Multiplierless Unified Architecture for Mixed Radix–2/3/4 FFTs

Fahad Qureshi, Jarmo Takala, Anastasia Volkova, Thibault Hilaire

Multiplierless Unified Architecture for Mixed Radix$-2/3/4$ FFTs

Fahad Qureshi, Jarmo Takala
Faculty of Computing and Electrical Engineering
Tampere University of Technology
Tampere, Finland
Email: fahad@tut.fi, jarmo.takala@tut.fi

Anastasia Volkova, Thibault Hilaire
Sorbonne Universités, UPMC Univ Paris 06,
UMR 7606, LIP6, F-75005,
Paris, France
Email: anastasia.volkova@lip6.fr, thibault.hilaire@lip6.fr

***Abstract*—This paper presents a novel runtime-reconfigurable, mixed radix core for computation $2-$, $3-$, $4-$ point fast Fourier transforms (FFT). The proposed architecture is based on radix-3 Wingorad Fourier transform, however multiplication is performed by constant multiplication instead of general multiplier. The complexity is equal to multiplierless 3-point FFT in terms of adders/subtractors with the exception of a few additional multiplexers. The proposed architecture supports all the FFT sizes which can be factorized into 2, 3, 4 point. It is also explained that the accuracy of the proposed architecture is not affected due to constant multiplication.**

***Index Terms*—Fast Fourier transform (FFT), Memory-based FFT, Mixed radix**

I. Introduction

Fast Fourier transform is an integral part of orthogonal frequency division multiplexing (OFDM) system, which is used in wide range of communication applications such as 3GPP long term evolution (LTE), wireless personal area network (WPAN), wireless metropolitan area network (WMAN), and other mobile applications. In order to design an FFT processor, various algorithms and architectures have been proposed over the last four last decades. These can be divided into two types, power-of-two and non-power-of-two. Certainly, the design of non-power-of-two FFT processors are more challenging because both data management and data processing is not regular [1]–[5].

Over time, various FFT algorithms and architectures have been proposed to design an efficient FFT processor. In order to meet the requirements of applications, mainly two principal FFT architectures are used: pipelined and memory-based [6]–[9] . In general, memory-based is preferred, as it has several advantages, such as low area or hardware resources occupied by the architecture and low power consumption with complex control logic. However, there are two important challenges during its design: one is the conflict-free memory access and other is the design of core for computation of butterflies (small point FFTs or radices). The design of the core for a single radix is relatively simple and well-known subject. However, the design of multiradix architectures still proposes some challenges. Usually, multiradix architectures are used for hardware cost reduction purposes. This is achieved by reusing the adders and multipliers [1], [3], [10]. These cores are based on mapping the all radices onto computations on the largest radix. However, the radix-2/3/4 FFTs cannot be easily mapped on the radix-4 by the usual approach. . This is due to the fact that radix-2/4 FFTs do not require any multiplication, while radix-3 has one non-trivial multiplication.

In this paper, we propose to replace the general multiplier of the radix-3 FFT by a constant multiplier and then to map the radix-2 and radix-4 on the multiplierless radix-3 architecture. As a result, we propose a core that is multiplierless and can be configured by multiplexers for multiradix purposes. This core can also be used for variable length FFTs, e.g. the 3GPP LTE and LTE require various transform sizes: 128, 256, 512, 1024, 1536, 2048 and $12-1296$ respectively [11]–[13]. The proposed design neither requires the separate hardware for each radix, nor requires the use of costly multiplier. Furthermore its accuracy is at least not worse than the accuracy of the classic implementations.

The paper is organized as follows. We briefly review the background on the individual architectures of 2, 3, and 4-point FFTs in Section II. Then, we explain the main idea behind the multiplier-less multiplication in fixed-point arithmetic in Section II-C. Then, Section III presents the proposed multiplierless architecture. In section Section IV we propose a computer arithmetic based approach on the error analysis of the obtained implementation. Finally Section V provides the main conclusions of paper.

II. Technical Pre-requisites

A. *Radix-$2/3/4$ FFT*

The Cooley-Tukey FFT algorithm [14] is based on the decomposition of a large FFT into two small FFTs. This decomposition is applied recursively until reaches the small point FFTs. The small FFTs are the basic processing elements of the FFT computation, they are often referred to as butterflies or radices.

The radix-2 FFT can be expressed as;

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

where $x(\cdot)$ and $X(\cdot)$ represent the indexes of the input and output sequences respectively.

The radix-3 FFT is the following computation:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -0.5-0.866i & -0.5+0.866i \\ 1 & -0.5+0.866i & -0.5-0.866i \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ x(2) \end{bmatrix}. \quad (2)$$

Finally, radix-4 corresponds to

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \cdot \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}. \quad (3)$$

The Signal Flow Graphs (SFG) that represent the architectures of radix-2, 3, and 4 are shown in Fig. 1. The SFG of radix-2, and 4 are based on the Cooley-Tukey FFT algorithm, whereas the radix-3 SFG is based on Winograd Fourier transform algorithm (WFTA).

B. Winograd Fourier Transform Algorithm

This algorithm has the minimum number of multiplications at the expense of introducing a few extra additions [15]. Although WFTA is very efficient for small prime size FFTs, for larger sizes the number of additions becomes too high for practical implementations. In the WFTA, the FFT computation equation is written as

$$[X(0)....X(N-1)]^T = O \cdot M \cdot I \cdot [x(0)....x(N-1)]^T, \quad (4)$$

where I is a matrix corresponding to additions between inputs, M is a diagonal matrix with the multiplications, and O is a matrix corresponding to additions after the multiplications. The advantages of such architecture is that the multiplication is either real or imaginary.

Denote $c = r + j \cdot i$ to be a complex constant only with real part $r = \cos\alpha, i = 0$, and $x = x_{re} + j \cdot x_{im}$ be a complex variable. Then, the complex output $Y = Y_{re} + Y_{im}$ is computed with

$$\begin{bmatrix} Y_{re} \\ Y_{im} \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 \\ 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} x_{re} \\ x_{im} \end{bmatrix}, \quad (5)$$

where imaginary part is zero. Conversely, multiplication of x by a constant c with only imaginary part $r = 0, i = \sin\alpha$ is described as

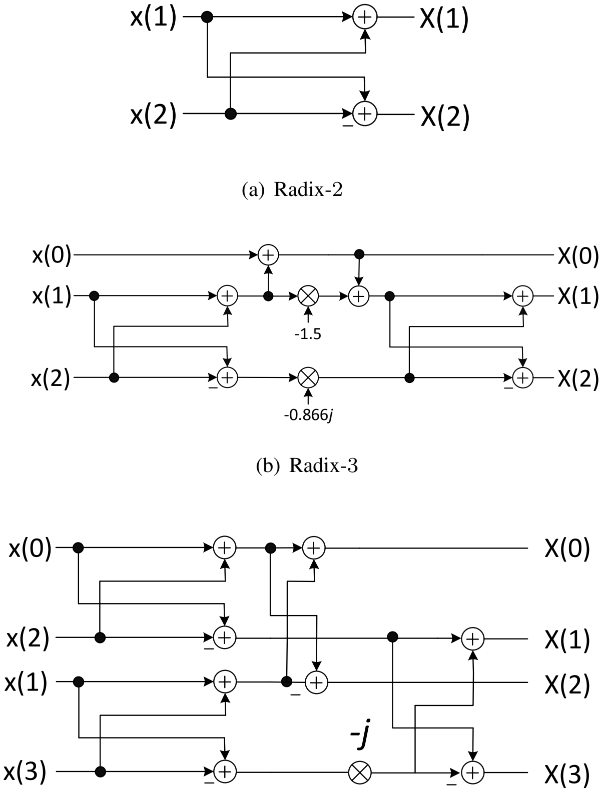
$$\begin{bmatrix} Y_{re} \\ Y_{im} \end{bmatrix} = \begin{bmatrix} 0 & -\sin\alpha \\ \sin\alpha & 0 \end{bmatrix} \begin{bmatrix} x_{re} \\ x_{im} \end{bmatrix}. \quad (6)$$

These considerations are in the basis of the WFTA and permit to transform (2) into (IV-A). Therefore, WFTA requires only a semi-complex multiplier for multiplication with complex data and provides two times smaller hardware cost.

C. Fixed-Point Arithmetic

In exact arithmetic computations such as a semi-complex multiplication would be easy to perform. However, in modern digital systems, usually the memory registers with finite capacity are used. Therefore, finite-precision arithmetic is used to represent values and perform computations.

Consider 2's complement Fixed-Point Arithmetic [16], [17]. In this arithmetic, real numbers are represented as integers scaled by an implicit quantization factor. Such arithmetic benefits from fast bit operations. However, some accuracy concerns are related to such passage to finite precision.



(a) Radix-2

(b) Radix-3

(c) Radix-4

Fig. 1. Signal flow graph of radix-2/3/4.

Let r and i be the real and imaginary part of the complex constant c. In our case $r < 1$ and $i < 1$, i.e. we have only fractional part to be represented. Therefore, if implemented with b bits, the real and imaginary parts are stored in machine as integers $R, I \in z \in [2^{b+1}, \ -2^{b+1} - 1] \cap \mathbb{Z}$, which are obtained by

$$R = \lfloor 2^{b+1} \cdot \cos\alpha \rceil, \quad I = \lfloor 2^{b+1} \cdot \sin\alpha \rceil, \quad (7)$$

where $\lfloor . \rceil$ represents the rounding operation (in our case, it is truncation). Therefore, the Fixed-Point counterparts of r and i are $R \cdot 2^{-b+1}$ and $I \cdot 2^{-b+1}$ respectively. The quantization error ϵ depends on the chosen rounding mode. In case of the truncation, we obtain that

$$-2^{-b+1} \le \epsilon < 0. \quad (8)$$

Another issue that inevitably follows the coefficient quantization is that the additions and multiplications are performed with finite number of bits as well, which may lead to computational errors. Moreover, the propagation of the quantization errors through the computations can sometimes lead to significant changes in the output. This issue will be addressed in Section IV.

TABLE I
NUMBER OF OPERATIONS OF SMALL POINT FFTS.

Size	adds/subs	trivial mults	non-trivial mults
Radix-2	2	0	0
Radix-3	6	0	2
Radix-4	8	1	0

III. PROPOSED ARCHITECTURE

The architecture of radix is obtained by direct implementation of the SFG from Fig. 1, which requires a number of hardware components equal to operations. Tables I and II present the number of operations and hardware cost of each radix respectively. A trivial multiplication does not require multiplier: it can be performed by swapping the real and imaginary part of data. As result, it does not influence the overall hardware cost. It can be seen from the Table II that there is no single radix on which we could to map the rest of the radices: radix-3 is the only one to have a multiplier but it lacks two adders for the radix-3 architecture.

We, on the other side, propose to replace the classic WFTA implementation by the multiplierless version, which introduces additional adders instead of the multiplier, and enables us to map the radix-2/4/FFTs on it.

TABLE II
HARDWARE COMPLEXITY.

Size	# of adder/subtractor	# of multiplier
Radix-2	2	0
Radix-3	6	2
Radix-4	8	0

A. *Multiplierless radix-*3 *WFTA architecture*

A single multiplication by a constant is used in radix-3 WFTA architecture. It is possible to use shift-and-add circuit to efficiently implement the multiplication instead of general multiplier.

Generally, the canonical signed digit representation is used to reduce the number of non-zero digits with respect to the simple binary representation and, therefore the number of adders [18]. Further simplification is achieved by single constant multiplication techniques [19], [20]. They exploit the redundancy in the multiplication by single constant. Additionally, improvement in accuracy and complexity can be achieved by addition-aware coefficient quantization method [21].

Using these techniques proposed in [18]–[21], we obtain a number of candidate coefficients whose fractional bits range are tabulated in Table III.

The selection is based on specifications like adder cost and coefficient quantization error ϵ. It can be observed that three different techniques are required for different coefficient wordlengths in Table III:

- using 2 adders
$$C \cdot A = -1j \cdot ((A \cdot 8 - A) \cdot 16 - A)/1024, \tag{9}$$
- using 3 adders:
$$C \cdot A = -1j \cdot ((A \cdot 8 - x) \cdot 64 - (3 \cdot x + 1))/1024, \tag{10}$$

TABLE III
DIFFERENT QUANTIZATIONS OF THE NON-TRIVIAL COEFFICIENT 0.866 AND NUMBER OF ADDERS TO REALIZE THE MULTIPLIERLESS STRUCTURE.

Fractional Bits	Coefficient	Adders	ϵ
7	$\frac{111}{128}$	2	1.12×10^{-3}
8	$\frac{222}{256}$	2	1.12×10^{-3}
9	$\frac{443}{512}$	3	7.91×10^{-4}
10	$\frac{887}{1024}$	3	1.85×10^{-4}
11	$\frac{1774}{2048}$	3	1.85×10^{-4}
12	$\frac{3547}{4096}$	4	5.86×10^{-5}
13	$\frac{7094}{8192}$	4	5.86×10^{-5}
14	$\frac{14189}{16384}$	4	2.42×10^{-6}
15	$\frac{28378}{32768}$	4	2.42×10^{-6}

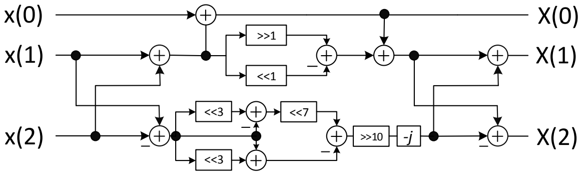


Fig. 2. Multiplierless architecture for radix-3 (Multiplication $[-0.866j], 3$ adders).

- using 4 adders
$$\begin{aligned} C \cdot A = -1j \cdot ((((A \cdot 8 - A) \cdot 4 - (A \cdot 8 - A)) \cdot 8 \\ - ((A \cdot 8 - A) \cdot 4 - (A \cdot 8 - A))) \\ - (A \cdot 8 - A) \cdot 2048)/16384 \end{aligned} \tag{11}$$

We selected the $10 - bit$ coefficient with 3 adders, which has the coefficient quantization error bounded by 1.85×10^{-4}. Usually, the shifts are free of cost as they are hardwired. The final multiplierless architecture is illustrated on the Fig. 2

B. *Multiplierless unified radix-*2/3/4 *FFT*

The multiplierless unified architecture is based on mapping the 2/3/4-radix into a single processing core. First, we design a multiplierless radix-3 FFT architecture. It contains enough computational resources (adders and substractors) to map the radix-2/4, the 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. Further, 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 seven two-to-one multiplexers controlled with two control signals, has been designed. The resulting architecture is shown in Fig. 3. The input and output relations are tabulated in Table IV, 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 V, where dash denotes ”don’t care” condition.

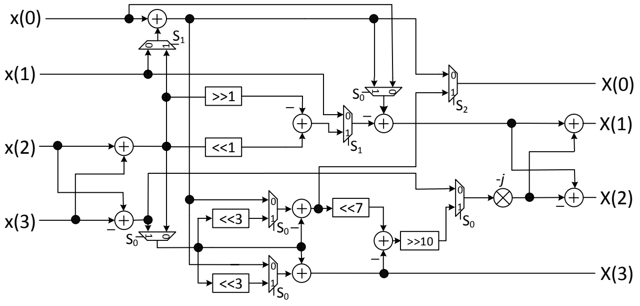


Fig. 3. Multiplierless unified radix-2/3/4 FFT.

TABLE IV
INPUTS AND OUTPUTS OF PROPOSED ARCHITECTURE.

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

IV. ERROR ANALYSIS

Any practical finite precision implementation of the above discussed classical and new FFTs will suffer from finite-precision effects. Usually, for the finite-precision error analysis in signal processing domain a stochastic approach is used: the quantization and computational errors are seen as a white uniformly distributed noise added to the signal. Such a statistical modeling, described in details by Constantinides [22], does not give precise values that the errors can take but provides the average and variance that describe the statistical dispersion of the error.

We, on the other hand, propose to perform accurate error-analysis while exploiting the properties of Fixed-Point Arithmetic. We will model the errors as intervals (tighter than the ones given by statistical approach) and investigate their propagation. In order to analyze the accuracy of any implementation, we need to fully specify the way the arithmetic operations are implemented in hardware. In our implementations we ensure the following:

- Multiplication: our Fixed-Point multiplier, $\otimes$, guarantees that the product of two real numbers x and y, that are represented with b_x and b_y bits respectively, is computed such that the output represented with $b = \min\{b_x, b_y\}$ bits has its error is bounded by
$$|x \cdot y - \texttt{fxp\_mul}(x,y)| < 2^{-b+1}$$
- Addition: we suppose that the input data is scaled such that no over- or under-flow occur in the computations; and the operands will always be in the same format and that no computational errors are introduced.

TABLE V
CONTROL SIGNALS TO OBTAIN DIFFERENT FFT SIZES.

FFT size	s_0	s_1	s_2
Radix-2	0	0	0
Radix-3	1	1	0
Radix-4	0	0	1

A. Classical WFTA radix-3 FFT

The WFTA radix-3 point FFT with a multiplier as shown in Fig. 2(c) can be represented as

$$[X(0)....X(N-1)]^T = B_4 \cdot B_3 \cdot W \cdot \cdot B_2 \cdot B_1 [x(0)....x(N-1)]^T,$$

where matrices $B_i(3\times 3)$ contain only trivial terms $(0, 1, -1)$ and $W(3\times 3)$ contains the non-trivial coefficient. Such computation can also be represented as a system of linear equations:

$$\left\{ \begin{array}{rl} Jt & = Nx \\ y & = Lt \end{array} \right. \tag{12}$$

where

$$J = \begin{pmatrix} -\mathbb{1} & \mathbb{0} & \mathbb{0} & \mathbb{0} \\ B_2 & -\mathbb{1} & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & W & -\mathbb{1} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & B_3 & -\mathbb{1} \end{pmatrix}, \; N = \begin{pmatrix} B_1 \\ \mathbb{0} \\ \mathbb{0} \\ \mathbb{0} \end{pmatrix}, \tag{13}$$
$$L = \begin{pmatrix} \mathbb{0} & \mathbb{0} & \mathbb{0} & B_4 \end{pmatrix}$$

and $J \in \mathbb{C}^{12\times 12}$, $N \in \mathbb{R}^{12\times 3}$, $L \in \mathbb{R}^{3\times 12}$; and $\mathbb{1}, \mathbb{0} \in \mathbb{R}^{3\times 3}$ are identity and zero matrices respectively.

Quantization errors: using the bound (8), we know that instead of J we actually implement the quantized to b bits matrix $J_q := J - \Delta J$, where

$$|\Delta J_{i,j}| \leq \left\{ \begin{array}{ll} 2^{-b+1}, & \text{if } J_{i,j} \text{ is the non-trivial coefficient} \\ 0, & \text{otherwise} \end{array} \right. \tag{14}$$

Therefore, instead of system (12) we compute

$$\left\{ \begin{array}{rl} J_q t & = Nx \\ y & = Lt \end{array} \right. \tag{15}$$

Multiplication errors: consider the above assumptions concerning Fixed-Point multiplication and that the wordlength of non-trivial factor 0.866 is at least equal to the the wordlength of the input. Then, instead of the exact vector t, we actually compute $\hat{t} := t - \Delta t$, where

$$|\Delta t_i| \leq \left\{ \begin{array}{ll} 2^{-b+1}, & \text{if } J_{i,:} \text{ contains non-trivial factor} \\ 0, & \text{otherwise} \end{array} \right. \tag{16}$$

Therefore, the actually implemented output $\hat{y}$ is computed with

$$\left\{ \begin{array}{rl} J_q \hat{t} & = Nx + \Delta t \\ \hat{y} & = L\hat{t} \end{array} \right. \tag{17}$$

Then, for a given input vector $x \in R^3$, the implementation error of the radix-3 WFTA is bounded by:

$$||y - \hat{y}|| \leq \left|\left| L \left(J^{-1} - J_q^{-1} \right) Nx - LJ_q \Delta t \right|\right|, \tag{18}$$

where $||\cdot||_\infty$ is some norm. Thanks to the sparse nature of ΔJ and Δt, by performing the error-free symbolic computations, the error bound (18) can be simplified:

$$||y - \hat{y}|| \leq ||Q \cdot x + p|| \,, \tag{19}$$

with

$$Q = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -2^{-b+1} & 2^{-b+1} \\ 0 & 2^{-b+1} & -2^{-b+1} \end{pmatrix}, \quad p = \begin{pmatrix} 0 \\ -2^{-b+1} \\ 2^{-b+1} \end{pmatrix} \tag{20}$$

Matrix Q represents the impact of the coefficient quantization errors and vector p holds the impact of multiplication errors to the final output.

Remark: Consider the case, when the coefficient 0.866 is quantized to b_c bits, and b_c is less than the wordlength b of the input/output of the FFT algorithm. Then, the bound p will change to

$$p = \begin{pmatrix} 0 \\ -2^{-b_c+1} \\ 2^{-b_c+1} \end{pmatrix}$$

*B. Multiplierless radix-*3 *point FFT*

Similarly to the case of the WFTA radix-3 FFT, our multiplierless design can be represented as a system of linear equations (12) but with slightly different matrix J. The difference between two designs is the use of shifts and adds technique instead of the multiplication. In order to show that the classic and proposed designs have equivalent behavior in terms of errors, consider following reasoning.

Quantization errors: the multiplierless architectures that we proposed in Section III-A are based on coefficients that have already been quantized (error bounds are given in Table III). Therefore, we obtain that modeling (14) holds.

Multiplication errors: if the additions in multiplierless architecture are performed with enough guard bits (it can be shown, that only 2 bits are required), then the error bound (16) is satisfied as well.

Therefore, our modeling of the error propagation stays the same and the new multiplierless design is equivalent to the implementation with multiplier (or better if additions in the multiplierless multiplication are done with care).

V. Conclusion

In this paper, we have presented a multiplierless architecture for the radix-3 WFTA and provided the shift-and-adders specifications for the non-trivial constant quantized to the $6 - 14$ bits, which covers the majority of real-life applications. This radix-3 multiplierless architecture permitted us to map the radix-2 and 4 classical FFTs upon it. The proposed universal architecture requires only a few additional multiplexers controlled by two control signals. Finally, we proposed a new modeling for the Fixed-Point error analysis of the classical WFTA and showed that the new multiplierless design respects the same error bounds.

Acknowledgment

This work is partially supported by Tekes - the Finnish Funding Agency for Innovation under funding decision $40142/14$ (StreamPro) in the FiDiPro program. This work is also partly supported by the Agence nationale de la recherche grant ANR-13-INSE-0007-02 MetaLibm.

References

[1] F. Qureshi, M. Garrido, and O. Gustafsson, “Unified architecture for 2, 3, 4, 5, and 7-point DFTs based on winograd fourier transform algorithm,” *Electron. Lett.*, vol. 49, no. 5, pp. 348–349, February 2013.

[2] T. Patyk, F. Qureshi, and J. Takala, “Hardware-efficient twiddle factor generator for mixed radix-2/3/4/5 FFTs,” in *Proc. IEEE Workshop Signal Process. Syst.*, Oct 2016, pp. 201–206.

[3] J. Chen, J. Hu, S. Lee, and G. E. Sobelman, “Hardware efficient mixed radix−25/16/9 FFT for LTE systems,” *IEEE Trans. VLSI Syst.*, vol. 23, no. 2, pp. 221–229, Feb 2015.

[4] M. Garrido, “A new representation of FFT algorithms using triangular matrices,” *IEEE Trans. Circuits Syst. I*, vol. 63, no. 10, pp. 1737–1745, Oct 2016.

[5] S. J. Huang and S. G. Chen, “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*, vol. 59, no. 8, pp. 1752–1765, Aug 2012.

[6] M. Garrido, “Efficient hardware architectures for the computation of the FFT and other related signal processing algorithms in real time,” Ph.D. dissertation, Universidad Politécnica de Madrid, 2009.

[7] M. Garrido, R. Andersson, F. Qureshi, and O. Gustafsson, “Multiplierless unity-gain SDF FFTs,” *IEEE Trans. VLSI Syst.*, vol. 24, no. 9, pp. 3003–3007, Sept 2016.

[8] M. Garrido, J. Grajal, M. A. Sanchez, and O. Gustafsson, “Pipelined radix-2^k feedforward FFT architectures,” *IEEE Trans. VLSI Syst.*, vol. 21, no. 1, pp. 23–32, Jan 2013.

[9] M. Garrido, M. . Snchez, M. L. Lpez-Vallejo, and J. Grajal, “A 4096-point radix-4 memory-based FFT using DSP slices,” *IEEE Trans. VLSI Syst.*, vol. PP, no. 99, pp. 1–5, 2016.

[10] A. Wang, J. Bachrach, and B. Nikoli, “A generator of memory-based, runtime-reconfigurable $2^n3^m5^k$ FFT engines,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, March 2016, pp. 1016–1020.

[11] I. Cho, T. Patyk, D. Guevorkian, J. Takala, and S. Bhattacharyya, “Pipelined FFT for wireless communications supporting $128 - 2048/1536$ -point transforms,” in *Proc. IEEE Global Conf. on Signal and Information Process.*, Dec 2013, pp. 1242–1245.

[12] Altera, “1536-point FFT for 3GPP long term evolution,” Tech. Rep., Oct 2007, application Note 480.

[13] Sheng-Yeng, Kai-Ting, Chao-Ming, and Yuan-Hao, “Energy-efficient 128-2048/1536-point FFT processor with resource block mapping for 3GPP-LTE system,” in *Proc. IEEE Int. Conf. Green Circuits Syst.*, June 2010, pp. 14–17.

[14] J. Cooley and J. Tukey, “’An algorithm for the machine calculation of complex Fourier series’,” *Math. Comput.*, vol. 19, pp. 297–301, 1965.

[15] S. Winograd, “On computing the discrete Fourier transform,” *Math. Comput.*, vol. 32, pp. 175–199, 1978.

[16] J. V. Newmann, “First draft of a report on the edvac,” Tech. Rep., 1945.

[17] T. Finley, “Two’s complement,” Cornell University lecture notes, 2000.

[18] H. Liu and H. Lee, “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.*, Nov 2008, pp. 834–837.

[19] O. Gustafsson, A. G. Dempster, K. Johansson, M. D. Macleod, and L. Wanhammar, “Simplified design of constant coefficient multipliers,” *Circuits, Systems and Signal Processing*, vol. 25, no. 2, pp. 225–251, 2006.

[20] J. Thong and N. Nicolici, “Time-efficient single constant multiplication based on overlapping digit patterns,” *IEEE Trans. VLSI Syst.*, vol. 17, no. 9, pp. 1353–1357, Sept 2009.

[21] M. Garrido, F. Qureshi, and O. Gustafsson, “Low-complexity multiplierless constant rotators based on combined coefficient selection and shift-and-add implementation (CCSSI),” *IEEE Trans. Circuits Syst. I*, vol. 61, no. 7, pp. 2002–2012, July 2014.

[22] W. L. G. A. Constantinides, Peter Y. K. Cheung, *Synthesis and optimization of DSP algorithms* . Kluwer,, 2004.