Low-Complexity Multiplierless Constant Rotators Based on Combined Coefficient Selection and Shift-and-Add Implementation (CCSSI)

Mario Garrido, *Member, IEEE*, Fahad Qureshi, *Member, IEEE*, and Oscar Gustafsson, *Senior Member, IEEE*

***Abstract*—This paper presents a new approach to design multiplierless constant rotators. The approach is based on a combined coefficient selection and shift-and-add implementation (CCSSI) for the design of the rotators. First, complete freedom is given to the selection of the coefficients, i.e., no constraints to the coefficients are set in advance and all the alternatives are taken into account. Second, the shift-and-add implementation uses advanced single constant multiplication (SCM) and multiple constant multiplication (MCM) techniques that lead to low-complexity multiplierless implementations. Third, the design of the rotators is done by a joint optimization of the coefficient selection and shift-and-add implementation. As a result, the CCSSI provides an extended design space that offers a larger number of alternatives with respect to previous works. Furthermore, the design space is explored in a simple and efficient way. The proposed approach has wide applications in numerous hardware scenarios. This includes rotations by single or multiple angles, rotators in single or multiple branches, and different scaling of the outputs. Experimental results for various scenarios are provided. In all of them, the proposed approach achieves significant improvements with respect to state of the art.**

***Index Terms*—Adder minimization, combined coefficient selection and shift-and-add implementation (CCSSI), complex multiplier, fast Fourier transform, multiple constant multiplication (MCM), rotation, shift-and-add.**

I. Introduction

A ROTATION is a transformation that describes a circular movement with respect to a point. Many digital signal processing algorithms calculate rotations of complex numbers by given angles with respect to the origin. This is the case for the fast Fourier transform (FFT) [1]–[7], the fast discrete cosine transform (DCT) [8], [9] and lattice filters [10], [11].

A rotator is the hardware component used to calculate rotations. There are two main types of rotators: general rotators and constant rotators. General rotators can carry out a rotation by any angle, which is provided as an input to the rotator. They are usually implemented by a complex multiplier or by the coordinate rotation digital computer (CORDIC) algorithm. A complex multiplier typically consists of four real multipliers and two adders. In this case, the rotation is simply calculated by multiplying the input of the multiplier by the rotation coefficient [12]. Conversely, the CORDIC algorithm [13]–[17] is based on breaking down the rotation angle into a series of micro-rotations by specific angles. These micro-rotations are carried out by means of shifts and additions, which are simple to implement in hardware. A review of CORDIC techniques can be found in [16].

Constant rotators calculate rotations by specific angles. They are mainly used to calculate the twiddle factors in FFT architectures [1]–[7]. In constant rotators, the *a priori* knowledge of the rotation angles allows for optimizing the implementation of the rotator. CORDIC-based approaches for constant rotators [18]–[23] are based on selecting stages of the conventional CORDIC [18]–[20] or increasing the amount of micro-rotation angles [21]–[23]. Multiplier-based approaches for constant rotators [4]–[7], [24]–[27] base on techniques to optimize real-valued constant multiplications [4]–[6], trigonometric identities [7], [24], optimization of the coefficient encoding [25], [26], and angle generation by a base-3 system [27].

The design of constant rotators bases on two fundamental elements: the coefficient selection and the shift-and-add implementation. The success of previous approaches is mainly due to an efficient shift-and-add implementation: On the one hand, the techniques used in multiplier-based approaches to implement constant multiplications as shifts and additions are widely developed [6], [28]–[37]. On the other hand, CORDIC-based approaches rely on using elementary angles that allow for an efficient shift-and-add implementation.

However, the coefficient selection has hardly been taken into account. In multiplier-based approaches the coefficients are traditionally obtained by rounding the sine and cosine components of the angle. However, it has been shown that an addition-aware quantization [37] can provide better coefficient. Likewise, the CORDIC elementary angles have been used since the CORDIC algorithm was proposed half a century ago, without questioning if another selection of angles might provide better results. Now, there are results that demonstrate the existence of better angle sets than the CORDIC one for the FFT rotations [27].

This work overcomes the old paradigm for the design of rotators where the main focus was put on the shift-and-add implementation. The new perspective presented in this paper sets the coefficient selection and the shift-and-add implementation as equally important elements in the design of the rotators. This removes the restrictions set by previous approaches to the coefficient selection, widening the amount of alternatives that are explored. This enables optimized solutions that cannot be achieved by using previous approaches.

Manuscript received May 09, 2013; revised October 13, 2013; accepted January 11, 2014. Date of publication March 04, 2014; date of current version June 24, 2014. This paper was recommended by Associate Editor P. K. Meher.

The authors are with the Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, (e-mail: mariog@isy.liu.se; fahadq@isy.liu.se; seoscarg@isy.liu.se).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2014.2304664

The main contributions of this work are:

- It presents a new paradigm for the design of constant rotators that combines the coefficient selection and the shift-and-add implementation in the design process.
- It provides a simple and efficient method to find optimized rotators.
- It can be applied to solve a variety of problems with different demands, including single constant rotation (SCR) and multiple constant rotations (MCR).
- Experimental results for different contexts are provided. In all cases, the proposed approach achieves significant improvements in area and accuracy with respect to the current state of the art.

This paper is organized as follows. Section II introduces the calculation of rotations in fixed-point arithmetic. Section III reviews previous multiplierless rotators. Section IV presents the proposed approach. Section V provides experimental results for the approach in multiple contexts. Finally, Section VI shows the main conclusions.

II. Rotations in Fixed-Point Arithmetic

A rotation of a complex number $x + jy$ by an angle α can be described as

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{1}$$

where $X + jY$ is the result of the rotation. Ideally, the real and imaginary components of the angle, $\cos\alpha$ and $\sin\alpha$, should be represented with infinite precision. However, in digital systems, numbers must be represented with a finite number of bits, which leads to quantization errors. Let C and S be the coefficients that represent $\cos\alpha$ and $\sin\alpha$, respectively. If C and S use b bits in 2's complement, then they can be viewed as integer numbers in the range $[-2^{b-1}, 2^{b-1} - 1]$, i.e., $C, S \in \mathbb{Z}_b$, where

$$\mathbb{Z}_b = \left\{ z \in \mathbb{Z} : -2^{b-1} \leq z \leq 2^{b-1} - 1 \right\}. \tag{2}$$

According to this, a rotation in a digital system can be described as

$$\begin{bmatrix} X_D \\ Y_D \end{bmatrix} = \begin{bmatrix} C & -S \\ S & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{3}$$

where $X_D + jY_D$ is the result of the rotation and C and S are obtained from the rotation angle as [12]

$$\begin{aligned} C &= R \cdot (\cos\alpha + \epsilon_c) \\ S &= R \cdot (\sin\alpha + \epsilon_s), \end{aligned} \tag{4}$$

where ϵ_c and ϵ_s are the relative quantization errors of the cosine and sine components, respectively, and R is the scaling factor. The output $X_D + jY_D$ is also scaled by R.

For constant rotations we can distinguish between a single constant rotation (SCR) and multiple constant rotations (MCR). These cases are explained next.

A. Single Constant Rotation (SCR)

This case refers to a rotation by a single angle, which is shown in Fig. 1(a). In this case, the rotation error is calculated [12] as $\epsilon = \sqrt{\epsilon_c^2 + \epsilon_s^2}$.

Different optimization problems can be defined for SCR depending on the scaling required by the rotator. Thus, the scaling can be *fixed*, *unity* or *arbitrary* depending on the freedom to choose the scaling factor. In fixed scaling, R is fixed to a specific value. Unity scaling is a particular case of fixed scaling in which the rotation has magnitude one or, in more general terms, $R = 2^q$. This is equivalent to considering that the binary point is in a different position of the binary representation. Conversely, arbitrary scaling means that R can take any value, i.e., no restriction is set to R. For arbitrary scaling the approximation error is equal to the angular error only, since R always will take on the optimal value.

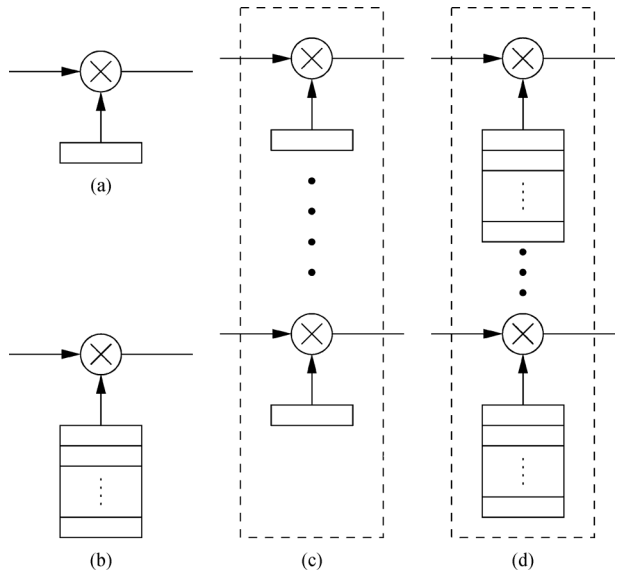


Fig. 1. Hardware layouts. (a) Single branch, single rotation. (b) Single branch, multiple rotations. (c) Multiple branches, single rotation for each branch. (d) Multiple branches, multiple rotations for each branch.

B. Multiple Constant Rotations (MCR)

This case refers to multiple angles that must be optimized together. This joint optimization happens when there is a dependency in the scaling of the angles. Given the angles α_i, $i = 1, \ldots, M$, each angle must be approximated by a complex number $P_i = C_i + jS_i$, where $C_i, S_i \in \mathbb{Z}_b$. The set of complex numbers P_i, $i = 1, \ldots, M$, is called a kernel. The error of a kernel is calculated as the maximum of the errors of the angles [12], i.e., $\epsilon = \max_i(\epsilon(i)) = \max_i\left(\sqrt{\epsilon_c^2(i) + \epsilon_s^2(i)}\right)$, $i = 1, \ldots, M$.

Different optimization problems for MCR can be defined depending on the scaling that is required and on the hardware layout. The scaling for multiple angles is classified based on the relation among the scaling factors of the angles. *Uniform* scaling means that R is the same for all the angles, and *non-uniform* scaling means that different angles may have different scaling factors. Note that in uniform and non-uniform scaling the scaling factor is not fixed from the beginning. Conversely, *fixed* and *unity* set a fixed scaling factor. In this cases, the angles can be treated independently and the problem is reduced to several SCR problems.

Depending on the hardware layout, an MCR problem can target a single rotator that is reconfigured to calculate multiple rotations [Fig. 1(b)], or several rotators in parallel that require the same scaling with one [Fig. 1(c)] or several rotations [Fig. 1(d)] each. The case in Fig. 1(b) is typical in feedback FFT architectures [2]. The case in Fig. 1(c) is typical in fully parallel FFT architectures and in some DCT architectures [9].

Finally, the case in Fig. 1(d) is typical in feedforward FFT architectures [1]. Note that these layouts represent the optimization problem that must be solved, but not the final solution to it, as the rotators will consist of adders and multiplexers instead of the multipliers and memories shown in Fig. 1.

III. Review of Multiplierless Rotators

In the literature, C and S are usually considered as numbers in the range $[-1, 1]$. However, as C and S are quantized to a certain number of bits, we find it more natural to consider them as integers in $\mathbb{Z}_b$, as explained in the previous section. In this section we use this convention to review previous works.

For general rotations, the CORDIC algorithm [13]–[17] breaks down the rotation angle into a series of k micro-rotations by the angles $\alpha_k = \pm \tan^{-1}(2^{-k})$. For each stage, k, the micro-rotation only uses two adders and calculates

$$\begin{bmatrix} X_D \\ Y_D \end{bmatrix} = \begin{bmatrix} 2^k & -\delta_k \\ \delta_k & 2^k \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{5}$$

where $\delta_k \in \{-1, 1\}$ determines the direction of the rotation, and the scaling factor of the stage is $R(k) = \sqrt{2^{2k} + 1}$.

For constant rotators, the extended elementary angle set (EEAS) CORDIC [21] considers the elementary angles $\alpha_k = \tan^{-1}(\delta_k 2^{-a_k} + \gamma_k 2^{-b_k})$, where $\delta_k, \gamma_k \in \{-1, 0, 1\}$ and $a_k, b_k \in \mathbb{N}$. Assuming that $b_k > a_k$ and $c_k = b_k - a_k$, the micro-rotation at stage k is defined by

$$\begin{bmatrix} X_D \\ Y_D \end{bmatrix} = \begin{bmatrix} 2^{b_k} & -(\delta_k 2^{c_k} + \gamma_k) \\ \delta_k 2^{c_k} + \gamma_k & 2^{b_k} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{6}$$

which requires four adders.

In the mixed-scaling-rotation (MSR) CORDIC [22], [23] the number of adders per stage is $2 \cdot (I_k + J_k + 1)$, and each stage calculates a rotation by

$$\begin{bmatrix} X_D \\ Y_D \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{I_k-1} \delta_{ki} 2^{a_{ki}} & -\sum_{j=0}^{J_k-1} \gamma_{ki} 2^{b_{kj}} \\ \sum_{j=0}^{J_k-1} \gamma_{ki} 2^{b_{kj}} & \sum_{i=0}^{I_k-1} \delta_{ki} 2^{a_{ki}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{7}$$

where $\delta_{ki}, \gamma_{ki} \in \{-1, 0, 1\}$ and $a_{ki}, b_{ki} \in \mathbb{N}$. Contrary to the conventional CORDIC, in both EEAS CORDIC and MSR-CORDIC, the scaling depends on the rotation angle. Thus, both approaches present solutions to compensate the scaling.

Other approaches for constant rotations [18]–[20] suggest to select a subset of CORDIC stages to approximate the rotation angle. This reduces both the rotation error and the number of micro-rotation stages.

Another alternative is to consider an elementary angle set that is different to that of the CORDIC. This is done in [27], where all the rotations are generated by combining a small set of FFT angles. This set fits the rotation angles of the FFT better than that of the CORDIC, which results in a reduction in the rotation error, number of adders and latency of the circuit.

Rotators based on techniques to optimize real constant multiplications [4]–[6] follow a different approach. In this case the coefficients C and S are obtained by quantizing $\cos\alpha$ and $\sin\alpha$ to a certain number of bits, b. This is usually done by

$$\begin{aligned} C &= \lfloor 2^b \cos\alpha \rceil \\ S &= \lfloor 2^b \sin\alpha \rceil, \end{aligned} \tag{8}$$

where $\lfloor \cdot \rceil$ represents a rounding operation, leading to

$$\begin{bmatrix} X_D \\ Y_D \end{bmatrix} = \begin{bmatrix} \lfloor 2^b \cos\alpha \rceil & -\lfloor 2^b \sin\alpha \rceil \\ \lfloor 2^b \sin\alpha \rceil & \lfloor 2^b \cos\alpha \rceil \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{9}$$

The multiplication by C and S is implemented as shift-and-add operations. A typical approach is to use the canonical signed digit (CSD) representation [6], [28]. This reduces the number of non-zero digits with respect to the purely binary representation and, therefore, the number of adders. Further simplification is achieved by single constant multiplication (SCM) techniques [30], [31]. They exploit the redundancy in the multiplication by a single constant. Additional reduction in complexity and improvements in accuracy can be obtained by the addition-aware coefficient quantization method [37]. Finally, as the input of a rotator is multiplied by the real and imaginary parts of the coefficient simultaneously, both multiplications can be optimized together [36]. This is done by multiple constant multiplication (MCM) techniques [32]–[35].

Finally, approaches based on trigonometrical identities [7], [24] search for expressions that are shared among the different angles. This results in a simplified rotator that includes a reduced number of adders, multiplexers and real constant multiplications. For instance, $\forall i \in \mathbb{Z}$, any angle $\alpha = i \cdot \pi/8$ can be calculated with real multiplications by only $\cos(\pi/8)$ and/or $\sin(\pi/8)$ [24]. These real constant multiplications are implemented by CSD [7] or SCM [24] techniques.

From the previous discussion we can note that previous approaches restrict the set of coefficients used for the rotations: According to (5), the CORDIC algorithm only calculates rotations by the coefficients $C + jS = 2^k + j\delta_k = 2^k \pm j$. For the EEAS CORDIC in (6) the coefficients only take values $C + jS = 2^{b_k} + j(\delta_k 2^{c_k} + \gamma_k)$. The MSR-CORDIC in (7) only considers values for C and S whose CSD representations have I and J non-zero terms, respectively. And in multiplier-based rotators $C + jS = \lfloor 2^b \cos\alpha \rceil + j\lfloor 2^b \sin\alpha \rceil$, according to (9).

Table I compares previous approaches in terms of coefficient selection and shift-and-add implementation, which defines the design space covered by the approach, i.e., the amount of alternative solutions that it explores. Table I also summarizes the optimization problems that each approach can solve according to Section II, and positions the proposed method, to be elaborated further in the next section.

IV. Proposed Multiplierless Constant Rotators

The proposed approach presents a new perspective to the design of multiplierless rotators. Contrary to previous approaches, the proposed approach does not set any restrictions to $C + jS$ *a priori*. Instead, the selection of the best coefficients $C + jS$ is done as a part of the design process, where it is combined to the shift-and-add implementation (CCSSI).

A. Design Process

The proposed approach can solve SCR and MCR problems. The optimization problem is defined by the angle set, the scaling and the hardware layout. The goal is to obtain coefficients with the smallest rotation error and the smallest number of adders.

Once the optimization problem has been defined, the design method takes as an input the word length of the coefficient search space, b, the maximum allowed error, $\epsilon_{\max}$, and the number of adders allowed to perform the rotations. In case of fixed or unity scaling, the radius, R_{fixed}, is provided. The

TABLE I
COMPARISON OF DIFFERENT APPROACHES TO IMPLEMENT ROTATORS BASED ON SHIFT-AND-ADD OPERATIONS

APPROACH	DESIGN SPACE			OPTIMIZATION PROBLEM	
	Coefficient Selection	Shift-and-Add Optimization	Design Space Size	Scaling	Angle Set
General Rotators: angles not known *a priori*.					
Conventional CORDIC [13]–[15]	Small	High (Direct)	Small	Uniform	General Rotations
Complex Multiplier	Small	Low	Small	Unity	General Rotations
Constant Rotators: angles known *a priori*.					
Lifting Schemes [29]	Small	Medium (CSD)	Small	Unity	SCR
EEAS CORDIC [21]	Medium	High (Direct)	Small	Unity/Arbitrary	SCR
MRS-CORDIC [22], [23]	Large	Medium (CSD)	Medium	Unity	SCR
CORDIC for Fixed Angles [18]–[20]	Medium	High (Direct)	Medium	Unity/Arbitrary	SCR
Trigonometric Identities (CSD) [7]	Medium	Medium (CSD)	Medium	Unity	MCR for FFT
Trigonometric Identities (SCM) [24]	Medium	High (MCM)	Medium	Unity	MCR for FFT
Base-3 Rotator [27]	Medium	High (SCM, MCM)	Medium	Uniform	MCR for FFT
Rotators Using CSD [4], [6]	Small	Medium (CSD)	Small	Unity	MCR
Rotators Using MCM [5]	Small	High (MCM)	Small	Unity	MCR
CCSSI (Proposed)	Maximum (Complete Freedom)	High (SCM, MCM)	Large	Any	SCR and MCR

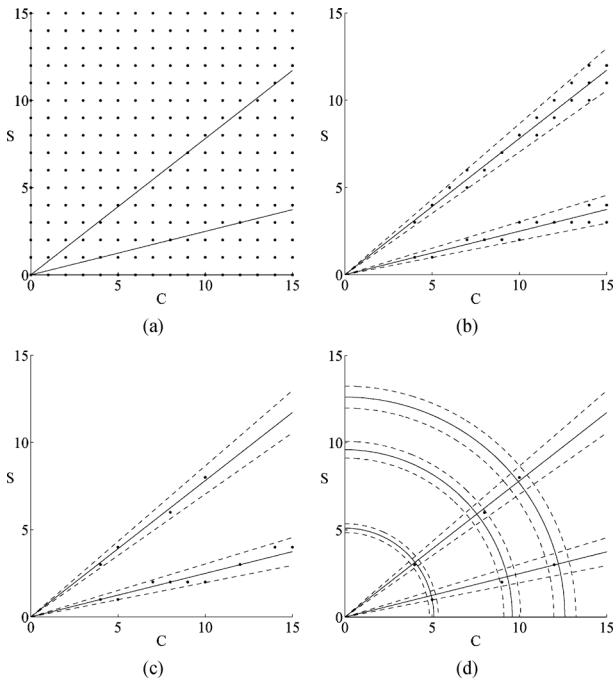


Fig. 2. Overview of the proposed method using the design example. (a) Initial coefficient space and required angles (Step 1). (b) Remaining coefficients after pruning based on the angle (Step 2). (c) Remaining coefficients after pruning based on the number of adders (Step 4). (d) Valid coefficients after forming kernels (Step 5).

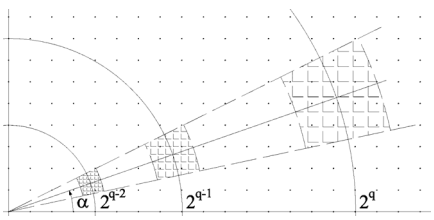


Fig. 3. Coefficient selection for unity scaling.

computation of the rotation error, ϵ, is done as discussed in Section II, while the computation of the number of adders is discussed in Sections IV-B.1 and IV-B2 . When explaining the method, we will in parallel consider the design of two different scenarios: An SCR rotator for the angle $\alpha = 38°$ and an MCR rotator for the angles $\alpha_1 = 14°$ and $\alpha_2 = 38°$. In the latter case, both angles shall have the same scaling. The design examples will be performed using a word length of five bits, i.e., $\mathbb{Z}_5 = \{-16, \ldots, 15\}$ according to (2), a maximum error $\epsilon_{\max} = 5 \times 10^{-2}$, and using at most four adders.

1) Step 1: First, the complete design space, consisting of all possible finite word length values is initialized, as illustrated in Fig. 2(a) for our example. Here, there are 2^{2b-2} different coefficients to consider for each angle, and $(2^{2b-2})^M$ cases for a kernel of M angles.

2) Step 2: Select all possible coefficients that differ at most an angle $\delta = \sin^{-1}(\epsilon_{\max})$ from the required angle(s), i.e.,

$$\left| \tan^{-1}\left(\frac{S_i}{C_i}\right) - \alpha_i \right| < \delta. \tag{10}$$

This is illustrated in Fig. 2(b). Naturally, for the SCR case, only the coefficients approximating 38° will be kept. After this step, the number of alternative coefficients for each angle is reduced to about $(\tan(\delta)/\max(\sin\alpha_i, \cos\alpha_i))\left(2^{2b-2}\right)$.

3) Step 3: If the scaling is fixed, such as for unity scaling, the search space is reduced further by selecting coefficients whose scaling factor is close to R_{fixed}. Any coefficient whose scaling differs more than $R_{\text{fixed}}\epsilon_{\max}$ from R_{fixed} is discarded. This is illustrated in Fig. 3 for the case of unity scaling, i.e., $R_{\text{fixed}} = 2^q$. It should be noted that all coefficients within the 2^{q-1} region will also be present in the 2^q region, although multiplied with a factor of two, which from a shift-and-add perspective is not significant. The number of coefficients in a region is about $4R_{\text{fixed}}^2\epsilon_{\max}^2$.

4) Step 4: The number of adders required to implement each rotation is determined as explained in Section IV-B1. This can be stored in a table for all pairs of C and S to speed up the computation of this step. Coefficients which require more than the allowed number of adders are discarded. The resulting coefficients for the example are illustrated in Fig. 2(c). To the best of the authors knowledge there are no known equations on how many adders are needed on average to realize a coefficient pair as a function of magnitude, which would be required to estimate the number of remaining coefficients after this step.

5) Step 5 – SCR: For the SCR case, the algorithm now has provided a number of candidate coefficients which all are valid based on the specification. Hence, one can directly evaluate the coefficients for $\alpha = 38°$ in Fig. 2(c) to come up with the most suitable coefficient. Typically the one with the smallest rotation error is selected as the number of adders are within the specification boundaries, but different trade-offs can be considered.

TABLE II
STEP 5 – SCR: REMAINING COEFFICIENTS FOR $\alpha = 38°$, ACCORDING TO FIG. 2(C)

$\alpha = 38°$	R	ϵ	Add.
$4 + j3$	5.00	1.97×10^{-2}	4
$5 + j4$	6.40	1.15×10^{-2}	4
$8 + j6$	10.00	1.97×10^{-2}	4
$10 + j8$	12.81	1.15×10^{-2}	4

TABLE III
STEP 5 – MCR: REMAINING KERNELS ACCORDING TO FIG. 2(D)

$\alpha_1 = 14°$	$\alpha_2 = 38°$	R	ϵ	Add.
$5 + j$	$4 + j3$	5.10	4.69×10^{-2}	4
$9 + j2$	$8 + j6$	9.59	4.66×10^{-2}	4
$12 + j3$	$10 + j8$	12.61	1.93×10^{-2}	4

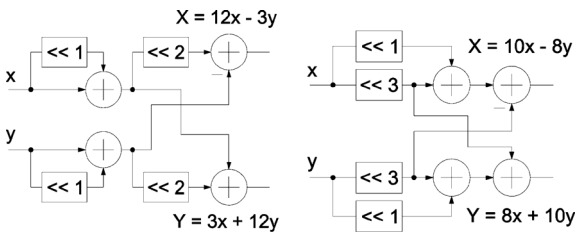


Fig. 4. Realization of the rotators by 14° and 38° from the example in Fig. 2. (a) Rotator by 14° using $12 + j3$. (b) Rotator by 38° using $10 + j8$.

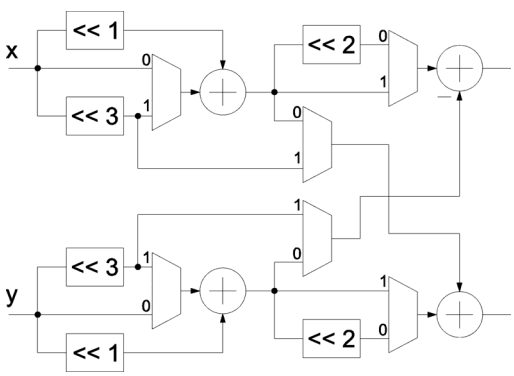


Fig. 5. Rotator that can rotate either 14° or 38° using 4 adders.

The candidates are listed in Table II. It should be noted that there exist power of two multiples of the same coefficient. Hence, for the SCR case it is actually enough to keep coefficients in which at least one part is odd after Step 2 (Step 3 if the scaling is fixed).

6) Step 5 – MCR: For the MCR case, combinations which have approximately the same radius are found. These can be initially pruned on the fact that no two coefficients whose radii differ more than twice the maximum error can form a kernel meeting the specification. For these candidates the maximum error is determined and those meeting the specification are kept. For the example, the remaining candidate coefficients are illustrated in Fig. 2(d). Depending on the hardware layout constraints, as further discussed in Section IV-B2, further pruning can be done. If not, the final set of candidate coefficients are obtained. For the example, these are listed in Table III. Here, it can be noted that some coefficients where both parts are even are used, and, hence, the same simplification that was possible for SCR is not possible. Instead, the corresponding simplification is that in a kernel, at least one of the included coefficients should have an odd part.

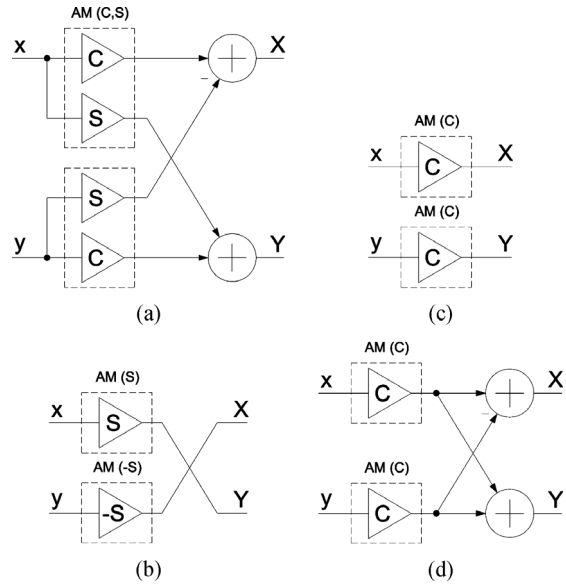


Fig. 6. Structure of the rotator for the cases in Table IV. (a) General case for which $P = C + jS$. (b) Rotation by $P = jS$. (c) Rotation by $P = C$. (d) Rotation by $P = C + jC$.

TABLE IV
ADDER COST OF A ROTATION BY AN ANGLE α

ANGLE	COEFFICIENT	ADDER COST
$\alpha = m \cdot \pi$	$P = C$	$2 \cdot \mathrm{AM}(C)$
$\alpha = m \cdot \pi + \pi/2$	$P = jS$	$2 \cdot \mathrm{AM}(S)$
$\alpha = m \cdot \pi/2 + \pi/4$	$P = C \pm jC$	$2 \cdot \mathrm{AM}(C) + 2$
Other angles	$P = C + jS$	$2 \cdot \mathrm{AM}(C, S) + 2$

7) Step 6: Implement the rotator. For SCR the implementation is straightforward from the shift-and-add realization. This is illustrated for the $\alpha = 38°$ case in Fig. 4(b). For MCR, if several rotation will be mapped to the same rotator, adder sharing, as discussed in Section IV-B2, should be applied. In Fig. 4 the two different realizations for $\alpha_1 = 14°$ and $\alpha_2 = 38°$ are shown, while the merged rotator is shown in Fig. 5.

B. Shift-and-Add Implementation

1) Number of Adders for SCR: The shift-and-add implementation depends on the rotation angle. In general, a rotation by $P = C + jS$ is calculated according to Fig. 6(a) and the total number of adders of the shift-and-add implementation is

$$\mathrm{AR}(P) = 2 \cdot \mathrm{AM}(C, S) + 2 \tag{11}$$

where $\mathrm{AM}(C, S)$ is the number of adders needed to multiply a real number by C and S simultaneously.

If the rotation coefficient is real, i.e., $P = C$, the rotator is reduced to two real multiplications. This case is shown in Fig. 6(c), and the number of adders is

$$\mathrm{AR}(P) = 2 \cdot \mathrm{AM}(C) \tag{12}$$

where $\mathrm{AM}(C)$ is the number of adders needed to multiply by a real number C.

Likewise, if the coefficient is a pure imaginary number, i.e., $P = jS$, the rotation has two real constant multiplications as shown in Fig. 6(b), and the number of adders is

$$\mathrm{AR}(P) = 2 \cdot \mathrm{AM}(S). \tag{13}$$

Finally, if $|C| = |S|$, which is true for angles $\alpha = m \cdot \pi/2 + \pi/4$, the structure of the rotator is shown in Fig. 6(d) and the number of adders is

$$\mathrm{AR}(P) = 2 \cdot \mathrm{AM}(C, C) + 2 = 2 \cdot \mathrm{AM}(C) + 2. \quad (14)$$

These special cases require less adders than the general case in Fig. 6(a). This fact is taken into account in order to make a better use of the adders and design simpler rotators.

According to (11)–(14), the number of adders of a rotation can be obtained directly from the number of adders for a real constant multiplication by C, S or both of them. The adder cost of a rotation is summarized in Table IV as a function of the rotation angle. SCM techniques [30], [31] are used to calculate the adder cost of multiplications by singles real constant represented by $\mathrm{AM}(C)$ and $\mathrm{AM}(S)$, and MCM techniques [32]–[35] are used for multiplications by two real constants represented by $\mathrm{AM}(C, S)$.

2) Number of Adders for MCR: The layout of the rotators influences the total number of adders. For a single angle in Fig. 1(a), the number of adders is obtained as explained in Section IV-B1. For multiple branches with one angle per rotator as in Fig. 1(c), the total number of adders of the kernel, AK, is equal to the sum of the adders in all the rotators, i.e.,

$$\mathrm{AK} = \sum_p \mathrm{AR}(P_p), \quad (15)$$

where P_p represents the rotation coefficient in branch p.

When data flows through a single branch and there are multiple rotation angles as in Fig. 1(b), only one coefficient is required at a time. This allows for merging the rotations and sharing the adders among them by using additional multiplexers. Thus, the total number of adders of the kernels is set by the coefficient with the highest adder cost, i.e.,

$$\mathrm{AK} = \max_i \{\mathrm{AR}(P_i)\}, \quad (16)$$

where P_i represents the coefficients that are merged. Most rotators admit different implementations, as sometimes additions and subtractions can be carried out in different orders. This allows for finding shared terms among the coefficients that reduce the number of multiplexers. For instance, both Fig. 4(a) and (b) include a 1-bit shift at the input. Therefore, the circuit in Fig. 5 does not need any multiplexer for the corresponding path. When the number of angles is large, the rotator may require more multiplexers to merge them. However, the amount of multiplexers can be reduced by not merging all the rotations, at the cost of a larger number of adders. For instance, the rotator in Fig. 5 has 4 adders and 6 multiplexers. Instead, the same rotator can be implemented with 8 adders and 2 multiplexers by implementing the circuits in Fig. 4(a) and (b) and multiplexing their outputs. An intermediate solution with 6 adders and 4 multiplexers is also possible.

Finally, the case of several branches and several rotations in each branch [Fig. 1(d)] is a combination of previous ones: Each rotator requires the maximum number of adders among the angles that it has to rotate, and the total number of adders is the sum of the adders of all the rotators, i.e.,

$$\mathrm{AK} = \sum_p \max_i \{\mathrm{AR}(P_{p,i})\}, \quad (17)$$

where $P_{p,i}$ is the ith coefficient of the pth branch.

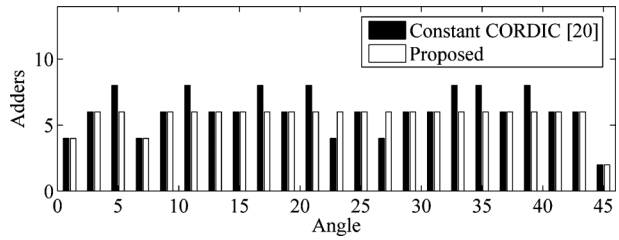


Fig. 7. Number of adders required to obtain an angular error smaller than $0.04°$ for angles $1°, 3°, 5°, \ldots, 45°$.

TABLE V
MAXIMUM ANGULAR ERROR IN DEGREES USING A GIVEN NUMBER OF ADDERS FOR ANGLES $1°, 3°, 5°, \ldots, 45°$

Adders	Constant CORDIC [20]	Proposed
4	1.875	1.31
6	n/a	0.0271
8	0.037	8.54×10^{-4}
10	n/a	5.37×10^{-6}
12	$\sim 5 \times 10^{-4}$	5.08×10^{-9}

V. EXPERIMENTAL RESULTS

This section presents experimental results of the proposed approach in several contexts. The experiments use the MCM algorithms in [32] and [33] to calculate the number of adders, and the best result among them is selected. For SCM calculations, the optimum results from [30] have been considered.

The search is done for coefficients that can be represented with word length up to 20 bits. This provides rotators with enough accuracy for most applications. If needed, higher accuracy can be achieved by increasing the maximum word length used in the search.

A. SCR With Arbitrary Scaling

For SCR with arbitrary scaling, a comparison is done based on the example in [20]. The work in [20] is based on finding the optimal sequence of CORDIC rotations. In the example, rotators for all odd degree angles between $1°$ and $45°$ are found. Two measures are used for comparison. First, the number of adders required to obtain an angular error smaller than $0.04°$ is shown in Fig. 7. As can be seen, the proposed approach requires six adders for two angles ($23°$ and $27°$) where the approach in [20] only requires four adders. However, there are seven angles where the approach in [20] requires eight adders (four CORDIC rotations), where the proposed approach only requires six adders. Hence, both the maximum and average number of adders are reduced using the proposed approach. Second, the maximum angular errors obtained using a given number of adders are shown in Table V. Clearly, the proposed approach results in a significantly smaller error, especially when more adders are allowed.

B. SCR With Unity Scaling

For SCR with unity scaling, Table VI compares the proposed approach with the MSR-CORDIC [23] for the twiddle factors $W_{128}^i = \cos(2\pi i/128) - j \cdot \sin(2\pi i/128)$, $i = 1, \ldots, 15$. The MSR-CORDIC consists of two stages in series with coefficients P_0 and P_1, leading to a rotation by a coefficient $P = P_0 \cdot P_1$. Conversely, the proposed approach uses a single stage. In both approaches, the scaling of each angle is very close to a power of two, 2^q, which provides unity scaling. The rotation error is the

TABLE VI
ANGLES WITH UNITY SCALING, 10 ADDERS

TF	MSR-CORDIC [23]						PROPOSED			
	P_0	P_1	$P = P_0 \cdot P_1$	ϵ	WL_E	WL	P	ϵ	WL_E	WL
W_{128}^1	$32 + j511$	$55 - j4096$	$2094816 - j102967$	9.57×10^{-5}	14.85	21	$4091 - j201$	1.68×10^{-5}	17.36	13
W_{128}^2	$256 + j7$	$2031 - j256$	$521728 - j51319$	1.50×10^{-4}	14.20	19	$8152 - j803$	6.77×10^{-5}	15.35	14
W_{128}^3	$129 - j16$	$126 - j3$	$16206 - j2403$	7.50×10^{-5}	15.20	14	$259249 - j38432$	2.53×10^{-4}	13.45	19
W_{128}^4	$4097 - j1024$	$62 + j3$	$257086 - j51197$	2.25×10^{-4}	13.62	18	$64287 - j12784$	1.58×10^{-4}	14.13	17
W_{128}^5	$513 + j2048$	$-j3973$	$8136704 - j2038149$	6.21×10^{-5}	15.47	23	$127147 - j31852$	3.91×10^{-5}	16.14	18
W_{128}^6	35	$56 - j17$	$1960 - j595$	2.59×10^{-4}	13.41	11	$3920 - j1189$	9.09×10^{-5}	14.92	13
W_{128}^7	$31 - j4$	$4097 - j896$	$123423 - j44164$	1.11×10^{-4}	14.61	17	$964 - j345$	1.40×10^{-4}	14.30	11
W_{128}^8	$30 - j$	$511 - j192$	$15138 - j6271$	9.81×10^{-5}	14.82	14	$7568 - j3135$	5.19×10^{-5}	15.73	14
W_{128}^9	$16 + j33$	$4 - j447$	$14815 - j7020$	9.44×10^{-4}	11.55	14	$7408 - j3503$	3.13×10^{-4}	13.14	14
W_{128}^{10}	$15 - j1024$	$31 + j56$	$57809 - j30904$	2.38×10^{-4}	13.54	16	$3612 - j1931$	9.38×10^{-5}	14.88	13
W_{128}^{11}	$126 - j$	$56 - j33$	$7023 - j4214$	5.24×10^{-4}	12.40	13	$28101 - j16856$	3.39×10^{-4}	13.03	16
W_{128}^{12}	$257 - j16$	$55 - j32$	$13623 - j9104$	9.47×10^{-5}	14.87	14	$54495 - j36414$	8.60×10^{-5}	15.01	17
W_{128}^{13}	$8 - j7$	$384 + j31$	$3289 - j2440$	2.30×10^{-4}	13.59	12	$6577 - j4880$	3.51×10^{-4}	12.98	14
W_{128}^{14}	$2 + j7$	$-56 - j129$	$791 - j650$	6.64×10^{-4}	12.06	10	$6334 - j5199$	3.10×10^{-4}	13.16	14
W_{128}^{15}	$48 - j$	$129 - j112$	$6080 - j5505$	1.31×10^{-3}	11.07	13	$1517 - j1376$	3.90×10^{-4}	12.82	12
W_{128}^{16}	$-j193$	$15 + j15$	$2895 - j2895$	4.52×10^{-4}	12.61	12	$46341 - j46336$	7.55×10^{-5}	15.19	17

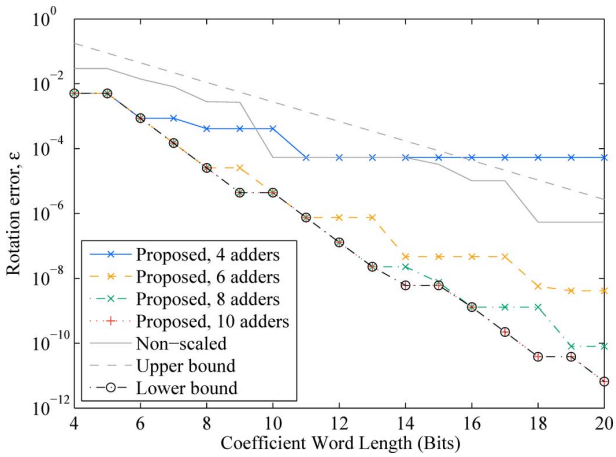


Fig. 8. Error of the proposed low-complexity rotators for an 8-point kernel (W_8) as a function of the word length and the number of adders.

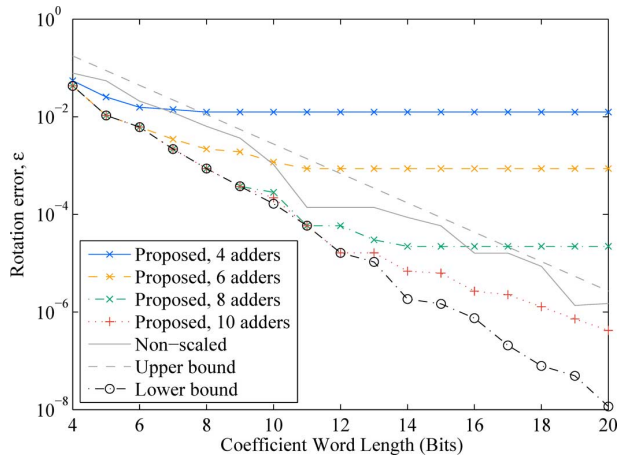


Fig. 9. Error of the proposed low-complexity rotators for a 16-point kernel (W_{16}) as a function of the word length and the number of adders.

distance from the complex coefficient to $2^k \cdot W_{128}^i$. The error is expressed in terms of effective word length (WL_E), which is defined as the number of bits of the output that are guaranteed to be accurate, and is calculated as

$$WL_E = -\log_2 \frac{\epsilon}{2\sqrt{2}} = -\log_2 \epsilon + \frac{3}{2}. \tag{18}$$

Finally, the table includes the coefficient word length (WL).

The results of both methods consider rotators that use at most 10 adders. The results for the MSR-CORDIC are taken from Table III in [23], and represented as $C + jS$ instead of numbers in the range $[-1, 1]$. By comparing the rotation error in both approaches, the maximum rotation error is 1.31×10^{-3} for the MSR-CORDIC and 3.90×10^{-4} for the proposed approach, i.e., the proposed approach reduces the maximum rotation error by a factor of 3.36. The mean error is also reduced from 3.46×10^{-4} in the MSR-CORDIC to 1.73×10^{-4} in the proposed approach, which is a reduction of 50%.

C. MCR With Uniform Scaling

The FFT calculates rotations by the twiddle factors $W_L^i = e^{-j \cdot 2\pi i/L}$, $i = 0, \ldots, L-1$. The number of angles in the set, L, is usually a power of two and its value depends on the FFT stage,

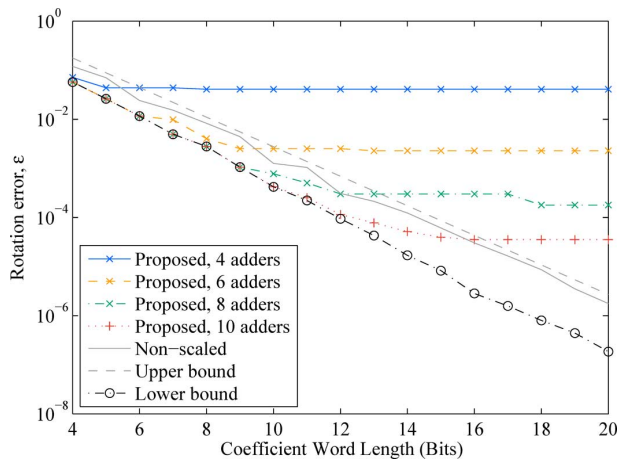


Fig. 10. Error of the proposed low-complexity rotators for a 32-point kernels (W_{32}) as a function of the word length and the number of adders.

as well as on the radix and decomposition [1], [4], [38]. Apart from W_4, which only involves trivial rotations [1] and is very simple to implement, W_8, W_{16} and W_{32} are the most common twiddle factors in FFT architectures: Radix-2 FFTs of size $N \geq 32$ calculate W_8, W_{16} and W_{32} rotations, a 4096-point radix-2^3 FFT needs W_8 rotators at four stages of the architecture, and a

TABLE VII
DESIGNED LOW-COMPLEXITY ROTATORS FOR FFT

TF	COEFFICIENTS					PROPERTIES			
	0	$\pi/16$	$\pi/8$	$3\pi/16$	$\pi/4$	ϵ	WL_E	WL	Add.
W_8	7	-	-	-	$5+j5$	$5.05\times10^{-3}\star$	9.13	4	4
	17	-	-	-	$12+j12$	$8.67\times10^{-4}\star$	11.67	6	4
	543	-	-	-	$384+j384$	5.34×10^{-5}	15.69	11	4
	577	-	-	-	$408+j408$	$7.51\times10^{-7}\star$	21.84	11	6
	6149	-	-	-	$4348+j4348$	4.64×10^{-8}	25.86	14	6
	196587	-	-	-	$139008+j139008$	4.14×10^{-9}	29.35	19	6
	19601	-	-	-	$13860+j13860$	$1.30\times10^{-9}\star$	31.02	16	8
	208885	-	-	-	$147704+j147704$	8.02×10^{-11}	35.04	19	8
	275807	-	-	-	$195025+j195025$	$6.57\times10^{-12}\star$	38.65	20	10
W_{16}	85	-	$80+j32$	-	$60+j60$	1.25×10^{-2}	7.82	8	4
	623	-	$576+j238$	-	$441+j441$	8.70×10^{-4}	11.67	11	6
	669	-	$618+j256$	-	$473+j473$	$5.86\times10^{-5}\star$	15.56	11	8
	8027	-	$7416+j3072$	-	$5676+j5676$	2.21×10^{-5}	16.97	14	8
	21059	-	$19456+j8059$	-	$14891+j14891$	2.67×10^{-6}	20.01	16	10
	349093	-	$322520+j133592$	-	$246846+j246846$	4.19×10^{-7}	22.69	20	10
	513764	-	$474656+j196609$	-	$363286+j363286$	8.51×10^{-8}	24.98	20	12
W_{32}	75	$72+j16$	$72+j32$	$64+j40$	$56+j56$	4.08×10^{-2}	6.12	8	4
	173	$170+j34$	$160+j66$	$144+j96$	$122+j122$	2.52×10^{-3}	10.13	9	6
	209	$205+j41$	$193+j80$	$174+j116$	$148+j148$	$1.06\times10^{-3}\star$	11.39	9	8
	1159	$1137+j226$	$1071+j444$	$964+j644$	$820+j820$	3.02×10^{-4}	13.19	12	8
	88600	$86912+j17280$	$81856+j33919$	$73680+j49248$	$62660+j62660$	1.79×10^{-4}	13.95	18	8
	21197	$20790+j4136$	$19584+j8112$	$17624+j11776$	$14988+j14988$	3.55×10^{-5}	16.28	16	10
	142009	$139280+j27704$	$131199+j54344$	$118076+j78896$	$100415+j100415$	3.55×10^{-6}	19.60	19	12

$\star$: Kernels labeled with ($\star$) reach the minimum achievable error for their word length.

4096-point radix-2^4 FFT needs W_{16} rotators at three stages [1], [38].

The twiddle factors are specific sets of angles generated by dividing the circumference in L equal parts. This leads to multiple symmetries in the complex plane. As a result, for an L-point kernel only $M = L/8 + 1$ angles in the range $[0, \pi/4]$ need to be considered. The rest of rotations can be calculated from those in $[0, \pi/4]$ by interchanging the real and imaginary components of the input and output data and/or the signs of the outputs. According to this and following the criterion of previous works, we present the results and the circuits for rotations in the range $[0, \pi/4]$. However, it is important to keep in mind that a circuit that computes the whole kernel in $[0, 2\pi]$ may require two additional real adders, which are equivalent to a complex adder.

Generally, a scaling in the rotations of the FFT is permissible, as long as it is the same for all the data [12]. Therefore, uniform scaling is considered in this experiment. Likewise, this experiment assumes the layout of a single branch with multiple rotations shown in Fig. 1(b).

1) Obtaining the Kernels: Figs. 8–10 show the proposed results for W_8, W_{16} and W_{32}, respectively. The figures show the trade-off between rotation error and number of adders. They include the upper and lower error bounds, and the error for non-scaled coefficients. The upper bound is the worst case approximation error corresponding to one half of the weight of the least significant bit (LSB) for both real and imaginary parts. This upper bound shows the points for which the effective word length is equal to the word length of the coefficients, i.e., $WL = WL_E$. The lower bound is the minimum error that can be achieved for a given incremental word length. This lower bound is provided in [12]. Finally, the error for non-scaled coefficients is the error of the kernels when the coefficients are simply obtained by rounding the sine and cosine of the angle to the closest values. The search has been done for coefficient word lengths up to 20 bits.

The upper and lower bounds and the case of non-scaled coefficients assume a full complex multiplier without adder optimization. Therefore, any result below the non-scaled case improves it in both accuracy and number of adders. For example, in Fig. 8 a W_8 kernels with $WL = 14$ that uses non-scaled coefficients achieves an error of 5.34×10^{-5} and requires four real multipliers and two adders. Conversely, the proposed approach only needs 6 adders to provide an error of 4.64×10^{-8}, i.e., more than three orders of magnitude smaller.

Table VII summarizes the most relevant results from Figs. 8 – 10. The first columns of the table show the coefficients that are used for the angles of the kernel, whereas the following columns include the parameters of the kernel: normalized error (ϵ), coefficient word length (WL) and number of adders. The error is provided both in linear units and in effective word length, WL_E. Those kernels marked with ($\star$) achieve the lowest rotation error for their word length.

The table shows various efficient alternatives to calculate accurate rotations with few adders. For instance, W_8 with accuracy of 29.35 bits can be calculated with only 6 adders. For W_{16} an accuracy of 22.69 bits is achieved with 10 adders. Compared to a general complex multiplier, this corresponds to two adders per real multiplier.

2) Shift-and-Add Implementation: Figs. 11 and 12 show the hardware circuits for some kernels in Table VII. They consists of adders, multiplexers and shifters. In all the implementations the rotation angle is selected using the control signals of the multiplexers. The different output configurations are shown by the symbols $\square$, $\bigcirc$ and $\triangle$. For the FFT the control signals can be generated directly from the bits of a counter [14]. This removes the necessity of a memory to store the rotation coefficients.

Fig. 11 shows two circuits for W_8, i.e., it considers the angles $\alpha_1 = 0$ and $\alpha_2 = \pi/4$. Fig. 11(a) shows the kernel $[543, 384 + j384]$. This circuit requires 4 adders and achieves an accuracy of 15.69 bits, as shown in Table VII. Depending on the con-

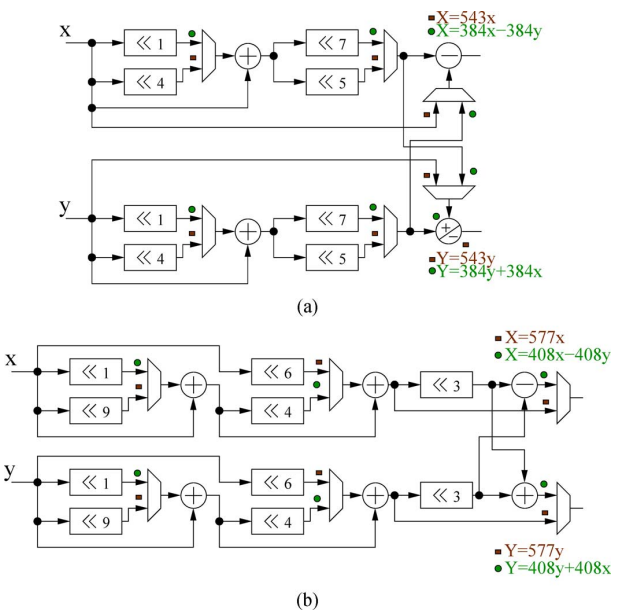


Fig. 11. Circuits for the calculation of W_8 rotations. (a) Kernel $[543, 384 + j384]$, 4 adders. (b) Kernel $[577, 408 + j408]$, 6 adders.

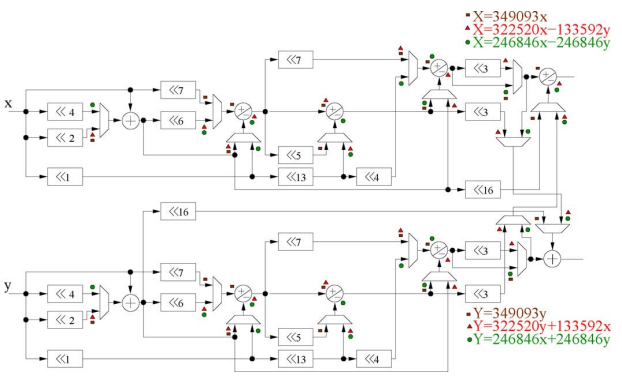


Fig. 12. Circuit for the calculation of W_{16} rotations. Kernel $[349093, 322520 + j133592, 246846 + j246846]$, 10 adders.

figuration of the multiplexers, the circuit multiplies the input signal either by 543 or by $384 + j384$. These multiplications are carried out by taking into account the shift-and-add representations of the numbers, i.e., $543A = 2^5 \cdot (2^4A + A) - A$ and $384A = 2^7 \cdot (2^1A + A)$.

Fig. 11(b) shows another option for W_8. In this case, the circuit considers the kernel $[577, 408 + j408]$. This circuit requires two adders more than that in Fig. 11(a). This reduces the rotation error to 7.51×10^{-7}, i.e., approximately two orders of magnitude or, equivalently, six correct fractional bits. As a result, both circuits in Fig. 11 are efficient implementations for W_8, and provide a trade-off between accuracy and hardware resources.

Finally, Fig. 12 shows an example for W_{16}. This kernel consists of the coefficients $[349093, 322520 + j133592, 246846 + j246846]$. The kernel achieves a precision of 22.69 correct bits by using 10 adders, as shown in Table VII.

3) Comparison: Figs. 13 and 14 compare the proposed rotators from Table VII with other multiplierless rotators for W_{16} and W_{32} in the literature. The previous approaches include rotators based on MCM [5][1], Booth encoding [25], trigonometric identities [7], [24], base-3 rotators [27], MSR-CORDIC [23][2] and non-redundant CORDIC [13]. The number of adders in the figures are for rotations in the range $[0, \pi/4]$. As said before, two more half adders are needed to calculate rotations in $[0, 2\pi]$.

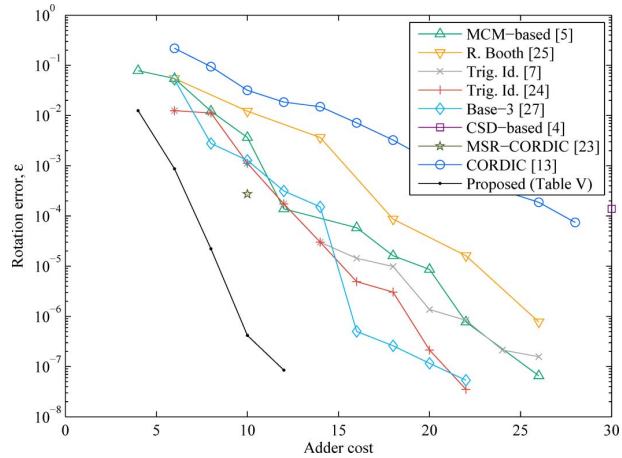


Fig. 13. Error versus number of adders of different W_{16} rotators.

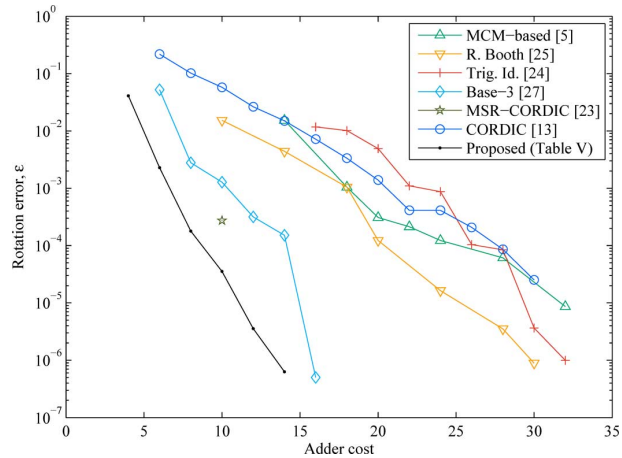


Fig. 14. Error versus number of adders of different W_{32} rotators.

Fig. 13 shows the results for W_{16}. Except for the CORDIC algorithm, which is a general rotator, all the approaches are constant rotators that offer *ad-hoc* solutions for W_{16}, leading to more accurate results. Among them, the proposed approach achieves less error and requires less adders than previous approaches. Quantitatively, it reduces the error more than one order of magnitude in kernels of 6 and 8 adders and more than three orders of magnitude for 10 adders. Furthermore, in order to meet the same precision as the proposed 10-adder kernel, previous approaches need at least 18 adders, i.e., almost twice the number of adders. For W_{32} rotators in Fig. 14, the proposed approach also outperforms previous ones with significant reductions in adders and error.

For the evaluation of the total hardware cost, Table VIII includes the number of 2-input multiplexers used in the rotators.

[1]For the results presented here, the algorithm in [33] is used, which generally gives fewer adders compared to the algorithm used originally [5].

[2]The W_{16} and W_{32} kernels are obtained by combining the W_{128}^i rotations in Table VI: W_{16} corresponds to $i = 0, 8, 16$ and W_{32} corresponds to $i = 0, 4, 8, 12, 16$.

TABLE VIII
COMPARISON OF W_{16} ROTATORS FOR $WL_E \geq 16$ BITS OR BEST RESULTS PROVIDED

Approach	Mux.	Add.	HW Cost	Lat.	Error	WL_E
Base-3 [27]	16	16	21.33	8	5.02e-7	21.58
CSD-based [4]	14	30	34.67	5	1.31e-4	14.31
Trig. Id. [24]	8	16	18.67	12	4.94e-6	19.12
MSR-CORDIC [23]	14	10	14.67	4	2.73e-4	13.33
R. Booth [25]	28	22	31.33	4	1.62e-5	17.41
MCM-based [5]	8	18	20.67	4	1.62e-5	17.41
Proposed, 8 adders	16	8	13.33	4	2.21e-5	16.97
Proposed, 10 adders (Fig. 12)	18	10	16	5	4.19e-7	22.69

The table considers rotators for which $WL_E \geq 16$ or, otherwise, the best cases provided. The hardware cost is calculated considering that the area of a multiplexer is one third of the area of an adder [39], i.e., $\text{HW Cost} = \text{Adders} + \text{Mux}/3$. The latency of the circuits is calculated as the number of adders in the critical path. The proposed rotators are taken from Table VII. The proposed 8-adder rotator has the lower latency and hardware cost among all the approaches. The proposed 10-adder rotator achieves the highest accuracy with low hardware cost and low latency.

VI. CONCLUSION

This paper presents a new approach to design low-complexity multiplierless constant rotators, based on combined coefficient selection and shift-and-add implementation. This combination increases the number of alternatives in the design, which widens the design space with respect to previous works.

The proposed approach applies to many hardware scenarios where rotations are carried out. These scenarios include rotations by a single angle or multiple angles, rotators in a single or multiple branches, and uniform, non-uniform, or unity scaling.

Experimental results for different contexts are provided. In all cases, significant reductions in complexity and improvements in accuracy are observed with respect to state of the art.

ACKNOWLEDGMENT

The authors would like to thank Dr. M. Kumm and Dr. S. Hemati for their valuable suggestions about the presentation of this work.

REFERENCES

[1] M. Garrido, J. Grajal, M. A. Sánchez, and O. Gustafsson, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 23–32, Jan. 2013.

[2] S. He and M. Torkelson, "Design and implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integr. Circuits Conf.*, May 1998, pp. 131–134.

[3] S.-N. Tang, J.-W. Tsai, and T.-Y. Chang, "A 2.4-GS/s FFT processor for OFDM-based WPAN applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 6, pp. 451–455, Jun. 2010.

[4] C.-H. Yang, T.-H. Yu, and D. Markovic, "Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example," *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 757–768, Mar. 2012.

[5] W. Han, A. T. Erdogan, T. Arslan, and M. Hasan, "High-performance low-power FFT cores," *ETRI J.*, vol. 30, no. 3, pp. 451–460, Jun. 2008.

[6] H. Liu and H. Lee, "A high performance four-parallel 128/64-point radix-24 FFT/IFFT processor for MIMO-OFDM systems," in *Proc. IEEE Asia-Pacific Conf. Circuits Syst.*, Nov. 2008, pp. 834–837.

[7] J.-Y. Oh and M.-S. Lim, "New radix-2 to the 4th power pipeline FFT processor," *IEICE Trans. Electron.*, vol. E88-C, no. 8, pp. 1740–1746, Aug. 2005.

[8] C. Loeffler, A. Ligtenberg, and G. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, May 1989, vol. 2, pp. 988–991.

[9] Z. Wu, J. Sha, Z. Wang, L. Li, and M. Gao, "An improved scaled DCT architecture," *IEEE Trans. Consum. Electron.*, vol. 55, no. 2, pp. 685–689, May 2009.

[10] A. Gray Jr. and J. Markel, "Digital lattice and ladder filter synthesis," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, no. 6, pp. 491–500, Jun. 1973.

[11] P. P. Vaidyanathan, "Passive cascaded-lattice structures for low-sensitivity FIR filter design, with applications to filter banks," *IEEE Trans. Circuits Syst.*, vol. 33, no. 11, pp. 1045–1064, Nov. 1986.

[12] M. Garrido, O. Gustafsson, and J. Grajal, "Accurate rotations based on coefficient scaling," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 58, no. 10, pp. 662–666, Oct. 2011.

[13] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, pp. 330–334, Sep. 1959.

[14] M. Garrido and J. Grajal, "Efficient memoryless CORDIC for FFT computation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Apr. 2007, vol. 2, pp. 113–116.

[15] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proc. ACM/SIGDA Int. Symp. FPGAs*, Feb. 1998, pp. 191–200.

[16] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures, and applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 9, pp. 1893–1907, Sep. 2009.

[17] C.-Y. Chen and C.-Y. Lin, "High-resolution architecture for CORDIC algorithm realization," in *Proc. Int. Conf. Commun. Circuits Syst.*, Jun. 2006, vol. 1, pp. 579–582.

[18] Y. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithm implementation," *IEEE Trans. Comput.*, vol. 42, no. 1, pp. 99–102, Jan. 1993.

[19] C.-S. Wu and A.-Y. Wu, "Modified vector rotational CORDIC (MVR-CORDIC) algorithm and architecture," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, , vol. 48, no. 6, pp. 548–561, Jun. 2001.

[20] P. K. Meher and S. Y. Park, "CORDIC designs for fixed angle of rotation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 2, pp. 217–228, Feb. 2013.

[21] C.-S. Wu, A.-Y. Wu, and C.-H. Lin, "A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, , vol. 50, no. 9, pp. 589–601, Sep. 2003.

[22] C.-H. Lin and A.-Y. Wu, "Mixed-scaling-rotation CORDIC (MSR-CORDIC) algorithm and architecture for high-performance vector rotational DSP applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 11, pp. 2385–2396, Nov. 2005.

[23] S. Y. Park and Y. J. Yu, "Fixed-point analysis and parameter selections of MSR-CORDIC with applications to FFT designs," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6245–6256, Dec. 2012.

[24] F. Qureshi and O. Gustafsson, "Low-complexity constant multiplication based on trigonometric identities with applications to FFTs," *IEICE Trans. Fundam.*, vol. E94-A, no. 11, pp. 324–326, Nov. 2011.

[25] Y.-E. Kim, K.-J. Cho, and J.-G. Chung, "Low power small area modified Booth multiplier design for predetermined coefficients," *IEICE Trans. Fundam.*, vol. E90-A, no. 3, pp. 694–697, Mar. 2007.

[26] V. Karkala, J. Wanstrath, T. Lacour, and S. P. Khatri, "Efficient arithmetic sum-of-product (SOP) based multiple constant multiplication (MCM) for FFT," in *Proc. IEEE/ACM Int. Comput.-Aided Design Conf.*, Nov. 2010, pp. 735–738.

[27] P. Källström, M. Garrido, and O. Gustafsson, "Low-complexity rotators for the FFT using base-3 signed stages," in *Proc. IEEE Asia-Pacific Conf. Circuits Syst.*, Dec. 2012, pp. 519–522.

[28] J. Jedwab and C. Mitchell, "Minimum weight modified signed-digit representations and fast exponentiation," *Electron. Lett.*, vol. 25, no. 17, pp. 1171–1172, Aug. 1989.

[29] S. C. Chan and P. M. Yiu, "An efficient multiplierless approximation of the fast Fourier transform using sum-of-powers-of-two (SOPOT) coefficients," *IEEE Signal Process. Lett.*, vol. 9, no. 10, pp. 322–325, Oct. 2002.

[30] O. Gustafsson, A. G. Dempster, K. Johansson, M. D. Macleod, and L. Wanhammar, "Simplified design of constant coefficient multipliers," *Circuits Syst. Signal Process.*, vol. 25, no. 4, pp. 225–251, Apr. 2006.

[31] J. Thong and N. Nicolici, "Time-efficient single constant multiplication based on overlapping digit patterns," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 9, pp. 1353–1357, Sep. 2009.

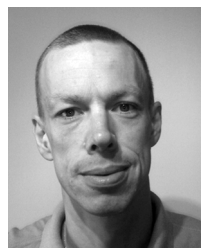
[32] A. G. Dempster and M. D. Macleod, "Multiplication by two integers using the minimum number of adders," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, vol. 2, pp. 1814–1817.
[33] O. Gustafsson, "A difference based adder graph heuristic for multiple constant multiplication problems," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 1097–1100.
[34] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algorithms*, vol. 3, pp. 1–39, May 2007.
[35] L. Aksoy, E. Günes, and P. Flores, "Search algorithms for the multiple constant multiplications problem: Exact and approximate," *Microprocess. Microsyst.*, vol. 34, no. 5, pp. 151–162, Aug. 2010.
[36] M. D. Macleod, "Multiplierless implementation of rotators and FFTs," *EURASIP J. Appl. Signal Process.*, vol. 2005, no. 17, pp. 2903–2910, 2005.
[37] O. Gustafsson and F. Qureshi, "Addition aware quantization for low complexity and high precision constant multiplication," *IEEE Signal Process. Lett.*, vol. 17, no. 2, pp. 173–176, Feb. 2010.
[38] F. Qureshi and O. Gustafsson, "Generation of all radix-2 fast Fourier transform algorithms using binary trees," in *Proc. Eur. Conf. Circuit Theory Design*, Aug. 2011, pp. 677–680.
[39] M. Janssen, F. Catthoor, and H. De Man, "A specification invariant technique for regularity improvement between flow-graph clusters," in *Proc. Eur. Design Test Conf.*, 1996, pp. 138–143.

Mario Garrido received the M.S. degree in electrical engineering and the Ph.D. degree from the Technical University of Madrid (UPM), Madrid, Spain, in 2004 and 2009, respectively.

In 2010 he moved to Sweden to work as a Postdoctoral Researcher at the Department of Electrical Engineering at Linköping University, Linköping, Sweden. Since 2012 he is Associate Professor in the same department. His research focuses on optimized hardware design for signal processing applications. This includes the design of hardware architectures for the calculation of transforms, such as the fast Fourier transform (FFT), circuits for data management, the CORDIC algorithm, and circuits to calculate statistical and mathematical operations. His research covers high-performance circuits for real-time computation, as well as designs for low area and low power consumption.

Fahad Qureshi was born in 1978. He received the M.Sc. degree from NED University of Engineering and Technology, Karachi, Pakistan, in 2004, and the Ph.D. degree from the Division of Electronics Systems, Linköping University, Linköping ,Sweden, in 2012.

His research interest is design and implementation of high performance resource flexible fast Fourier transforms.



Oscar Gustafsson (S'98–M'03–SM'10) received the M.Sc., Ph.D., and Docent degrees from Linköping University, Linköping, Sweden, in 1998, 2003, and 2008, respectively.

He is currently an Associate Professor and Head of the Electronics Systems Division, Department of Electrical Engineering, Linköping University. His research interests include design and implementation of DSP algorithms and arithmetic circuits. He has authored and coauthored over 140 papers in international journals and conferences on these topics.

Dr. Gustafsson is a member of the VLSI Systems and Applications and the Digital Signal Processing technical committees of the IEEE Circuits and Systems Society. Currently, he serves as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS and *Integration, the VLSI Journal*. He has served and serves in various positions for conferences such as ISCAS, PATMOS, PrimeAsia, Asilomar, Norchip, ECCTD, and ICECS.