Area–Time–Power Efficient FFT Architectures Based on Binary-Signed-Digit CORDIC

Hossein Mahdavi and Somayeh Timarchi

***Abstract*— Fast Fourier transform (FFT) is a significant technique in the digital signal processing (DSP). The intrinsic weak point of the primary designs of FFT was the use of multipliers. CORDIC is the most prevalent method to replace the multipliers of FFTs. In order to enhance the speed of CORDIC, redundant binary signed-digit (BSD) encodings can be utilized, so-called BSD-CORDIC. The main goal of this paper is to improve the efficiency of BSD-CORDIC-based FFT. The redundant improved composed (RIC) CORDIC is proposed as a novel BSD-CORDIC. The main idea behind the RIC CORDIC lies in prior determination of the required iterations in such a way that the maximum number of iterations is reduced, which results in speedup and decrease in hardware resources. Also, some of the iterations still can be skipped, which leads to further reduction in the power consumption. The synthesis results demonstrate that the FFT based on the RIC BSD-CORDIC enhances speed, power consumption, and area overhead.**

***Index Terms*— FFT, CORDIC, binary signed-digit number system, posibit-negabit encoding.**

I. Introduction

DISCRETE Fourier Transform (DFT) is an essential conversion between time and frequency domains in various applications of communication systems, signal, speech and image processing, etc. Fast Fourier Transform (FFT) was presented in 1965 to optimize the DFT by decreasing the number of its required arithmetic functions. The weakness of the conventional FFTs is employment of the complex multiplier blocks, which increases hardware, power consumption, and delay. In the literature, several methods have been developed to replace the complex multipliers, such as CORDIC, bit-parallel multipliers, distributed arithmetic [1], and modified-booth encoding [2]; among which CORDIC is the most prevalent one. To refine the CORDIC algorithm, some techniques have been introduced to decrease the number of iterations, including the deployment of the higher radix CORDICs [3]–[6], an approximation of the target angle within one iteration by means of a look-up table [7], [8] and exploitation of the greedy algorithm [9]. Moreover, some other methods have been developed to ameliorate the CORDIC performance, including elimination of the iterative nature of CORDIC [10], applying the scale free CORDIC which leads to removing the scale factor compensation stage [11]–[13], using a novel angle set which results in faster convergence [14], and removal of the sign detection required at each iteration by pre-computing the directions of the micro-rotations [4], [7], [15], [16]. In the specific applications such as FFT, the initial angles of CORDIC structure are known; therefore, a structure called look-ahead CORDIC can be employed which results in lower iterations. To further reduce hardware complexity, the low-power look-ahead CORDIC [17] deploys a single common hardware for the two CORDIC outputs. Moreover, redundant arithmetic has been utilized in some papers to improve the CORDIC speed [3]–[5], [15], [18]–[23]. The Binary Signed-Digit (BSD) encoding has been deployed in CORDIC to increase its speed by removing carry propagation during its iterative additions. However, due to the lack of a particular sign bit, sign detection of the BSD numbers is a slow process, so that direction determination for each micro-rotation in the BSD-CORDIC can be so slow that diminishes the benefits of utilizing BSD encoding.

In some BSD-CORDIC algorithms, the sign of residual angle is estimated by checking only a small window of the Most Significant Digits (MSDs). Although inspection of just a window of digits leads to sign estimation with a short and constant latency, it leads to a variable scale factor. In [18], two methods were presented, which utilized 50% extra iterations to avoid or correct the errors due to wrong sign estimations. Then, the branching method [19] deployed two basic modules to perform two CORDICs in parallel. Subsequently, double step branching CORDIC [20] used an extra module with better hardware utilization and more speedup. Indeed, the methods presented in [18]–[20] could achieve a constant scale factor at the cost of either additional iterations or extra module, which results in slow computation or high resources usage. To achieve further speedup, [3] developed the radix-4 CORDIC and [15] introduced the P-CORDIC for pre-computing the directions of micro-rotations. Eventually, [4] presented the parallel radix-4 CORDIC that achieves better latency and area overhead by employing both the two preceding ideas presented in [3], [15]. Also, [24] used modified booth encoding to accelerate the presented floating-point (FP) constant multiplier.

As more recently presented methods, we refer to the composed CORDICs presented in [7], [8], being composed of the coarse and precise blocks. The mentioned architectures reduce

Manuscript received August 28, 2018; revised December 11, 2018 and March 8, 2019; accepted June 4, 2019. This paper was recommended by Associate Editor P. K. Meher. *(Corresponding author: Somayeh Timarchi.)*

The authors are with the Faculty of Electrical Engineering, Shahid Beheshti University, G.C., Tehran 1983969411, Iran (e-mail: ho.mahdavi@mail.sbu.ac.ir; s_timarchi@sbu.ac.ir).

Digital Object Identifier 10.1109/TCSI.2019.2922988

the number of iterations and hardware resources. In the coarse stage, the input angle's MSDs are employed for memory addressing to retrieve the initial estimations of the trigonometric functions from an LUT. In the precise stage, the rest of rotations are predicted by the P-CORDIC algorithm, and then the final results are obtained by the CORDIC pipeline module. Moreover, an improved rotation strategy was proposed in [8], utilizing a greedy algorithm in the precise block to choose the best elementary angles for minimizing the number of steps. The composed architectures presented in [7], [8] are non-redundant CORDICs, and in addition, they cannot be deployed in the FFT architectures because they calculate only the trigonometric functions whose rough approximation values have already been stored in an LUT. However, for mixed multiplications required in FFT, since the values being multiplied are not predetermined, primary approximations of the results cannot be given in advance and stored in an LUT. As a result, the existing composed CORDICs cannot be deployed in FFT. This paper addresses an implementation of pipelined FFT based on the proposed Redundant Improved Composed (RIC) BSD-CORDIC. The main advantages of the contributions are as follows:

1) The proposed RIC CORDIC is able to perform all the same functions as the conventional CORDIC and is not limited to only the trigonometric functions. The versatility of the RIC CORDIC permits it to be deployed in the FFT architecture.

2) The maximum number of the iterations decreases by predetermining the required iterations, which leads to speed-up and reduction in area overhead.

3) Further speedup is achieved by deploying the posibit-negabit BSD encoding.

4) Power consumption is reduced by applying the data gating technique to the RIC CORDIC in addition to the reduction in the number of iterations.

The rest of this paper is organized as follows: In section II, the backgrounds are reviewed. Section III describes the proposed RIC CORDIC. Section IV makes comparisons and discusses the synthesis results. Finally, the paper will be concluded in section V.

II. Backgrounds

A. FFT

The DFT of input sequence $x\left(n\right)$ is obtained according to equation (1) where $W_N^{kn} = e^{-\frac{j2\pi kn}{N}}$ is called twiddle factor.

$$X(k) = \sum\nolimits_{n=0}^{N-1} x(n) W_N^{kn}; \quad K = 0, 1, 2, \ldots, N-1 \qquad (1)$$

Obtaining an N-point DFT entails a slow computation. Cooley and Tukey [25] published the FFT algorithm as an efficient method to compute the DFT. In fact, FFT exploits divide-and-conquer approach besides the symmetry and the periodicity properties of the twiddle factor so that the computational complexity is reduced from $O(N^2)$ to $N \log N$. The FFT architectures in which the sequence $x\left(n\right)$ is split into smaller subsequences are classified into the decimation-in-time (DIT) class, whereas the decimation in frequency (DIF) in which the sequence $X\left(k\right)$ is split. The radix of FFT is defined according to the number of the subsequences which the original sequence is split into. Radix-2 and radix-4 are the most popular FFT algorithms because of their simple structures [26]. FFT is computed by performing some fundamental arithmetic operations, called butterfly units.

The complex multiplication $B.W_N^{kn}$ of the butterfly structure corresponds to "the rotation of vector B by angle $-\frac{2\pi kn}{N}$" which can be implemented by CORDIC as the most prevalent alternative for the complex multipliers [27].

B. CORDIC

Volder [28] presented the CORDIC algorithm, which performs a vector rotation by simple shifts and additions. The basis of the CORDIC is the replacement of the original rotation by angle θ with a series of micro-rotations by the elementary angles $\alpha_i = \tan^{-1}\left(d_i\, 2^{-i}\right)$. The parameter $d_i = \pm 1$ determines the direction of each micro-rotation and is chosen in such a way that $\theta = \sum_{i=0}^{\infty} \alpha_i$. Each iteration of the CORDIC are calculated by (2), where Z_i represents the residual angle and Z_0 is called the initial angle. Moreover, the iteration components X_{i+1}, Y_{i+1} must be scaled by the scale factor K_i given by (3). Product of K_i s is denoted by K factor, $K = \prod_{i=0}^{N} K_i$, which can be executed as a constant multiplication either at the beginning or end of the iterations.

$$\begin{cases} X_{i+1} = X_i - d_i 2^{-i} Y_i \\ Y_{i+1} = Y_i + d_i 2^{-i} X_i; \quad d_i = \begin{cases} -1, Z_i < 0 \\ +1, Z_i \geq 0 \end{cases} \\ Z_{i+1} = Z_i - d_i \tan^{-1} 2^{-i} \end{cases} \qquad (2)$$

$$K_i = \frac{1}{\sqrt{1 + d_i^2 2^{-2i}}} = \frac{1}{\sqrt{1 + 2^{-2i}}} \qquad (3)$$

In general, the CORDIC algorithm is implemented in two forms of folded and unfolded. The folded architectures compute all the CORDIC iterations by time multiplexing on a single functional element; therefore, they consume less area at the cost of high latency. But in the unfolded (pipelined or non-pipelined) architectures, each processing unit is designated to perform a specific iteration so that the latency decreases at the expense of high area [3], [4]. In the remainder of the paper, the term "rotation" will be used instead of "micro-rotation".

C. Binary Signed-Digit

Since addition is regarded as the basic operation in digital arithmetic, any reduction in its latency improves the speed of many digital computational elements. To accelerate addition, some approaches have been deployed to restrict or even eliminate the carry propagation at the cost of additional area overhead, such as the carry look-ahead adder, parallel prefix tree adder, and utilization of a redundant number system. A number system with radix r and digit set $[\alpha, \beta]$ is called redundant if $\beta - \alpha + 1 > r$. The BSD number system, as a kind of the redundant number systems with $r = 2$ and the digit set {-1,0,1}, offers a carry-free addition with constant latency independent of the operands word-length [29].

In order to use BSD number system, the digit set needs to be encoded in binary representation. Several BSD encodings

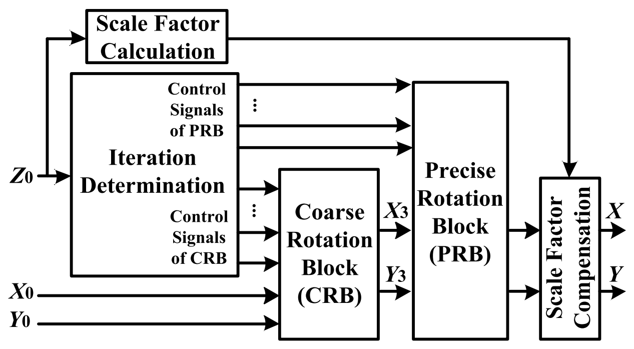

Fig. 1. Function blocks of the proposed RIC CORDIC architecture.

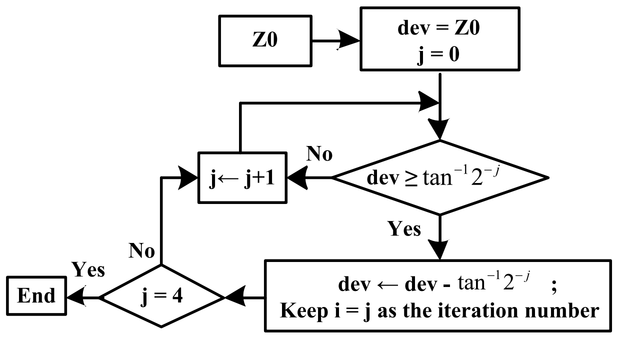

Fig. 2. The flowchart used for obtaining the iterations of the coarse block.

and addition schemes have been discussed in the literature [29]–[33]. As an example, in the posibit-negabit BSD encoding, the three BSD values -1, 0, and 1 are represented by “00”, “01”, and “11”, respectively. This representation permits to deploy simple NOT gates for easily inverting a value by reversing all of its bits. The posibit-negabit adder presented in [30] employs standard non-redundant binary addition components.

III. Proposed Redundant Improved Composed CORDIC (RIC CORDIC)

In this section, we propose an improved version of the composed CORDIC, called Redundant Improved Composed (RIC) CORDIC, which aims to achieve speedup by employing the posibit-negabit BSD encoding, while it is not restricted to calculation of just the trigonometric functions. The basic idea behind the proposed pipeline RIC CORDIC architecture is reducing the number of iterations by analyzing the MSDs of the input angle and digits of the residual angles in advance. The proposed RIC CORDIC is composed of the following blocks: iteration determination, coarse rotation, precise rotation, scale factor calculation, and scale factor compensation, as demonstrated in Fig. 1. The *iteration determination* block determines the required iterations in such a way that the number of iterations is minimized for all ranges of Z_0, leading to hardware reduction. Furthermore, to decrease power consumption, this block generates some signals to control data gating in the RIC CORDIC. To accomplish speedup, X_i and Y_i are represented with the posibit-negabit BSD encoding. However, due to the redundancy nature, the analysis of the digits in a redundant number is complicated; therefore, Z_0 is considered as a non-redundant operand for easily determining the required iterations. In Fig. 1, p is the number of pipeline stages in precise rotation block, which depends on the desired precision of computation.

A. Iteration Determination Block

In this block, two sets of control signals are produced for coarse rotation block (CRB) and precise rotation block (PRB). The first set of control signals are produced for stage0, stage1, and stage2 of CRB. The control signals *stg*0, *stg*0_*i*0, *stg*0_*i*1, *stg*0_*i*2, *stg*0_*i*3, *stg*0_*i*4 are produced for stage0 of CRB. The control signals *stg*1, *stg*1_*i*1, *stg*1_*i*2, *stg*1_*i*3, *stg*1_*i*4 are produced for stage1 of CRB. Finally, the control signals *stg*2, *stg*2_*i*2, *stg*2_*i*3, *stg*2_*i*4 are produced for stage2 of CRB.

The second set of control signals are produced for stage3, stage4, and stage5 of PRB. The control signals *stg*3, *stg*3_*i*3, *stg*3_*i*4, *stg*3_*i*5, *stg*3_*i*6, *stg*3_*i*7, *stg*3_*i*8 are produced for stage3 of PRB. The control signals *stg*4, *stg*4_*i*5, *stg*4_*i*6, *stg*4_*i*7, *stg*4_*i*8, *stg*4_*neg* are produced for stage4 of PRB. Finally, the control signals *stg*5, *stg*5_*i*6, *stg*5_*i*7, *stg*5_*i*8, *stg*5_*neg* are produced for stage5 of PRB.

- Control Signals for CRB

Firstly, for a fast and rough convergence of the elementary angles to the input angle Z_0, the numbers of the required iterations are obtained by looking at the MSDs of Z_0. The more inspecting MSDs, the more complexity of architecture. In this work, five MSDs of Z_0 are examined while Z_0 is limited to the first quadrant. For other initial angles, the angles in first quadrant are employed after affecting the positions and signs of X_0 and Y_0 according to trigonometric formulas. Considering all the 32 possible states for five MSDs, six states belong to the angles in the second quadrant; therefore, there are 26 angle ranges in the first quadrant. For each range, the required iterations have been already obtained by using the flowchart of Fig. 2, which is a specific greedy mechanism for the positive input angles while all the rotations are in just positive direction and precision is five digits.

TABLE I
The Required Iterations of the Stages of the Coarse Block, Which Are Determined in the Iteration Determination Block

"$Z_{0_0}Z_{0_1}Z_{0_2}Z_{0_3}Z_{0_4}$"	stage0	stage1	stage2
"00000"	-	-	-
"00001"	i_4	-	-
"00010"	i_3	-	-
"00011"	i_3	i_4	-
⋮	⋮	⋮	⋮
"11000"	i_0	i_1	i_2
"11001"	i_0	i_1	i_2

Table I explores the proposed method for 26 existing ranges and tabulates the required iterations of the coarse block according to the five MSDs of Z_0, where weight of Z_{0_i} is 2^{-i}. Also, i_j denotes the iteration related to the elementary angle $\tan^{-1}2^{-j}$, which is executed in one of the pipeline stages of the coarse block represented by stage0, stage1, and stage2. By deploying the flowchart of Fig. 2, all the 26 existing ranges require less than four iterations, except the ranges with MSDs of "10111" and "11001". In these two cases, the 4th iteration is eliminated but this elimination will be compensated in the precise block. Thus, the coarse block needs only three hardware stages instead of implementing all the five possible iterations of the conventional CORDIC. In addition, for some ranges of Z_0, the number of required iterations is even less than three; therefore, data gating technique can be used. As an instance, for the initial angles $0.0625\,rad \le Z_0 < 0.125\,rad$, whose MSDs are "00001", CORDIC can be computed more rapidly by performing only i_4 instead of all the five iterations $i_0, i_1, \ldots, i_4$. Whereby, as depicted in Table I, for the angles

Control signals of stage0:

$$\begin{aligned}
stg0\_i0 &= Z_{0_0} + Z_{0_1} \cdot Z_{0_3} \cdot \left(Z_{0_3} + Z_{0_4}\right)\\
stg0\_i1 &= \overline{Z_{0_0}} \cdot Z_{0_1} \cdot \left(\overline{Z_{0_2}} + Z_{0_2} \cdot \overline{Z_{0_3} + Z_{0_4}}\right)\\
stg0\_i2 &= \overline{Z_{0_0}} \cdot \overline{Z_{0_1}} \cdot Z_{0_2}\\
stg0\_i3 &= \overline{Z_{0_0}} \cdot \overline{Z_{0_1}} \cdot \overline{Z_{0_2}} \cdot Z_{0_3}\\
stg0\_i4 &= \overline{Z_{0_0}} \cdot \overline{Z_{0_1}} \cdot \overline{Z_{0_2}} \cdot \overline{Z_{0_3}} \cdot Z_{0_4}\\
stg0 &= \overline{Z_{0_0}} \cdot \overline{Z_{0_1}} \cdot \overline{Z_{0_2}} \cdot \overline{Z_{0_3}} \cdot \overline{Z_{0_4}}
\end{aligned}$$

Control signals of stage1:

$$\begin{aligned}
stg1\_1i1 &= Z_{0_0} \cdot \left(Z_{0_1} + \overline{Z_{0_1}} \cdot Z_{0_2}\right)\\
stg1\_i2 &= \overline{Z_{0_0}} \cdot Z_{0_1} . Z_{0_2} \cdot \overline{Z_{0_3}} \cdot \overline{Z_{0_4}} + Z_{0_0} \cdot \overline{Z_{0_1}} \cdot \overline{Z_{0_2}}\\
&\quad \cdot \left(Z_{0_3} + Z_{0_4}\right)\\
stg1\_i3 &= Z_{0_0} \cdot \overline{Z_{0_1}} \cdot \overline{Z_{0_2}} \cdot \overline{Z_{0_3}} \cdot \overline{Z_{0_4}}\\
&\quad + \overline{Z_{0_0}} \cdot Z_{0_3} \cdot Z_{0_1} \cdot Z_{0_2} \cdot Z_{0_4} + (Z_{0_1} \oplus Z_{0_2})\\
stg1\_i4 &= \overline{Z_{0_0}} \cdot [Z_{0_4} \cdot (\overline{Z_{0_1}} \cdot (Z_{0_2} \oplus Z_{0_3}) + Z_{0_1} . \overline{Z_{0_2}} \cdot \overline{Z_{0_3}})\\
&\quad + Z_{0_1} \cdot Z_{0_2} \cdot Z_{0_3} \cdot \overline{Z_{0_4}}]\\
stg1 &= \overline{Z_{0_0}} \cdot [\overline{Z_{0_1}} \cdot \overline{Z_{0_2}} \cdot \overline{Z_{0_3} \cdot Z_{0_4}}\\
&\quad + \overline{Z_{0_3}} \cdot (\overline{Z_{0_4}} \cdot (Z_{0_1} \oplus Z_{0_2}) + Z_{0_1} \cdot Z_{0_2} \cdot Z_{0_4})]
\end{aligned} \tag{4}$$

Control signals of stage2:

$$\begin{aligned}
stg2\_i2 &= Z_{0_0} \cdot Z_{0_1}\\
stg2\_i3 &= Z_{0_0} \cdot \overline{Z_{0_1}} \cdot Z_{0_3} \cdot \left(Z_{0_2} + Z_{0_4}\right)\\
stg2\_i4 &= \overline{Z_{0_0}} \cdot Z_{0_3} \cdot Z_{0_4} \cdot \left(Z_{0_1} \oplus Z_{0_2}\right)\\
&\quad + Z_{0_0} \cdot \overline{Z_{0_1}} \cdot \left(\overline{Z_{0_2}} \cdot \overline{Z_{0_4}} + Z_{0_2} \cdot \overline{Z_{0_3}} \cdot Z_{0_4}\right)\\
stg2 &= \overline{Z_{0_0}} \cdot (\overline{Z_{0_1}} \cdot \overline{Z_{0_2} \cdot Z_{0_3} \cdot Z_{0_4}}\\
&\quad + Z_{0_1} \cdot \left(\overline{Z_{0_2}} \cdot \overline{Z_{0_3} \cdot Z_{0_4}} + Z_{0_2}\right)) + Z_{0_0} . \overline{Z_{0_1}} \cdot \overline{Z_{0_3}}\\
&\quad \cdot \left(Z_{0_2} \oplus Z_{0_4}\right)
\end{aligned}$$

with such MSDs, stage0 executes iteration i_4 while stage1 and stage2 are disactivated to decrease power consumption.

By analyzing Table I, the control signals of coarse block are obtained as demonstrated in (4). Where, $stgX\_i0$, $stgX\_i1$, $stgX\_i2$, $stgX\_i3$, and $stgX\_i4$ are the signals used in stage X (X = 0,1,2) to execute the iterations i_0, i_1, i_2, i_3, and i_4, respectively, and $stgX$ is the signal which entails skipping stage X. For example, according to Table I, iteration i_3 is done in stage0 on the condition that the MSDs are "00010" or "00011" so signal $stg0\_i3$ is obtained by simplification of these two cases, resulting in $stg0\_i3 = \overline{Z_{0_0}}.\overline{Z_{0_1}}.\overline{Z_{0_2}}.Z_{0_3}$.

- Control Signals for PRB

In the iteration determination block, during the generation of the control signals of coarse block, the iteration determination related to precise block is done in parallel. The sum of the elementary angles corresponding to the iterations of coarse block, which is pre-computed for the different cases of Table I, is subtracted from the input angle Z_0. The result of this subtraction is called the residual angle Z_R whose four MSDs are definitely zero. Assuming $\tan^{-1} 2^{-i} \approx 2^{-i}$, which is plausible for $i > n/4$ (n is the number of digits that defines the precision), the digits of the residual angle will easily represent the iterations required in the precision block. As an example, for $n = 12$ and $Z_0 = \pi/16 \equiv 0.00110010010_2$, i.e. "$Z_{0_0} Z_{0_1} Z_{0_2} Z_{0_3} Z_{0_4}$" = "0.0011", two iterations i_3 and i_4 must be executed according to Table I. Thus, the sum of the

Control signals of stage3:

$$\begin{aligned}
stg3\_i3 &= Z_{R_4} \cdot [Z_{R_5} Z_{R_6} (Z_{R_7} + Z_{R_8})\\
&\quad + (Z_{R_5} \oplus Z_{R_6}) Z_{R_7} Z_{R_8}]\\
stg3\_i4 &= \overline{Z_{R_4}} Z_{R_5} Z_{R_6} Z_{R_7} Z_{R_8} + Z_{R_4} \cdot [\cdot \overline{Z_{R_5}} \cdot \overline{Z_{R_6} \cdot Z_{R_7} \cdot Z_{R_8}}\\
&\quad + Z_{R_5} \cdot (\cdot \overline{Z_6} \cdot \overline{Z_{R_7} \cdot Z_{R_8}} + Z_{R_6} \cdot \overline{Z_{R_7} + Z_{R_8}})]\\
stg3\_i5 &= \overline{Z_{R_4}} Z_{R_5} \cdot \overline{Z_{R_6} \cdot Z_{R_7} \cdot Z_{R_8}}\\
stg3\_i6 &= \overline{Z_{R_4}} \cdot \overline{Z_{R_5}} Z_{R_6}\\
stg3\_i7 &= \overline{Z_{R_4}} \cdot \overline{Z_{R_5}} \cdot \overline{Z_{R_6}} Z_{R_7}\\
stg3\_i8 &= \overline{Z_{R_4}} \cdot \overline{Z_{R_5}} \cdot \overline{Z_{R_6}} \cdot \overline{Z_{R_7}} Z_{R_8}\\
stg3 &= \overline{Z_{R_4}} \cdot \overline{Z_{R_5}} \cdot \overline{Z_{R_6}} \cdot \overline{Z_{R_7}} \cdot \overline{Z_{R_8}}
\end{aligned}$$

Control signals of stage4:

$$\begin{aligned}
stg4\_i5 &= Z_{R_4} . [\overline{Z_{R_5}} \cdot Z_{R_6} \cdot Z_{R_7} \cdot Z_{R_8} + Z_{R_5} (\overline{Z_{R_6}} \cdot \overline{Z_{R_7} \cdot Z_{R_8}}\\
&\quad + Z_{R_6} . \overline{Z_{R_7} + Z_{R_8}})]\\
stg4\_i6 &= (Z_{R_4} \oplus Z_{R_5}) \cdot Z_{R_6} \cdot \overline{Z_{R_7} \cdot Z_{R_8}} + Z_{R_4} \cdot Z_{R_5} \cdot \overline{Z_{R_6}}\\
&\quad \cdot Z_{R_7} Z_{R_8}\\
stg4\_i7 &= \overline{Z_{R_4}} \cdot Z_{R_7} (Z_{R_5} \oplus Z_{R_6}) + Z_{R_4} \cdot [\overline{Z_{R_5}} \cdot \overline{Z_{R_6}} \cdot Z_{R_7}\\
&\quad + Z_{R_5} \cdot Z_{R_6} (Z_{R_7} \oplus Z_{R_8})]\\
stg4\_i8 &= Z_{R_8} \cdot [\overline{Z_{R_4}} (\overline{Z_{R_5}} \cdot \overline{Z_{R_6}} \cdot Z_{R_7} + (Z_{R_5} \oplus Z_{R_6}) \cdot \overline{Z_{R_7}})\\
&\quad + Z_{R_5} \cdot Z_{R_6} \cdot Z_{R_7} + Z_{R_4} \cdot \overline{Z_{R_5}} \cdot \overline{Z_{R_6}} \cdot \overline{Z_{R_7}}]\\
stg4 &= \overline{Z_{R_4}} \cdot \overline{Z_{R_5}} (\overline{Z_{R_6}} \cdot \overline{Z_{R_7} \cdot Z_{R_8}} + Z_{R_6} \cdot \overline{Z_{R_7} + Z_{R_8}})\\
&\quad + (Z_{R_4} \oplus Z_{R_5}) \cdot \overline{Z_{R_6}} \cdot \overline{Z_{R_7}} \cdot \overline{Z_{R_8}}\\
stg4\_neg &= Z_{R_7} Z_{R_8} [\overline{Z_{R_4}} . Z_{R_5} . Z_{R_6} + Z_{R_4} . (Z_{R_5} \oplus Z_{R_6})]\\
&\quad + Z_{R_4} Z_{R_5} Z_{R_6} (Z_{R_7} + Z_{R_8})
\end{aligned} \tag{5}$$

Control signals of stage5:

$$\begin{aligned}
stg5\_i6 &= Z_{R_4} \cdot Z_{R_5} \cdot Z_{R_6} \overline{Z_{R_7}} \cdot \overline{Z_{R_8}}\\
stg5\_i7 &= Z_{R_7} \cdot \overline{Z_{R_8}} \cdot [\overline{Z_{R_4}} \cdot Z_{R_5} \cdot Z_{R_6} + Z_{R_4} (Z_{R_5} \oplus Z_{R_6})]\\
stg5\_i8 &= Z_{R_8} . [\overline{Z_{R_4}} (\overline{Z_{R_5}} \cdot Z_{R_6} \cdot Z_{R_7} + Z_{R_5} \cdot (Z_{R_6} \oplus Z_{R_7}))\\
&\quad + Z_{R_4} \cdot (\overline{Z_{R_5}} \cdot (Z_{R_6} + Z_{R_7}) + Z_{R_5} \cdot \overline{Z_{R_6} \cdot Z_{R_7}})]\\
stg5 &= \overline{Z_{R_4}} \cdot [\overline{Z_{R_5}} \cdot \overline{Z_{R_6} \cdot Z_{R_7} \cdot Z_{R_8}} + Z_{R_5} \cdot (\overline{Z_{R_6}} \cdot \overline{Z_{R_7} \cdot Z_{R_8}}\\
&\quad + Z_{R_6} \cdot \overline{Z_{R_7} \oplus Z_{R_8}})] + Z_{R_4} \cdot (\overline{Z_{R_5}} \cdot \overline{Z_{R_6}} \cdot \overline{Z_{R_7} \cdot Z_{R_8}}\\
&\quad + (Z_{R_5} \oplus Z_{R_6}) \overline{Z_{R_7} + Z_{R_8}} + Z_{R_5} \cdot Z_{R_6} \cdot Z_{R_7})\\
stg5\_neg &= Z_{R_4} \cdot Z_{R_8} \cdot (\overline{Z_{R_5}} \cdot Z_{R_6} \cdot Z_{R_7} + Z_{R_5} (Z_{R_6} \oplus Z_{R_7}))
\end{aligned}$$

TABLE II
PROPOSED ITERATION DETERMINATION OF THE PRECISE BLOCK ACCORDING TO THE FIRST WINDOW OF THE RESIDUAL ANGLE Z_K

"$Z_{R_4}Z_{R_5}Z_{R_6}Z_{R_7}Z_{R_8}$"	iterations in the PRB					
	i_3	i_4	i_5	i_6	i_7	i_8
"00000"	0	0	0	0	0	0
"00001"	0	0	0	0	0	1
⋮						
"01111"	0	1	0	0	0	$\bar{1}$
⋮						
"10111"	1	0	$\bar{1}$	0	0	$\bar{1}$
⋮						
"11011"	1	0	0	$\bar{1}$	0	$\bar{1}$
"11100"	0	1	1	1	0	0
"11101"	1	0	0	0	$\bar{1}$	$\bar{1}$
"11110"	1	0	0	0	$\bar{1}$	0
"11111"	1	0	0	0	0	$\bar{1}$

TABLE III
THE REQUIRED ITERATIONS IN THE STAGES OF THE PRECISE BLOCK, WHICH ARE DETERMINED IN THE ITERATION DETERMINATION BLOCK ACCORDING TO THE FIRST WINDOW OF THE RESIDUAL ANGLE Z_R

"$Z_{R_4}Z_{R_5}Z_{R_6}Z_{R_7}Z_{R_8}$"	stage3	stage4	stage5
"00000"	-	-	-
"00001"	i_8	-	-
⋮	⋮	⋮	⋮
"01111"	i_4	i'_8	-
⋮	⋮	⋮	⋮
"11110"	i_3	i'_7	-
"11111"	i_3	i'_8	-

elementary angles is $\tan^{-1} 2^{-3} + \tan^{-1} 2^{-4} \equiv 0.00101111110_2$, which results $Z_R = Z_0 - 0.00101111110_2 = 0.00000010100_2$. For high precision computations, in order to verify condition $i > n/4$ for the digits of Z_R, either number of the MSDs of Z_0 associated with the iterations of coarse block must increase or more than one coarse block must be employed.

Because the precision block is considered for the iterations $i > n/4$, equation $\tan^{-1} 2^{-i} \approx 2^{-i}$ is valid. Thus, the required iterations are equivalent to the digits order in Z_R. The digits of Z_R, except its four MSDs that are definitely zero, are divided into five-digit windows. For reducing the number of iterations, in the cases that a five-digit window has more than three '1's, the digits of the window change so that each window corresponds to at maximum three iterations. Table II summarizes the possible values of the first window of Z_R and their corresponding iterations required to be performed in the precise block, in which all the particular cases with more than three '1' digits have been modified to avoid the cases require more than three iterations. Where, the digits of Z_R are denoted by Z_{R_i} whose weight is equal to 2^{-i}.

For the first five-digit window, the iterations required to be executed in the stages of the precise block are extracted from Table II and listed in Table III. Where, stage3, stage5 are the pipeline stages and i'_j represents the negative rotation direction of iteration i_j.

By analyzing Table III, the control signals of precise block are obtained and illustrated in (5). Where, $stg3$, $stg4$ and $stg5$ are the signals which entail skipping stage3, stage4 and stage5, respectively, to exploit data gating. $stg4\_neg$ and $stg5\_neg$ are the signals imply negative rotation directions in stage4 and stage5, respectively. Moreover, $stg3\_i3$, $stg3\_i4$, $stg3\_i5$, $stg3\_i6$, $stg3\_i7$, and $stg3\_i8$ are the signals used in stage3 to execute iterations i_3, i_4, i_5, i_6, i_7, and i_8, respectively. Likewise, the other signals in (5) are employed in stage4 and stage5 to control execution of iterations.

The number of analyzed five-digit windows of Z_R depends on our desired precision of computation. For each window, the exact same modifications as Table II are done to evade the cases need more than three iterations. Despite the probable repetitions of the iteration i_3 or i_4 for the first window and i_8 for the second window and so on, the total number of required stages has almost 35% reduction.

B. CRB

Fig. 3 depicts the designs of stage0, stage1, and stage2 of the coarse block whose operation corresponds to Table I. In the figure, to execute subtractions by adders, each module NOT, which is a series of NOT gates, inverts the posibit-negabit data by reversing all the bits. In this paper, the data gating technique is performed in the CRB by utilizing the signals $stg0$, $stg1$, and $stg2$. The second input of each adder is the shifted signal determined by the control signals obtained from (4).

C. PRB

Fig. 4 displays the designs for stage3, stage4, and stage5. In this paper, the control signals obtained from (5) are exploited in the stages of PRB to control performing the iterations by using the proper shifters and to skip the stages by using the signals $stg3$, $stg4$, and $stg5$ on the condition that data gating is possible. As it is shown in Table III, the negative direction of rotations is probable in stage4 and stage5. Hence, in the designs of these stages, the signals $stg4\_neg$ and $stg5\_neg$ have been deployed to control the direction of iterations. The implementation of the other stages of PRB proceeds in the same way until reaching the desired precision.

D. Scale Factor Calculation and Compensation

Since the proposed RIC CORDIC employs different iterations dependent on the range of the input angle, the scale factor is variable and needs to be calculated. The scale factor is obtained by considering the K_i values for $i \leq n/4$ because K_i is approximated by one for $i > n/4$. Hence, for low precision computations, the scale factor is easily obtained by inspecting the MSDs of Z_0. Furthermore, to avoid increasing complexity of the high precision computations, some of K_is in the range of $i \leq n/4$ can be approximated by some terms of Taylor series expansion like the method explained in [3], [4]. To compensate the obtained scale factor, redundant tree adder circuit is deployed after the precise block.

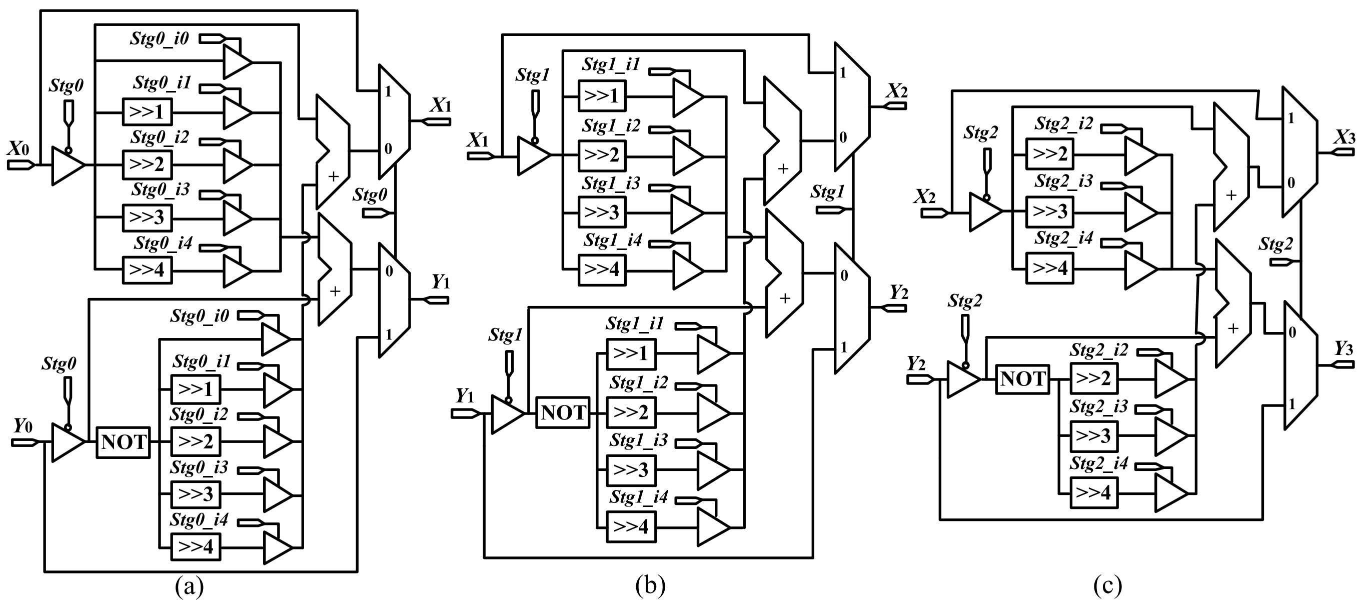

Fig. 3. (a) stage0 (b) stage1 (c) stage2 structures of the coarse block in the proposed RIC CORDIC.

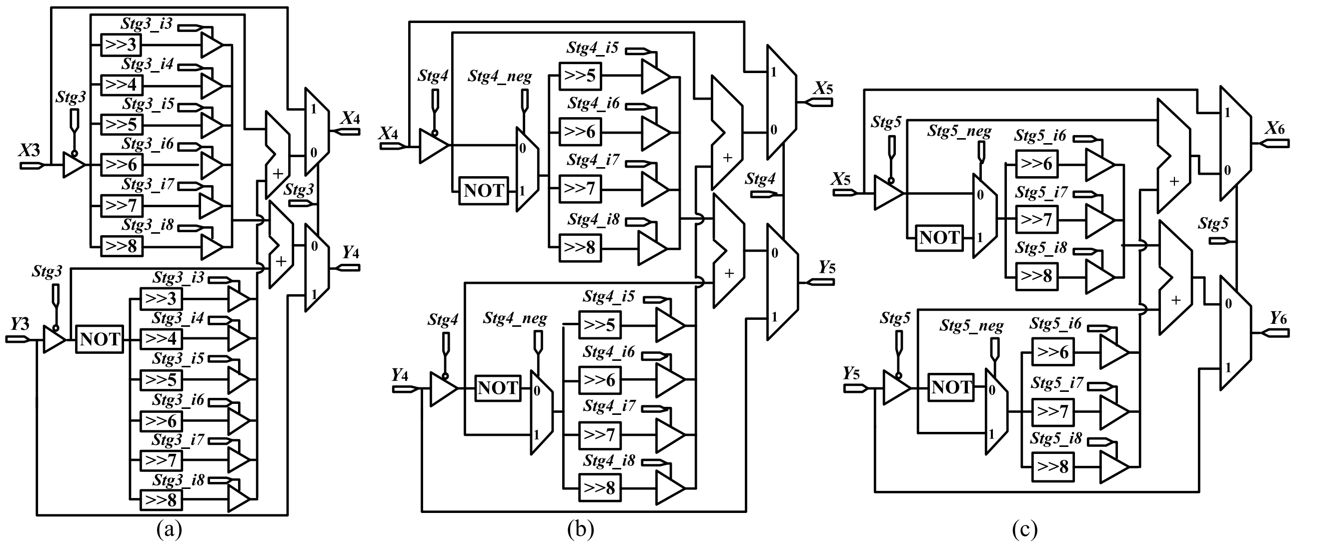

Fig. 4. (a) stage3 (b) stage4 (c) stage5 structures of the precise block in the proposed RIC CORDIC.

IV. Simulation and Comparison

To make a fair comparison, a 16-point radix-2 DIT FFT with pipeline unfolded architecture was considered for all the structures. The architectures were realized in a pipeline structure with equal depth (10 stages) and computation precision (8-digit precision). Also, the posibit-negabit BSD-adders were employed according to [30]. Since the proposed structure employs BSD arithmetic, the architectures based on the signed-digit CORDIC structures presented in [4], [18]–[20], and the BSD FP butterfly presented in [24] have been selected for synthesis comparison. All the proposed and the existing architectures have been described by VHDL and implemented using the Xilinx ISE Design Suit 14.7. After the functional verification by Mentor's Modelsim simulator, the analyzed circuits were synthesized using the Synopsys Design Compiler with the TSMC 90 nm technology in the typical conditions $V_{DD} = 1.2\,v$ and $T = 25\,°\mathrm{C}$. Finally, the synthesized FFT architectures were compared in terms of delay, power consumption, and area overhead.

A. Complexity Comparison of CORDICs

Table IV depicts the hardware complexity comparison of the proposed and existing CORDICs in terms of the number of adders and multiplexers. The table shows the number of adders (in x/y path, z path, and scaling compensation) and multiplexers involved in all the iterations. Since each rotation in the double rotation [18] method is done by a combination of two rotation-extensions, it requires around 50% additional adders in the x/y path compared with the conventional CORDIC. In both the branching [19] and double step branching [20]

TABLE IV
COMPLEXITY ANALYSIS OF THE DIFFERENT CORDICS WITH 9-DIGIT PRECISION

	Number of Iterations (Stages)	Hardware Cost			
		Number of Adders			Number of Multiplexers
		Rotation (x/y path)	Rotation Direction Determination (z path)	Scaling compensation	
Double Rotation [18]	9 iterations	28	8	4	-
Branching [19]	9 iterations	36	18	4	27
Double Step Branching [20]	5 iterations	40	10	4	15
Radix-4 CORDIC [4]	5 iterations	10	3	6	1
BSD FP Complex Multiplier [24]	-	16	-	-	20
CORDIC II [14]	6 stages	16	2	-	21
RIC [proposed]	6 stages	12	1	6	16

TABLE V
SIMULATION RESULTS OF THE 16-POINT RADIX-2 DIT FFT ARCHITECTURES WITH 10-STAGE PIPELINE STRUCTURE USING DIFFERENT 16-DIGIT SD-BUTTERFLIES WITH 8-DIGIT PRECISION

	Max. Frequency (MHz)	Power (mw)			Area (μm^2)		
		400 MHz	500 MHz	666 MHz	400 MHz	500 MHz	666 MHz
Double Rotation [18]	418	314	-	-	7349k	-	-
Branching [19]	483	247.6	-	-	8455k	-	-
Double Step Branching [20]	537	210.9	303.6	-	7914k	8170k	-
Radix-4 CORDIC-based [4]	694	167.8	226.1	381.5	6384k	6452k	7101k
BSD FP butterfly-based [24]	515	213.8	307.5	-	8811k	9508k	-
RIC [proposed]	877	140.1	182.4	268.3	4640k	4693k	4733k

methods, the number of adders in the x/y path doubles due to deployment of two basic CORDIC modules in parallel. Also, in both the mentioned methods, the z path requires one adder per iteration in each module. The parallel radix-4 CORDIC [4] requires less adders due to the decrease in the number of the iterations and pre-computation of the directions of rotations. In the BSD FP multiplier [24], the number of adders is reduced by storing the twiddle factors in modified booth encoding. The multiplexers of the BSD FP multiplier are utilized to generate the partial products. Also, CORDIC II [14] decreases the number of the adders by substituting the CORDIC micro-rotation by three types of rotators.

Considering the higher-radix adders deployed in Radix-4 CORDIC and the adders with wider word-length employed in the BSD FP multiplier, we can draw the conclusion that the proposed RIC CORDIC has plausible complexity.

B. Speed Comparison

Since all the analyzed FFT architectures have the same pipeline depth and throughput (one output per cycle), their speeds have been compared at the maximum frequency whereby each structure could operate accurately. Table V displays the maximum frequency of the FFT architectures using different SD-butterflies based on the BSD-CORDIC or BSD FP butterfly. It is obvious that the proposed RIC CORDIC-based FFT is the fastest FFT structure amongst the analyzed FFTs so that its latency decreases 21% compared with the FFT using the radix-4 CORDIC [4]. This notable speedup is a result of almost 35% decrease in the maximum number of iterations besides the employment of the posibit-negabit BSD adders.

C. Power Comparison

The power consumption results for the FFT architectures operating at the frequencies 400 MHz, 500 MHz, and 666 MHz are tabulated in Table V. According to the results, the FFT using the proposed RIC CORDIC consumes less power due to decreasing the number of iterations as well as employing data gating technique to deactivate the idle pipeline stages at some ranges of the initial angles of CORDICs.

D. Area Comparison

The area results of the analyzed FFTs are also listed in Table V. Reduction in the maximum number of iterations as well as shortening the z-path in the proposed RIC CORDIC make the RIC CORDIC-based FFT the most area-efficient. Considering the frequency 666 MHz, the area reduction of the FFT using the RIC CORDIC is 34% compared with the FFT utilizing the radix-4 CORDIC [4].

V. CONCLUSION

The proposed FFT using the RIC CORDIC improves performance of the FFT structure based on the radix-4 CORDIC [4] by 21%, 30%, and 34% reductions in delay, power, and area, respectively. In fact, in addition to speedup due to the use of posibit-negabit BSD number system and reduction in the number of iterations, the proposed RIC CORDIC achieves the best efficiency in resources usage by decreasing the maximum number of required iterations, i.e. pipeline stages, and shortening the z-path. Moreover, the power consumption of the RIC CORDIC further decreased by utilizing data gating technique to deactivate some stages for the angles whose rotations are realized by less iterations. To conclude, the proposed RIC CORDIC is the most area-time-power efficient candidate to be employed in FFT architectures.

REFERENCES

[1] A. Kaivani and S. Ko, "High-speed FFT processors based on redundant number systems," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 2237–2240.

[2] E. Antelo, J. Villalba, J. D. Bruguera, and E. L. Zapata, "High performance rotation architectures based on the radix-4 CORDIC algorithm," *IEEE Trans. Comput.*, vol. 46, no. 8, pp. 855–870, Aug. 1997.

[3] B. Lakshmi and A. S. Dhar, "VLSI architecture for parallel radix-4 CORDIC," *Microprocess. Microsyst.*, vol. 37, no. 1, pp. 79–86, Feb. 2013.

[4] J. D. Bruguera, E. Antelo, and E. L. Zapata, "Design of a pipelined radix 4 CORDIC processor," *Parallel Comput.*, vol. 19, no. 7, pp. 729–744, Jul. 1993.

[5] B. Lakshmi and A. S. Dhar, "VLSI architecture for low latency radix-4 CORDIC," *Comput. Elect. Eng.*, vol. 37, no. 6, pp. 1032–1042, Nov. 2011.

[6] F. Aguirre-Ramos, A. Morales-Reyes, R. Cumplido, and C. Feregrino-Uribe, "An Area efficient composed CORDIC architecture," *Adv. Electr. Comput. Eng.*, vol. 14, no. 2, pp. 113–116, Jan. 2014.

[7] Y. Zhang, Y.-T. Liao, K.-T. Chen, and T. Baba, "Hardware architecture for CORDIC with improved rotation strategy," *J. Signal Process.*, vol. 20, no. 4, pp. 141–144, Jul. 2016.

[8] R. Xu, Z. Jiang, H. Huang, and C. Dong, "An optimization of CORDIC algorithm and FPGA implementation," *Int. J. Hybrid Inf. Technol.*, vol. 8, no. 6, pp. 217–228, 2015.

[9] B. Gisuthan and T. Srikanthan, "Pipelining flat CORDIC based trigonometric function generators," *Microelectron. J.*, vol. 33, nos. 1–2, pp. 77–89, Jan. 2002.

[10] F. J. Jaime, M. A. Sánchez, J. Hormigo, J. Villalba, and E. L. Zapata, "Enhanced scaling-free CORDIC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 7, pp. 1654–1662, Jul. 2010.

[11] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 11, pp. 1463–1474, Nov. 2005.

[12] M. Raguram and R. Parameshwaran, "A modified scale free CORDIC architecture to improve the speed," Contemp. Eng. Sci., SASTRA Univ., India, Tech. Rep., 2014, pp. 435–441, vol. 7, no. 9.

[13] M. Garrido, P. Källström, M. Kumm, and O. Gustafsson, "CORDIC II: A new improved CORDIC algorithm," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 63, no. 2, pp. 186–190, Feb. 2016.

[14] M. Kuhlmann and K. K. Parhi, "P-CORDIC: A precomputation based rotation CORDIC algorithm," *EURASIP J. Appl. Signal Process.*, vol. 9, no. 1, pp. 936–943, Sep. 2002.

[15] T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai, "Para-CORDIC: Parallel CORDIC rotation algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 8, pp. 1515–1524, Aug. 2004.

[16] A. Shabani and S. Timarchi, "Low-power DCT-based compressor for wireless capsule endoscopy," *Signal Process., Image Commun.*, vol. 59, pp. 83–95, Nov. 2017.

[17] N. Takagi, T. Asada, and S. Yajima, "Redundant CORDIC methods with a constant scale factor for sine and cosine computation," *IEEE Trans. Comput.*, vol. 40, no. 9, pp. 989–995, Sep. 1991.

[18] J. Duprat and J.-M. Müller, "The CORDIC algorithm: New results for fast VLSI implementation," *IEEE Trans. Comput.*, vol. 42, no. 2, pp. 168–178, Feb. 1993.

[19] D. S. Phatak, "Double step branching CORDIC: A new algorithm for fast sine and cosine generation," *IEEE Trans. Comput.*, vol. 47, no. 5, pp. 587–602, May 1998.

[20] H. Dawid and H. Meyr, "The differential CORDIC algorithm: Constant scale factor redundant implementation without correcting iterations," *IEEE Trans. Comput.*, vol. 45, no. 3, pp. 307–318, Mar. 1996.

[21] R. Shukla and K. C. Ray, "Low latency hybrid CORDIC algorithm," *IEEE Trans. Comput.*, vol. 63, no. 12, pp. 3066–3078, Dec. 2014.

[22] D. Timmermann, H. Hahn, and B. J. Hosticka, "Low latency time CORDIC algorithms," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 1010–1015, Aug. 1992.

[23] A. Kaivani and S. Ko, "Floating-point butterfly architecture based on binary signed-digit representation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 3, pp. 1208–1211, Mar. 2016.

[24] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, 1965.

[25] J. Takala and K. Punkka, "Butterfly unit supporting radix-4 and radix-2 FFT," in *Proc. Int. TICSP Workshop Spectral Methods Multirate Signal Process. (SMMSP)*, Riga, Latvia, vol. 30, pp. 47–54, Jun. 2005.

[26] X. Xiao, E. Oruklu, and J. Saniie, "Reduced memory architecture for CORDIC-based FFT," in *Proc. IEEE Int. Symp. Circuits Syst.*, May/Jun. 2010, pp. 2690–2693.

[27] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. 8, no. 3, pp. 330–334, Sep. 1959.

[28] A. Avizienis, "Signed-digit numbe representations for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, no. 3, pp. 389–400, Sep. 1961.

[29] S. Timarchi, P. Ghayour, and A. Shahbahrami, "A novel high-speed low-power binary signed-digit adder," in *Proc. 16th CSI Int. Symp. Comput. Archit. Digital Syst. (CADS)*, May 2012, pp. 70–74.

[30] S. Timarchi and N. Akbarzadeh, "Area-Time-power efficient maximally redundant signed-digit modulo 2^n–1 adder and multiplier," *Circuits, Syst., Signal Process.*, vol. 38, no. 5, pp. 2138–2164, May 2018.

[31] M. Saremi and S. Timarchi, "Efficient modular binary signed-digit multiplier for the moduli set2^n- 1, 2^n, 2^n+ 1," *CSI J. Comput. Sci. Eng.*, vol. 9, no. 2, pp. 52–62, 2011.

[32] S. Timarchi, M. Fazlali, and S. D. Cotofana, "A unified addition structure for moduli set 2^n-1, 2^n, 2^n+1 based on a novel RNS representation," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Amsterdam, The Netherlands, Oct. 2010, pp. 247–252.

Hossein Mahdavi received the B.Sc. degree in electrical and electronics engineering from Islamic Azad University, Bojnourd, Iran, in 2006, and the M.Sc. degree in digital electronics from Shahid Beheshti University, Tehran, Iran, in 2017. He is currently a Researcher with the Faculty of Electrical Engineering, Shahid Beheshti University. His research interests include designing low-power and high-speed digital circuits, VLSI design, and digital signal processing.

Somayeh Timarchi received the B.Sc. degree in computer engineering from Shahid Beheshti University in 2002, the M.Sc. degree in computer system architecture from the Sharif University of Technology in 2004, and the Ph.D. degree in computer system architecture from Shahid Beheshti University in 2010. She performed researches at the University of Siena, Italy, and the Delft University of Technology (TU Delft). In 2010, she joined the Department of Electrical Engineering, Shahid Beheshti University, as an Assistant Professor. Her research interests include computer arithmetic, circuit design for digital signal processing and cryptography, VLSI design, low-power circuits, and approximate computing.