}) = \sqrt{\frac{5+\sqrt{5}}{8}}$, $\gamma = \cos(\frac{\pi}{5}) = \frac{\sqrt{5}+1}{4}$ and $\delta = \sin(\frac{\pi}{5}) = \sqrt{\frac{5-\sqrt{5}}{8}}$.

The signal flow graphs (SFG) that represent the basic computations of radix-2, radix-3, radix-4, and radix-5 FFTs are shown in Fig. 1. The SFG of radix-2 and radix-4 are based on the direct implementation of DFT algorithm and Cooley-Tukey FFT respectively, whereas the SFG's of the radix-3 and radix-5 are based on Winograd Fourier transform algorithm (WFTA) discussed in the following section.

2.2 Winograd Fourier Transform Algorithm

The Winograd Fourier Transform Algorithm (WFTA) has the minimum number of multiplications at the expense of introducing few extra additions [24]. Although WFTA is very efficient for small prime size DFTs, it becomes quickly unpractical for larger sizes due to the number of additions. In the WFTA, the DFT equation is written as

$$[X(0)....X(N-1)]^T = O \cdot M \cdot I \cdot [x(0)....x(N-1)]^T, \tag{6}$$

where I is a matrix that corresponds to additions between inputs, M is a diagonal matrix with the multiplications and O is a matrix corresponding to additions after the multiplications. These algorithms require multiplication by either real or imaginary parts of a complex number.

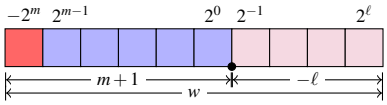


Figure 2: Two's complement fixed-point number. Here $m = 5, \ell = 4$.

Denote $x = x_{re} + jx_{im}$ and $y = y_{re} + jx_{im}$ two complex variables such that $y = cx$, where c is a complex constant.If c is on the unit circle $c = e^{j\theta}$, then the complex multiplication $y = cx$ can be decomposed into real multiplications and additions with:

$$\begin{bmatrix} y_{re} \\ y_{im} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{7}$$

The WFTA is based on these considerations an Based on these considerations, one can transform the classical FFT (3) into the WFTA (6). This way, WFTA requires only a semi-complex multiplier for multiplication with complex data and provides two times smaller hardware cost.

2.3 Fixed-Point Arithmetic

Ideally, arithmetic operations in thr FFT algorithm should be represented with infinite precision. However, in practice one disposes of memory registers only with a finite capacity. There exist different ways of approximation of real numbers using finite number of digits. One of the most common ways for embedded systems is the radix-2 two's complement Fixed-Point (FxP) arithmetic [4].

Two's complement FxP number system is a subset of signed real numbers whose elements are w-bit integers scaled by a fixed factor and have form

$$t = \pm T \cdot 2^\ell, \tag{8}$$

where $T \in [-2^{w-1}; 2^{w-1} - 1] \cap \mathbb{Z}$ is an integer mantissa and 2^ℓ is an implicit quantization factor. In binary representation, t is written as

$$t = -2^m t_m + \sum_{i=\ell}^{m-1} 2^i t_i, \tag{9}$$

where t_i is the i^{th} bit, and m and ℓ are the positions of the most and the least significant bits of t, respectively (see Fig. 2).

Obviously, not every real number is exactly representable with a given wordlength and, usually, a rounding must be performed. Without loss of generality we assume to be operating over numbers c such that $|c| < 1$, i.e. the most significant bit position is $m = 0$. Then, given a real number c and wordlength w, the FxP counterpart of c is computed via $c_q = \lfloor c \cdot 2^{w-1} \rceil \cdot 2^{1-w}$, where $\lfloor\ \rceil$ is the round-to-nearest operator. For round-to-nearest, the rounding error is $|c - c_q| \leq 2^\ell$.

One of the goals of a FFT system designer is to round the FFT coefficients as much as possible (thus, win space ont he curcuit) while guaranteeing a certain level of noise

that these roundings introduce into the output. The propagation and accumulation of the coefficient rounding errors through the FFT algorithms is taken into account in Section 4.

Other sources of rounding errors are arithmetic operations, e.g. addition and multiplication. Usually, when performing an operation over two FxP numbers, an intermediate result is computed with some extended precision (e.g. by adding guard bits) and then rounded to the target output format. Ideally, one wishes to construct operators that will behave as if the result was computed exactly and then rounded only once. The rounding error will depend on the number of guard bits and the rounding mode that is used. While the smallest error is provided by the round-to-nearest operator, we will use a truncation, which has smaller hardware cost and yields a rounding error Δ bounded by $0 \leq \Delta < 2^{\ell}$.

In Section 4 we provide a new model of the computational errors that occur in the FFT algorithm.

3 Proposed Architecture

The architectures for radix-2/3/4/5 FFTs can be obtained by direct implementation of the SFG from Fig. 1. This would which require a number of hardware components equal to arithmetic operations. However, one can reduce the hardware cost by replacing the constant multiplications by shift-and-add circuits. In the following section we propose new multiplierless designs for the standalone radix-3 and radix-5 WFTAs. Then, we combine those designs and present a reconfigurable radix-2/3/4/5 FFT.

3.1 Multiplierless Radix-3/5 WFTA Architecture

A number of multiplications by a constant is used in radix-3/5 WFTA architecture. It is possible to use shift-and-add circuit to efficiently implement the multiplication instead of a general complex-valued multiplier. The design of shift-and-add circuit depends on the selection of coefficients and shift-and-add implementation. Traditionally, the coefficients are obtained by the rounding of pre-computed constants; however, the canonical signed digit representation is used to reduce the number of non-zero digits with respect to the simple binary representation, which reduces the number of adders [16, 17]. Further simplification is achieved by exploiting the redundancy in the single constant multiplication [11, 25]. Additionally, improvement in accuracy and complexity can be achieved by addition-aware coefficient quantization method [12]. With this technique, the constant is not simply rounded to nearest but may be slightly perturbated to find the trade-off between the number of required adders and the rounding error.

By using the techniques proposed in [9, 11, 16, 17, 25], we obtain a number of candidate coefficients whose fractional bits range are tabulated in Table 1. The selection is based on specifications like adder cost, which has relationship with the coefficient quantization error ε. For implementation and comparison, we chose 8-bit coefficients, which have the minimum number of adders; however, the length can be increased according to accuracy requirements of the design at hand. The circuits can be changed according to the selection of the word length.

For instance, the 8-bit coefficients circuit can be expressed as:

$$\begin{aligned} 320X &= ((X\cdot 4+X)\cdot 64);\\ 143X &= (((X\cdot 8+X)\cdot 16)-X);\\ 243X &= (X\cdot 256-(X\cdot 16-(X\cdot 2+X)));\\ 393X &= ((X\cdot 2+X)\cdot 128+(X\cdot 8+X));\\ 93X &= ((X\cdot 4-X)\cdot 8-(X\cdot 4-X)). \end{aligned} \tag{10}$$

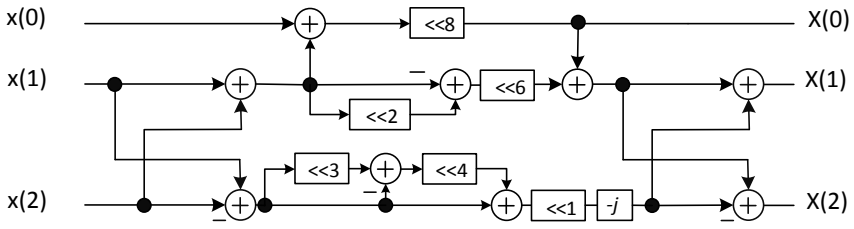
Usually, the shifts are free of cost as they are hardwired. The final multiplierless architectures of radix-3/5 are illustrated in Fig. 3.

Table 1: Different non-trivial coefficients and total number of adders to realize the multiplierless structure for radix-5.

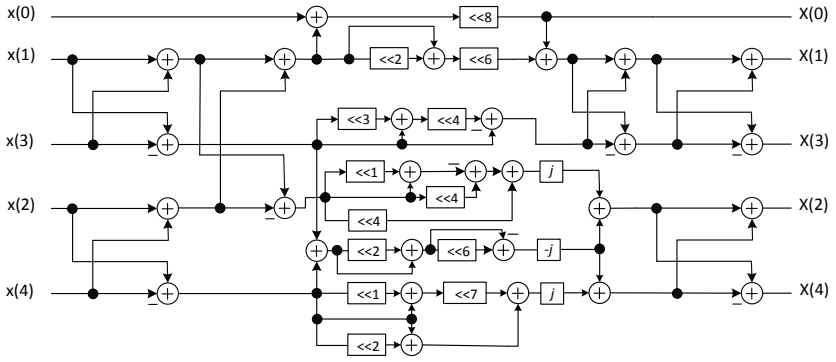
Fractional Bits	Coefficients	Number of Adders
8	$\frac{320}{256}, \frac{143}{256}, \frac{243}{256}, \frac{394}{256}, \frac{93}{256}$	11
9	$\frac{640}{512}, \frac{286}{512}, \frac{487}{512}, \frac{788}{512}, \frac{186}{512}$	11
10	$\frac{1280}{1024}, \frac{572}{1024}, \frac{974}{1024}, \frac{1576}{1024}, \frac{372}{1024}$	11
11	$\frac{2560}{2048}, \frac{1145}{2048}, \frac{1948}{2048}, \frac{3152}{2048}, \frac{774}{2048}$	12
12	$\frac{5120}{4096}, \frac{2290}{4096}, \frac{3896}{4096}, \frac{6303}{4096}, \frac{1488}{4096}$	12
13	$\frac{10240}{8192}, \frac{4579}{8192}, \frac{7791}{8192}, \frac{12606}{8192}, \frac{2976}{8192}$	14
14	$\frac{20480}{16384}, \frac{9159}{16384}, \frac{15582}{16384}, \frac{25212}{16384}, \frac{5952}{16384}$	14
15	$\frac{40960}{32768}, \frac{18318}{32768}, \frac{31164}{32768}, \frac{50425}{32768}, \frac{11904}{32768}$	15

3.2 Multiplierless Reconfigurable Radix-2/3/4/5 FFT

The multiplierless reconfigurable processing element is based on mapping the 2/3/4/5-radix onto a single processing core. We reuse the hardware circuit by multiplexers for computation of 2/3/4/5-radix. The complexity of circuit is equal to radix-5 with a few additional multiplexers. The multiplierless radix-5 WFTA requires the most adders. Further, main challenge is to reduce the number of multiplexers. To obtain this, it is important to find common parts in the signal flow graphs that can be mapped without multiplexers. In addition, multiplexer can be avoided by setting the unused inputs of the circuit to zero, which removes the unnecessary connections of the circuit. Using these techniques, a solution with only nine two-to-one multiplexers controlled with two control signals, has been designed. The resulting architecture is shown in Fig. 4. The input and output relations are tabulated in Table 2, where the dashes denote "don't care" conditions and 0 denotes that the input should be zeroed for proper operations. Finally, signals controlling the multiplexers are shown in Table 3, where dash denotes "don't care" condition.



(a) Radix-3



(b) Radix-5

Figure 3: Multiplierless architectures for radix-3 and radix-5 FFTs.

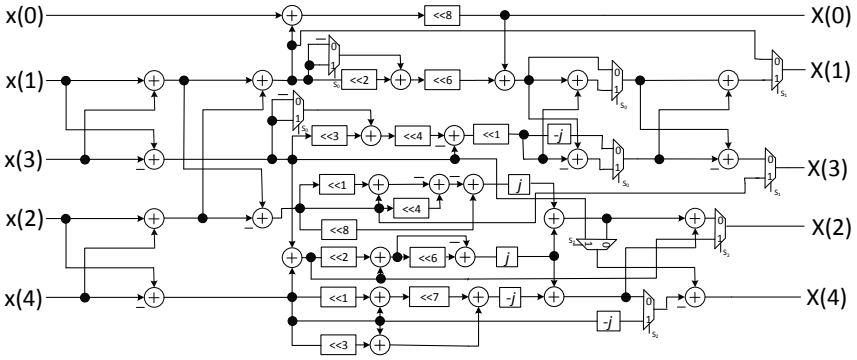


Figure 4: Multiplierless reconfigurable architecture for radix-2/3/4/5 FFT.

Table 2: Inputs and outputs of proposed architecture.

Input configurations					Output configurations				
Index IN	DFT size				Index OUT	DFT size			
	2	3	4	5		2	3	4	5
0	0	x(0)	0	x(0)	0	-	X(0)	-	X(0)
1	x(0)	x(1)	x(0)	x(1)	1	X(0)	X(1)	X(0)	X(1)
2	x(1)	x(2)	x(1)	x(2)	2	X(1)	X(2)	X(1)	X(2)
3	0	0	x(2)	x(3)	3	-	-	X(2)	X(3)
4	0	0	x(3)	x(4)	4	-	-	X(3)	X(4)

Table 3: Control signals to obtain different DFT sizes.

DFT size	s_0	s_1	s_2
Radix-2	-	0	0
Radix-3	0	1	-
Radix-4	-	0	1
Radix-5	1	1	0

4 Error analysis

In this section we give an overview of a new model for the error analysis of FFT algorithms. A classic $N = 2^n$ point FFT algorithm can be represented as

$$[X(0)....X(N-1)]^T = \left(B_n \left(W_{n-1} \left(\dots \left(B_1 \cdot [x(0)\dots x(N-1)]^T\right)\dots\right)\right)\right), \tag{11}$$

where matrices $B_i \in \mathbb{R}^{N\times N}$ express additions between inputs, thus contain only trivial terms $(0,1,-1)$ and $W_i \in \mathbb{C}^{N\times N}$ express the multiplications by twiddle factors.

We propose to represent the computations in (11) as an implicit system of linear equations:

$$\begin{cases} Jt &= Gx \\ y &= Lt \end{cases} \tag{12}$$

where

$$J = \begin{pmatrix} \mathbb{1} & \mathbb{0} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ -W_1 & \mathbb{1} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & B_2 & \mathbb{1} & \mathbb{0} & \mathbb{0} \\ & & & \ddots & \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & W_{n-1} & \mathbb{1} \end{pmatrix}, \; G = \begin{pmatrix} B_1 \\ \mathbb{0} \\ \mathbb{0} \\ \vdots \\ \mathbb{0} \end{pmatrix}, \tag{13}$$
$$L = \begin{pmatrix} \mathbb{0} & \mathbb{0} & \mathbb{0} & \dots & B_4 \end{pmatrix}$$

and $J \in \mathbb{C}^{(2n-2)N\times(2n-2)N}$, $G \in \mathbb{R}^{(2n-2)N\times N}$, $L \in \mathbb{R}^{N\times(2n-2)N}$; and $\mathbb{1}, \mathbb{0}$ are identity and zero matrices respectively.

For instance, for the classic radix-3 WFTA FFT given with (6), we have

$$J = \begin{pmatrix} \mathbb{1} & \mathbb{0} \\ M & \mathbb{1} \end{pmatrix}, \quad G = \begin{pmatrix} I \\ \mathbb{0} \end{pmatrix}, \quad L = \begin{pmatrix} \mathbb{0} & O \end{pmatrix} \tag{14}$$

This representation facilitates the analysis fo the rounding errors that occur during the implementation of FFT algorithms in the following way.

Coefficient rounding errors: with this matrix matrix representation of data-flows, it is straightforward to capture the propagation of coefficient rounding errors. Since all twiddle factors are contained int he matrix J, it is sufficient to assume that instead of J, we actually use $J_q = J + \Delta J$, where ΔJ contains the cefficient rounding errors. Therefore, instead of system (12) we compute

$$\begin{cases} J_q t &= Nx \\ y &= Lt \end{cases} \tag{15}$$

Suppose all non-trivial elements of J are rounded to nearest to w_c bits. Then,

$$|\Delta J_{i,j}| \leq \begin{cases} 2^{-w_c+1}, & \text{if } J_{i,j} \text{ is the non-trivial coefficient} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

Multiplication errors: instead of the exact vector t, we actually compute $\hat{t} := t - \Delta t$, where Δt contains the rounding errors due to multiplications by non-trivial twiddle factors. Therefore, the actually implemented output $\hat{y}$ is computed with

$$\begin{cases} J_q \hat{t} &= Nx + \Delta t \\ \hat{y} &= L\hat{t} \end{cases} \tag{17}$$

Without loss of geenrality, one can assume that the multiplications are performed with last-bit accuracy with an output on w_m bits, i.e.

$$|\Delta t_i| \leq \begin{cases} 2^{-w_m+1}, & \text{if } J_{i,:} \text{ contains non-trivial factor} \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

Overall implementation error: for a given input vector x, the implementation error is bounded by:

$$||y - \hat{y}||_\infty \leq \left|\left| L\left(J^{-1} - J_q^{-1}\right) Nx - LJ_q^{-1}\Delta t \right|\right|_\infty. \tag{19}$$

For instance, for a radix-3 WFTA implemented with w_c-bit coefficients and w_m-bit multipliers, the implementation error is bounded with

$$||y - \hat{y}|| \leq ||Q \cdot x + p||, \tag{20}$$

where

$$Q = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -2^{-w_m+1} & 2^{-w_m+1} \\ 0 & 2^{-w_m+1} & -2^{-w_m+1} \end{pmatrix}, \quad p = \begin{pmatrix} 0 \\ -2^{-w_c+1} \\ 2^{-w_c+1} \end{pmatrix} \tag{21}$$

Analogously, the multiplierless architecture can be analyzed using this matrix representation. It is sufficient to unroll the shift-and-adds operations, as in (10). If the inputs of the algorithms are scaled such that it is guaranteed that no overflow occurs in the intermediate operations, the additions in the multiplierless implementation will be exact and the proposed architecture is even better than the classic implementations with the multipliers. However, in general case, to guarantee the same bound on the accuracy as in the classic algorithms, in the multiplierless architecture one needs to add $\lceil \log_2 k \rceil$ guard bits, where k is the number of additions in the constant multiplication.

5 Results

5.1 Comparison

In this section we compare the proposed multiplierless designs for the radix-3, radix-5 and unified radix-2/3/4/5 multiplierless FFTs with their counterparts [1, 21] that use constant multiplication units. The Table 4 illustrates the complexity comparison in terms of equivalent number of adders. For the semi-complex multipliers we assume that the implementations use Booth encoding, which is equal to 16 adders [8]. In the proposed designs, apart from adder/subtractor circuits, we use some additional multiplexers. We assume that the cost of a multiplexer is equivalent to $1/4$ of the cost of an adder. The Table 4 illustrates that the proposed multiplier-less designs require significantly less adders compared to previously published architectures while guaranteeing analogous accuracy of the output.

Table 4: Hardware complexity (Equivalent number of Adders).

DFT size	Proposed	[1, 21]
Radix-3	9	23
Radix-5	28	97
Mixed radix-2/3/4/5	30.25	99.5

5.2 Implementation

To provide a practical comparison and illustrate the advantages of the proposed architectures, we have described them in VHDL. For the practical implementations, all the architectures are implemented as distributed logic (slices). Post place-and-route results for the Virtex-7 XC7V585tffg1157 FPGA are shown in Table. 5, which includes the area in terms of slices. FOr the architectures containing multipliers, we used constant-coefficient multipliers based on Look-Up Tables. These implementations target inputs and outputs on 36 bits (18 bits for real and 18 for imaginary parts), the non-trivial twiddle factors are rounded to 8 bits, adder/subtractor circuits are standard 18-bit units.

The comparison shows that the proposed architectures reduce by $30-33\%$ the number of slices with respect to previousely known approaches. This implementation is a clear illustration of the gain due to the substitution of constant multipliers by shift-and-add structures.

Table 5: Comparison of different radices implemented on FPGA.

DFT size	Proposed (Slice LUTs)	[1, 21] (Slice LUTs)	Reduction(%)
Radix-3	380	254	33.15
Radix-5	795	1176	32.40
Mixed radix-2/3/4/5	962	1382	30.40

6 Conclusion

This paper presents the multiplierless processing elements for design of the non-power-of-two FFTs. The proposed processing elements are based on Wingorad Fourier transform algorithm. In general, one can avoid implementation of the complex-valued multipliers and reduce the circuit surface by using constant multipliers instead. This paper illustrates that replacment of constant multipliers by shift-and-add structures can bring significant improvements. More precisely, multiplierless processing elements for the radix-3 and radix-5 FFTs were proposed. On top of that, a mutliplierless unified architecture for the radix-2/3/4/5 FFTs is developed. This architecture required only a few additional multiplexers to ensure input/output configurations. Practical implementations showed that in comparison to existing analogous architectures [1, 21] that use multiplier blocks, the multiplierless elements have $\approx 30\%$ gain in terms of slice LUTs.

A new approach on the modeling of the FFT algorithms for the error analysis purposes have been proposed. This new matrix-based model permits an easier and unified approach for the description of the FFT algorithms. We determined the requirements for the multiplierless implementation to have at least the same accuracy as their analogues that use the multiplier block.

In this paper, all FFT coefficients have been rounded to the same wordlength. The next step will be to consider different wordlengths for different twiddle factors. Then, the goal might be to minimize the number of adders in the multiplierless blocks while guaranteeing a certain error bound. Using the proposed modeling, such task can be formalized into a linear programming problem.

Acknowledgment

This work was partially supported by Tekes - the Finnish Funding Agency for Innovation under funding decision 40142/14 (StreamPro) in the FiDiPro program.

References

[1] Chen, J., Hu, J., Lee, S., Sobelman, G.E.: Hardware efficient mixed radix$-25/16/9$ FFT for LTE systems. IEEE Trans. VLSI Syst. **23**(2), 221–229 (2015). DOI 10.1109/TVLSI.2014.2304834

[2] Cho, I., Patyk, T., Guevorkian, D., Takala, J., Bhattacharyya, S.: Pipelined FFT for wireless communications supporting 128-2048 / 1536 -point transforms. In:

Proc. IEEE Global Conf. Signal Inf. Process., pp. 1242–1245. Austin, TX (2013). DOI 10.1109/GlobalSIP.2013.6737133

[3] Cooley, J.W., Lewis, P.A.W., Welch, P.D.: Historical notes on the fast Fourier transform. Proc. of the IEEE **55**(10), 1675–1677 (1967). DOI 10.1109/PROC.1967.5959

[4] Finley, T.: Two's complement. Cornell University lecture notes (2000)

[5] Garrido, M.: A new representation of FFT algorithms using triangular matrices. IEEE Trans. Circuits Syst. I **63**(10), 1737–1745 (2016). DOI 10.1109/TCSI.2016.2587822

[6] Garrido, M.: A new representation of FFT algorithms using triangular matrices. IEEE Trans. Circuits Syst. I **63**(10), 1737–1745 (2016). DOI 10.1109/TCSI.2016.2587822

[7] Garrido, M., Andersson, R., Qureshi, F., Gustafsson, O.: Multiplierless unity-gain SDF FFTs. IEEE Trans. VLSI Syst. **24**(9), 3003–3007 (2016). DOI 10.1109/TVLSI.2016.2542583

[8] Garrido, M., Huang, S.J., Chen, S.G.: Feedforward FFT hardware architectures based on rotator allocation. IEEE Trans. Circuits Syst. I **PP**(99), 1–12 (2017). DOI 10.1109/TCSI.2017.2722690

[9] Garrido, M., Qureshi, F., Gustafsson, O.: Low-complexity multiplierless constant rotators based on combined coefficient selection and shift-and-add implementation (CCSSI). IEEE Trans. Circuits Syst. I **61**(7), 2002–2012 (2014). DOI 10.1109/TCSI.2014.2304664

[10] Garrido, M., Sánchez, M.Á., López-Vallejo, M.L., Grajal, J.: A 4096-point radix-4 memory-based FFT using DSP slices. IEEE Trans. VLSI Syst. **25**(1), 375–379 (2017). DOI 10.1109/TVLSI.2016.2567784

[11] Gustafsson, O., Dempster, A.G., Johansson, K., Macleod, M.D., Wanhammar, L.: Simplified design of constant coefficient multipliers. Circuits, Systems and Signal Processing **25**(2), 225–251 (2006)

[12] Gustafsson, O., Qureshi, F.: Addition aware quantization for low complexity and high precision constant multiplication. IEEE Signal Process. Lett. **17**(2), 173–176 (2010). DOI 10.1109/LSP.2009.2036384

[13] Han, W., Arslan, T., Erdogan, A.T., Hasan, M.: Low power commutator for pipelined FFT processors. In: Proc. IEEE Int. Symp. Circuits Syst., pp. 5274–5277. Kobe, Japan (2005). DOI 10.1109/ISCAS.2005.1465825

[14] He, S., Torkelson, M.: Design and implementation of a 1024-point pipeline FFT processor. In: Proc. IEEE Custom Integrated Circuits Conf., pp. 131–134. Santa Clara, CA (1998). DOI 10.1109/CICC.1998.694922

[15] Huang, S.J., Chen, S.G.: A high-throughput radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15.3c systems. IEEE Trans. Circuits Syst. I **59**(8), 1752–1765 (2012). DOI 10.1109/TCSI.2011.2180430

[16] Jedwab, J., Mitchell, C.J.: Minimum weight modified signed-digit representations and fast exponentiation. Electron. Lett. **25**(17), 1171–1172 (1989). DOI 10.1049/el:19890785

[17] Liu, H., Lee, H.: A high performance four-parallel $128/64$-point radix-2^4 FFT/IFFT processor for MIMO-OFDM systems. In: Proc. IEEE Asia-Pacific Conf. Circuits Syst., pp. 834–837. Macao, China (2008). DOI 10.1109/APCCAS.2008.4746152

[18] Löfgren, J., Liu, L., Edfors, O., Nilsson, P.: Improved matching-pursuit implementation for LTE channel estimation. IEEE Trans. Circuits Syst. I **61**(1), 226–237 (2014). DOI 10.1109/TCSI.2013.2264695

[19] Patyk, T., Qureshi, F., Takala, J.: Hardware-efficient twiddle factor generator for mixed radix-$2/3/4/5$ FFTs. In: Proc. IEEE Workshop Signal Process. Syst., pp. 201–206. Dallas, TX (2016). DOI 10.1109/SiPS.2016.43

[20] Qureshi, F., Ali, M., Takala, J.: Multiplierless reconfigurable processing element for mixed radix-2/3/4/5 FFTs. In: Proc. IEEE Workshop Signal Process. Syst., pp. 1–6 (2017). DOI 10.1109/SiPS.2017.8110007

[21] Qureshi, F., Garrido, M., Gustafsson, O.: Unified architecture for 2, 3, 4, 5, and 7-point DFTs based on Winograd Fourier transform algorithm. Electron. Lett. **49**(5), 348–349 (2013). DOI 10.1049/el.2012.0577

[22] Qureshi, F., Gustafsson, O.: Generation of all radix-2 fast Fourier transform algorithms using binary trees. In: Proc. Europ. Conf. Circuit Theory Design, pp. 677–680. Linköping, Sweden (2011). DOI 10.1109/ECCTD.2011.6043634

[23] Shih, X.Y., Liu, Y.Q., Chou, H.R.: 48-mode reconfigurable design of sdf fft hardware architecture using radix-3 and radix-2 design approaches. IEEE Trans. Circuits Syst. I **64**(6), 1456–1467 (2017). DOI 10.1109/TCSI.2017.2654451

[24] Silverman, H.: An introduction to programming the Winograd Fourier transform algorithm (WFTA). IEEE Trans. on Acoustics, Speech, and Signal Process. **25**(2), 152–165 (1977). DOI 10.1109/TASSP.1977.1162924

[25] Thong, J., Nicolici, N.: Time-efficient single constant multiplication based on overlapping digit patterns. IEEE Trans. VLSI Syst. **17**(9), 1353–1357 (2009). DOI 10.1109/TVLSI.2008.2003004

[26] Wang, A., Bachrach, J., Nikolic, B.: A generator of memory-based, runtime-reconfigurable $2^n3^m5^k$ FFT engines. In: Proc. IEEE Int. Conf. Acoust. Speech Signal Process., pp. 1016–1020. Shanghai, China (2016). DOI 10.1109/ICASSP.2016.7471829

[27] Yu, C., Yen, M.H.: Area-efficient 128-to 2048/1536-point pipeline FFT processor for LTE and mobile WiMAX systems. IEEE Trans. VLSI Syst. **23**(9), 1793–1800 (2015)