Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2012, Article ID 185784, 8 pages
doi:10.1155/2012/185784

Research Article

Redesigned-Scale-Free CORDIC Algorithm Based FPGA Implementation of Window Functions to Minimize Area and Latency

Supriya Aggarwal and Kavita Khare

Department of Electronics and Communication Engineering, MANIT, Bhopal 462007, India

Correspondence should be addressed to Supriya Aggarwal, sups.aggarwal@gmail.com

Received 4 January 2012; Revised 29 April 2012; Accepted 25 June 2012

Academic Editor: Michael Hübner

One of the most important steps in spectral analysis is filtering, where window functions are generally used to design filters. In this paper, we modify the existing architecture for realizing the window functions using CORDIC processor. Firstly, we modify the conventional CORDIC algorithm to reduce its latency and area. The proposed CORDIC algorithm is completely scale-free for the range of convergence that spans the entire coordinate space. Secondly, we realize the window functions using a single CORDIC processor as against two serially connected CORDIC processors in existing technique, thus optimizing it for area and latency. The linear CORDIC processor is replaced by a shift-add network which drastically reduces the number of pipelining stages required in the existing design. The proposed design on an average requires approximately 64% less pipeline stages and saves up to 44.2% area. Currently, the processor is designed to implement Blackman windowing architecture, which with slight modifications can be extended to other widow functions as well. The details of the proposed architecture are discussed in the paper.

1. Introduction

Window filtering techniques [1, 2] are commonly employed in signal processing paradigm to limit time and frequency resolution. Various window functions are developed to suit different requirements for side-lobe minimization, dynamic range, and so forth. Commonly, many hardware efficient architectures are available for realizing FFT [3–5], but the same is not true for windowing–architectures. The conventional hardware implementation of window functions uses lookup tables which give rise to various area and time complexities with increase in word lengths. Moreover, they do not allow user-defined variations in the window length. An efficient implementation of flexible and reconfigurable window functions using CORDIC algorithm is suggested in [6, 7]. Though they allow user-defined variations in window length, latency is a major problem. The CORDIC algorithm [8–10] inherently suffers from latency issues and using two CORDIC processors in series, as is done in [6, 7]; the overall latency of the system is hampered.

In this paper, a new area-time efficient FPGA implementation to realize Blackman window function is suggested. We first redesign the conventional CORDIC algorithm to eliminate scale-factor compensation network and optimize its microrotation sequence identification. We then replace the linear CORDIC processors used in the existing design by shift-add tree derived using Booth multiplication. These modifications scale down the area consumption of the window architecture, with decrease in the number of pipeline stages.

The rest of the paper is structured as follows. Section 2 provides a comprehensive idea about various window functions and the conventional CORDIC algorithm. In Section 3, we propose a new CORDIC algorithm as redesigned-scale-free CORDIC. Section 4 deals with architecture for implementing the window functions. Section 5 presents the FPGA implementation and complexity issues, while Section 6 concludes the paper.

2. Background

2.1. Window Filtering Techniques. Window filtering is a well-known processing technique for limiting any signal to

short-time segment in various fields, like audio or video signal processing, communication systems, and so forth. The rectangular, Gaussian, Hamming, Hanning, Blackman-Harris, and Kaiser are some of the most common available windowing techniques [2, 11]. The selection of the available windows is based on the spectral characteristics desired by the applications. Equations (1a)–(1c) explains the Hanning, Hamming, and the Blackman window family as follows:

$$W_{\text{Hann}}(n) = 0.5 + 0.5\cos\left(\frac{2\pi n}{(N-1)}\right), \tag{1a}$$

where N is the window length.

$$W_{\text{Hamm}}(n) = \alpha + \beta\cos\left(\frac{2\pi n}{(N-1)}\right); \tag{1b}$$

where $\beta = 1 - \alpha$.

The values of α and β are determined to achieve maximum side-lobe cancellation. For Hamming window, the coefficients are calculated as $\alpha = 25/46$ and $\beta = 21/46$;

$$W_{\text{Blkman}}(n) = \alpha_0 + \alpha_1\cos\left(\frac{2\pi n}{(N-1)}\right) + \alpha_2\cos\left(\frac{4\pi n}{(N-1)}\right), \tag{1c}$$

where $\alpha_0 + \alpha_1 + \alpha_2 = 1$.

The Blackman Harris window has three degrees of freedom which can be used to design a family of window functions having different window amplitudes, roll-off rates, and side-lobe rejections. The Blackman window with coefficients $\alpha_0 = 0.42$, $\alpha_1 = 0.5$ and $\alpha_2 = 0.08$ has side-lobe roll off rate of 18 dB/octave and the worst case side-lobe level is about 58 dB; while with coefficients $\alpha_0 = 0.4243801$, $\alpha_1 = 0.4973406$ and $\alpha_2 = 0.0782793$ the side-lobe level is 71.48 dB with side-lobe roll off rate of 6 dB/octave.

The hardware implementation of window functions invlove trigonometric computations. The primitive technique to compute trigonometric functions uses LUTs. But this approach fails to support user-defined changes in the window-length. Another popular algorithm for computing trigonometric functions is known as CORDIC (coordinate rotation digital computer) algorithm. This algorithm is used in [6, 7] for efficient window implementation in hardware and to provide application dependent changes in the window length. It uses two serially connected CORDIC processors operating in different modes, one in linear and other in circular. Inherently, the CORDIC algorithm suffers from latency issues; and the design in [6, 7] operates two CORDIC processors in series, as a result the latency is the major drawback in the existing designs of [6, 7]. Therefore, we redesign the CORDIC algorithm to minimize the number of iterations and hence reduce latency. Moreover, we replace one of the CORDIC processors with a booth multiplication shift-add tree to further minimize latency and area.

2.2. CORDIC Algorithm. The conventional CORDIC algorithm [9, 10, 12, 13] has various modes of operation and trajectories. But as window functions use CORDIC algorithm in rotation mode following circular trajectory, we restrict our discussion to circular-rotation mode CORDIC only.

The coordinates of two vectors $\mathbf{V_A}[x_A, y_A]$ and $\mathbf{V_B}[x_B, y_B]$ separated by an angle "θ" are related as

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \mathbf{R_p} \cdot \begin{bmatrix} x_A \\ y_A \end{bmatrix}, \quad \mathbf{R_p} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{2}$$

Equation (2) forms the basic principle for iterative CORDIC coordinate calculations [8]. The key concept for realizing rotations using CORDIC algorithm is to express the desired rotation angle "θ" as an aggregation of predefined elementary angles, defined as:

$$\theta = \sum_{i=0}^{b} \mu_i \cdot \alpha_i, \tag{3}$$

where $\mu_i \in \{-1, 1\}$, $\alpha_i = \tan^{-1}2^{-i}$ and b is the word length.

The rotation matrix $\mathbf{R_p}$ in its original form (2) requires determining the sine and cosine values and four multiplication operations. Factoring the cosine term simplifies the rotation matrix (4) by converting the multiplication operations to shift, as the tangent of elementary angles are defined in the negative powers of two (3) as

$$\mathbf{R_p} = K_i \cdot \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix}, \quad K_i = \frac{1}{\sqrt{1 + 2^{-2i}}}. \tag{4}$$

The rotation matrix $\mathbf{R_p}$ in (4) is applicable for anticlockwise vector rotations. To support both clockwise and anticlockwise CORDIC rotations, the rotation matrix is altered as

$$\mathbf{R_p} = K_i \cdot \begin{bmatrix} 1 & -\mu_i \cdot 2^{-i} \\ \mu_i \cdot 2^{-i} & 1 \end{bmatrix}, \tag{5}$$

where $\mu_i = 1$ for anticlockwise rotations and $\mu_i = -1$ for clockwise rotations.

In its original form, the CORDIC algorithm suffers from major disadvantages like scale-factor compensation, latency, and optimal identification of micro-rotations. We propose a redesigned-scale-free CORDIC algorithm to overcome these disadvantages.

3. Redesigned-Scale-Free CORDIC Algorithm

The proposed CORDIC algorithm is an improved version of the conventional CORDIC algorithm in circular-rotation mode. The major ideas which lead to the proposed CORDIC algorithm are as follows: (i) redefine the elementary angles to eliminate the ROM required in conventional CORDIC algorithm to store the elementary angles, (ii) extend the Taylor series approximation of Scaling-Free CORDIC [13] to provide completely scale-free solution over the entire coordinate space, and (iii) obviate the redundant CORDIC iterations using new micro-rotation sequence identification. However, the existing scaling-free CORDIC [13] is outperformed by the conventional CORDIC beyond 20 bit implementation. But since an extensive set of applications work on word lengths up to 16 bits, our aim is to redesign the scaling-free CORDIC for word-length up to 16 bits.

Table 1: Mean square error in x-coordinate and y-coordinate for varying approximation orders.

Order of approximation	Mean square error	
	x-coordinate	y-coordinate
3	1.6881×10^{-7}	3.8830×10^{-7}
4	2.2539×10^{-7}	3.3926×10^{-7}
5	2.2360×10^{-7}	3.3404×10^{-7}
6	2.2343×10^{-7}	3.3413×10^{-7}

3.1. Redefining the Elementary Angles. We redefine the elementary angles used in conventional CORDIC (3) as

$$\alpha_i = 2^{-i}. \tag{6}$$

The above definition of elementary angles obviates the ROM required by the conventional CORDIC algorithm to store the elementary angles.

3.2. Coordinate Calculation Equations. We derive a new set of coordinate calculation equations by modifying the Taylor series expansion of sine and cosine functions used in scaling-free CORDIC [13]. Instead of using second order approximation of scaling-free CORDIC, we use third order of Taylor series approximation. It is necessary to analyze various orders of Taylor series approximation before third order is finalized for use in coordinate equations. We compare the mean square errors in the x-coordinate and y-coordinate for various orders of approximation in Table 1. The errors are calculated from the results obtained after simulating the CORDIC processors. The rotation matrix of the CORDIC processors was designed using the orders of approximation mentioned in Table 1 in $\mathbf{R_p}$ (2) and given by:

$$\begin{aligned}
\mathbf{R_{pc1}} &= \begin{bmatrix} 1-\frac{\alpha_i^2}{2!} & -\left(\alpha_i - \frac{\alpha_i^3}{3!}\right) \\ \alpha_i - \frac{\alpha_i^3}{3!} & 1-\frac{\alpha_i^2}{2!} \end{bmatrix}, \\
\mathbf{R_{pc2}} &= \begin{bmatrix} 1-\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!} & -\left(\alpha_i - \frac{\alpha_i^3}{3!}\right) \\ \alpha_i - \frac{\alpha_i^3}{3!} & 1-\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!} \end{bmatrix}, \\
\mathbf{R_{pc3}} &= \begin{bmatrix} 1-\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!} & -\left(\alpha_i - \frac{\alpha_i^3}{3!}+\frac{\alpha_i^5}{5!}\right) \\ \alpha_i - \frac{\alpha_i^3}{3!}+\frac{\alpha_i^5}{5!} & 1-\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!} \end{bmatrix}, \\
\mathbf{R_{pc4}} &= \begin{bmatrix} 1-\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!}-\frac{\alpha_i^6}{6!} & -\left(\alpha_i - \frac{\alpha_i^3}{3!}+\frac{\alpha_i^5}{5!}\right) \\ \alpha_i - \frac{\alpha_i^3}{3!}+\frac{\alpha_i^5}{5!} & 1-\frac{\alpha_i^2}{2!}+\frac{\alpha_i^4}{4!}-\frac{\alpha_i^6}{6!} \end{bmatrix}.
\end{aligned} \tag{7}$$

The errors are calculated for 16 bit word length, for angles lying in the range $[0, \pi/4]$, since for sine/cosine functions this range can be easily extended over the entire coordinate space using the octant wave symmetry. From Table 1, we conclude that the errors are of the same order for various orders of approximation of Taylor series expansion. Therefore, to keep the hardware complexity to minimum, we choose third order of approximation. Thus, the rotation matrix of the proposed CORDIC algorithm is given by

$$\mathbf{R_p} = \begin{bmatrix} 1-\frac{\alpha_i^2}{2!} & -\left(\alpha_i - \frac{\alpha_i^3}{3!}\right) \\ \alpha_i - \frac{\alpha_i^3}{3!} & 1-\frac{\alpha_i^2}{2!} \end{bmatrix}. \tag{8}$$

In order to implement the above rotation matrix using shift-add implementation, we approximate (3!) to 2^3. With this approximation, the mean square errors in the x-coordinate and y-coordinate are 1.5839×10^{-7} and 2.7664×10^{-7}, respectively. The errors are calculated for 16 bit word length, for angles lying in the range $[0, \pi/4]$ since for sine/cosine functions this range can be easily extended over the entire coordinate space using the octant wave symmetry. As these errors are of the same order as the errors in Table 1, this approximation does not affect the accuracy. Finally, the rotation matrix of the proposed CORDIC algorithm is defined as

$$\begin{aligned} &\text{For } \alpha_i = 2^{-r_i} \\ &\mathbf{R_p} = \begin{bmatrix} 1-2^{-(2r_i+1)} & -\left(2^{-r_i} - 2^{-(3r_i+3)}\right) \\ 2^{-r_i} - 2^{-(3r_i+3)} & 1-2^{-(2r_i+1)} \end{bmatrix}. \end{aligned} \tag{9}$$

3.3. Determination of Highest Elementary Angle. The use of Taylor series approximation imposes a restriction on the highest elementary angle being used in CORDIC iterations [13]. This restriction ensures that the higher order terms neglected due to the order of approximation used do not affect the accuracy of the processor. For third order of approximation, fourth and subsequent higher order terms should be zero after the shift operation of CORDIC so that their role in mathematical operations is obviated.

For a word length of N-bits, nth order term T_n is zero if it gets right shift by r-bits, defined as

$$\begin{aligned} T_n &= \frac{\alpha^n}{n!} \\ \text{for } \alpha = 2^{-r}, &\quad T_n = 2^{-(r\cdot n + \log_2 n!)} \\ \text{for } T_n \longrightarrow 0 &\quad r \cdot n + \log_2 n! \geq N \\ \Longrightarrow r &\geq \frac{N - \log_2 n!}{n}. \end{aligned} \tag{10}$$

For third order of approximation, $n = 4$, the smallest value of $r_{\min}$ and the highest permissible elementary angle are given by:

$$\begin{aligned} r_{\min} &= \left\lfloor \frac{N - \log_2 4!}{4} \right\rfloor. \\ \alpha_{\max} &= 2^{-r_{\min}} \end{aligned} \tag{11}$$

Thus, for 16 bit word length, $r_{\min} = 2$ and the highest elementary angle permissible is $\alpha_{\max} = 0.25$ radians.

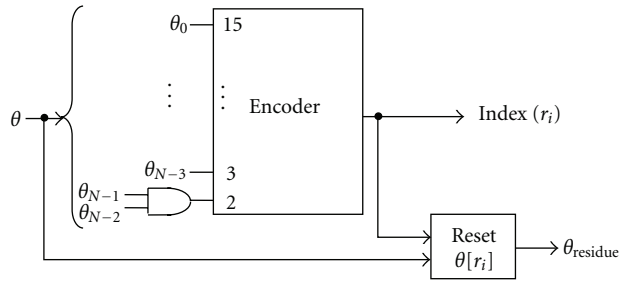

Figure 1: Micro-rotation sequence generator.

3.4. Micro-Rotation Sequence Identification. The proposed micro-rotation sequence generation is different from the conventional CORDIC micro-rotation identification. In conventional technique, each elementary angle is used only once; while we allow multiple micro-operations corresponding to the same elementary angle. Then, the use of every elementary angle is a must in conventional CORDIC, where as we have selective micro-rotations that depend on the angle of rotation. Further, we restrict the micro-rotations in single direction (anticlockwise) only as against bidirectional micro-rotations (clockwise and anticlockwise) in conventional CORDIC.

The micro-rotation sequence generator selects appropriate elementary angle for the current CORDIC iteration. Using the redefined elementary angles (6), the micro-rotations can be identified using the circuit shown in Figure 1. It comprises of a priority encoder and a reset circuitry. The input to the micro-rotation sequence generator is the rotation angle $\theta[N-1:0]$, where N is the word length. The priorities of the encoder are hooked in the reverse order with θ_{N-1} having the highest priority and θ_0 the least. The reset circuitry resets a bit of the input rotation angle to generate the residue angle for next CORDIC iteration. Since, the micro-rotation sequence generates the shift-index r_i for one CORDIC iteration, it is required in every stage of the CORDIC pipeline (the implementation of CORDIC stage is discussed in the forthcoming sections). The micro-rotation sequence generation block handles the angles in the range $[0, \pi/4]$. This range can be extended to the entire coordinate space using the octant symmetry of sine and cosine functions [14].

3.5. Number of Iterations. The number of iterations required to realize this range $[0, \pi/4]$ of rotation angles is decided based on: (i) maximum iterations of the highest elementary angle and (ii) the iterations of the other elementary angles. The rotation angle θ is given by

$$\theta = n_1 \cdot \alpha_{\max} + \sum_{r_i}^{n_2 \text{iterations}} \alpha_{r_i}, \tag{12}$$

where $r_i \neq r_{\min}$ and $\alpha_{r_i} = 2^{-r_i}$.

The maximum angle that can be handled by the micro-rotation sequence generator is $\pi/4 \approx 0.785$ radians. Therefore, no more than 3 iterations of highest elementary angle ($\alpha_{\max} = 2^{-2} = 0.25$ radians) is required, that is, maximum of $n_1 = 3$ iterations are required to realize any angle of rotation in the range $[0, \pi/4]$. The rest n_2 iterations determine accuracy. To select an appropriate value of n_2, we simulate the CORDIC processor for varying n_2 iterations, the mean square error is tabulated in Table 2. After observing the errors in Table 2, we can say that the errors for $n_2 = 4$ and $n_2 = 5$ are of same order. Therefore, to minimize the number of CORDIC iterations, we select $n_2 = 4$. We require a maximum of $n_1 + n_2 = 3 + 4 = 7$ iterations for the proposed CORDIC processor.

Table 2: Mean square error in x-coordinate and y-coordinate for varying n_2 iterations.

Number of n_2 iterations	Mean square error	
	x-coordinate	y-coordinate
3	4.3028×10^{-6}	2.9493×10^{-6}
4	1.6412×10^{-7}	2.6070×10^{-7}
5	1.0640×10^{-7}	2.1940×10^{-7}

3.6. Error Analysis. The error analysis of the proposed CORDIC algorithm is divided into two parts: (i) residue angle error and (ii) error in the coordinate values.

3.6.1. Residue Angle Error. In the proposed methodology, desired angle of rotation is expressed as

$$\theta = \sum^{\text{for } n \text{ iteration}} 2^{-r_i}, \quad r_{\min} \leq r_i \leq (N-1), \tag{13}$$

where $r_{\min}$ is minimum shift-index (11), N is word length.

We identify the micro-rotations by using the bit representation of the desired rotation angle. The residue angle error depends on the number of bits set in the radix-2 representation of the rotation angle and varies for different rotation angles. Therefore, we derive the worst-case angle error in the range of convergence $[0, \pi/4]$.

The maximum number of iterations is fixed for all rotation angles. The input rotation `angle` with the MSB-nibble value of 4′b1011, requires four iterations of $r_{\min} = 2$, while, three or less iterations are required for other MSB-nibble values. From second MSB-nibble onwards each bit set to 1′b1 in the radix-2 representation of the rotation angle would require one iteration; therefore, maximum four iterations are required if the second MSB nibble value is 4′b1111. Since the iteration count is seven, the worst-case error is $(2^{-7} - 2^{-16})$. This worst-case residue angle error is specific to the rotation angle of 16′b1011_1111_1111_1111, while for other rotation angles the residue angle error will be less. In the proposed 16 bit fixed point representation scheme, 16′b1011_1111_1111_1111 is 42.97°; the worst-case residue angle error is 0.4467°.

3.6.2. Error in Coordinate Values. For fixed-point implementation, the error is represented in terms of bit-error position (BEP). The BEP in x and y coordinates calculated using the proposed CORDIC processor is shown in Figure 2. For a BEP of n, the conventional CORDIC requires a word length of $n + \log_2 n + 2$-bits [15]. For a BEP of 10 bits as achieved by the

proposed CORDIC algorithm, the conventional CORDIC will require 16 bit word length. We, therefore, compare the proposed design with the existing design using conventional CORDIC processor [7] for 16 bits.

4. Architecture for Implementing Window Functions

In this section, we focus on implementing the pipelined architecture to generate window functions. The length of the window function is selected by the user at run time. Currently, the architecture implements the Blackman window, but with slight modifications it can be extended to other window functions as well. In the proposed architecture, the output bit width is set to 16 bits.

Figure 3 shows the block diagram for generating Blackman window function. The circuit consists of theta generator unit (TGU), window coefficient multiplier (WCM), circular CORDIC processor (CCP) and FIFO. The TGU generates the two angle values ($\theta = 2\pi n/(N-1)$) and $2\theta = 4\pi n/(N-1)$ required in the three-term Blackman window function. WCM multiplies the input signal samples with the window constants using a shift-add tree derived from Booth multiplication algorithm. CCP is used for generating the cosine terms in the window function. The FIFO is used for proper synchronization between the window coefficients having cosine terms and constants.

4.1. Circular CORDIC Processor (CCP). The CCP is pipelined implementation of the proposed redesigned-scale-free CORDIC algorithm discussed in Section 3. A total of seven ($n_1 = 3$ and $n_2 = 4$) iterations are required (as discussed in Section 3.5), since each pipeline stage performs one iteration, the proposed CCP-pipeline is seven stages long. Each stage (Figure 4) is a combination of three blocks (i) the coordinate calculation unit, (ii) the shift-index calculation, and (iii) the micro-rotation sequence generation. The coordinate calculation unit implements (9) using shift-add implementation. The shift-index calculation computes the necessary shifts ($(2r_i+1)$ and $(3r_i+3)$), required by the coordinate calculation unit. The micro-rotation sequence generation is shown in Figure 1.

The complexity of coordinate calculation unit is equal to six N bit logic shifters and six N bit adder/subtractor. The shift-index calculation unit requires three $(\log_2 N + P)$-bit adders, where P are the extra bits required to store the sum. Even though, the coordinate calculation unit of the proposed redesigned-scale-free CORDIC is more complex than the conventional CORDIC [8]; the overall gate count of the proposed window architecture using the proposed CCP-pipeline is reduced.

4.2. Window Coefficient Multiplier (WCM). The WCM unit multiplies the input samples with the Blackman window coefficients (α_0, α_1, and α_2). The shift-add tree for multiplication with α_0, α_1, and α_2 is derived using the Booth multiplication algorithm. In radix-2 representation system, multiplication with 0.5 is equivalent to single right shift.

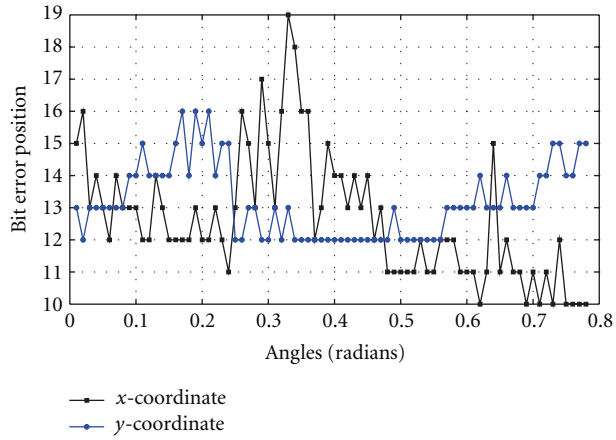

Figure 2: Bit error position in coordinate values calculated using the proposed CORDIC algorithm.

Therefore, multiplication with $\alpha_1 = 0.5$ is realized using a hardwired shifter. The coefficient α_0 is represented in 16 bit fixed-point format as 0001_1011_0010_0000, that requires four 16 bit adders and five hardwired shifters, while α_2 is represented as 0000_0101_0000_0010 and requires two 16 bit adders and three hardwired shifters.

The complexity of the WCM unit is equivalent to six 16 bit adders, as hardwired shifters do not incur any hardware costs.

4.3. Theta Generator Unit (TGU). The TGU generates the two angles given by

$$\theta = \frac{2\pi n}{(N-1)}, \qquad 2\theta = \frac{4\pi n}{(N-1)}, \tag{14}$$

where N is a multiple of 2 such that $N = 2^M$.

The difference between the consecutive values of θ is given by

$$\begin{aligned} \Delta\theta &= \theta_{n+1} - \theta_n \\ \Delta\theta &= \frac{2\pi}{(N-1)}, \end{aligned} \tag{15a}$$

$$\text{For } N = 2^M, \qquad \Delta\theta = \frac{2\pi}{(2^M-1)}, \tag{15b}$$

$$\Longrightarrow \Delta\theta = \frac{2\pi}{2^M} \cdot \left(1 - 2^{-M}\right)^{-1}. \tag{15c}$$

Using binomial theorem (B.T.), we simplify (15c) to the following:

$$\begin{aligned} &\text{using B.T.} (1-x)^{-1} = 1 + x + x^2 + x^3 + \cdots \\ &\text{B.T. of} \Longrightarrow \left(1 - 2^{-M}\right)^{-1} = 1 + 2^{-M} + 2^{-2M} + \cdots, \end{aligned} \tag{16a}$$

$$\Delta\theta = \frac{2\pi}{2^M} \cdot \left(1 + 2^{-M} + 2^{-2M}\right), \tag{16b}$$

$$\begin{aligned} \Longrightarrow \Delta\theta &= \frac{2\pi}{2^M} + \frac{2\pi}{2^{2M}} + \frac{2\pi}{2^{3M}} \\ \Longrightarrow \Delta\theta &= (2\pi \gg M) + (2\pi \gg 2M) + (2\pi \gg 3M). \end{aligned} \tag{16c}$$

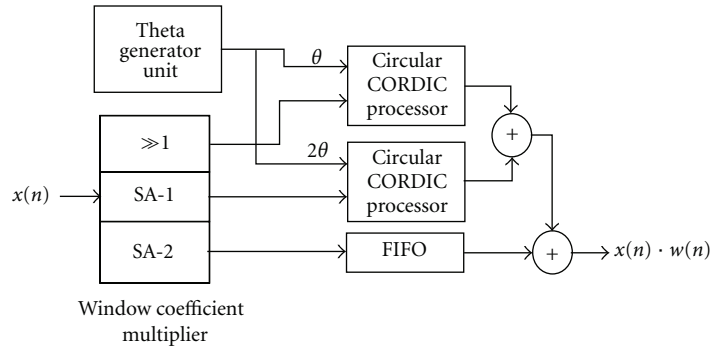

Figure 3: Block diagram for generating blackman window function.

Generally in most signal processing applications, not less than 16-point DFT is used which implies $N \geq 16$ and $M \geq 4$. Therefore, only three terms of binomial expansion are sufficient for 16 bit accuracy as follows:

$$\text{fourth term of B.T. in (16c) } \Delta\theta_4 = \frac{2\pi}{2^{4M}} \implies \Delta\theta_4 = 2\pi \gg 4M. \quad (17)$$

For 16 bit word length and $M \geq 4$, the term $\Delta\theta_4$ always gets a right shift greater than or equal to 16. Therefore, $\Delta\theta_4$ is zero for 16 bit word length.

Figure 5 shows the block diagram representation of TGU. The angles in the windowing function are uniformly distributed over the entire coordinate space. The CCP unit handles angles in the range of $[0, \pi/4]$. Therefore, the TGU divides the entire coordinate space into octants, so that the input angle to CCP always lies in the range $[0, \pi/4]$. The octants are distinguished as shown in Figure 6; the TGU also generates signals for proper octant mapping of values generated by CCP.

The TGU requires three 16 bit adders, two barrel-shifters and one encoder.

4.4. Window Generation. In Figure 7, we compare the Blackman window generated using the proposed processor with that of MATLAB inbuilt function blackman() for $N = 32$.

5. FPGA Implementation and Complexity Issues

The proposed architecture is coded in Verilog and simulated and synthesized using Xilinx ISE 9.2i Design Suite to be mapped on Xilinx Virtex 2Pro (XC2VP50-6FF1148) device. For 16 bit implementation, the proposed design consumes 1800 slices and 3371 4-Input LUTs, with a maximum operating frequency of 101.284 MHz. The total delay of 9.873 nsec is distributed as 58.7% logic delay and 41.3% route delay. The total gate count of the proposed design is 34739.

5.1. Comparison with Existing Architecture. The CORDIC processor both linear and circular used in [7] is designed using conventional CORDIC algorithm. The scaling-free CORDIC [13] and enhanced scaling-free CORDIC [16] are

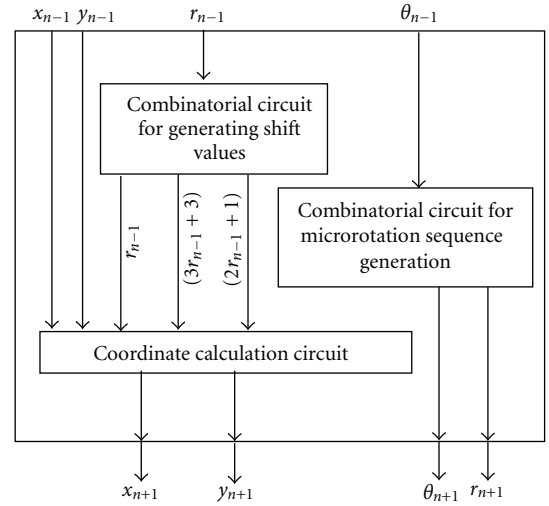

Figure 4: Pipeline stage of new redesigned CORDIC processor.

currently the best available hardware designs for circular CORDIC implementation. We compare our processor with three designs: (i) the existing design in [7] using conventional circular CORDIC, (ii) replace the conventional circular CORDIC in [7] with scaling-free CORDIC [13], and (iii) replace the conventional circular CORDIC in [7] with enhanced scaling-free CORDIC [16]. The area complexity and latency of the proposed design with three variants of existing design [7] mentioned above are compared in Table 3.

5.1.1. Area Comparison. The area of conventional circular CORDIC processor is calculated using Xilinx CORDIC IP v3.0. The Xilinx CORDIC Core is optimized for circular CORDIC computation with maximum pipelining for 16 bit word length. The gate count is 20122. In [13], the complexity of 16 bit scaling-free CORDIC is computed to be equivalent to 1000 1 bit full adders and 597 1 bit registers. This area complexity approximately uses 16776 gates for implementation. The SFB4C architecture of enhanced scaling-free (ESF) CORDIC [16] replaces the initial four scaling-free CORDIC iterations with conventional CORDIC iterations. Thus, the complexity of 16 bit ESF CORDIC without scale-factor compensation is equivalent to 512 1 bit full adders and 420 1 bit registers, approximately equal to 9504 gates.

Table 3: Complexity comparison of proposed design with existing design variants.

	Existing design		% Savings	
Circular CORDIC	Area	Latency[1]	Area[2]	Latency[3]
Conventional	73585	32	52.79	68.75
Scaling-free [13]	66273	28	47.58	64.28
ESF-CORDIC [16]	51264	25	32.23	60

[1] Latency is defined in terms of number of pipelining stages required by the design.
[2] The gate count of the proposed design is 34739.
[3] The latency of the proposed design is 10.

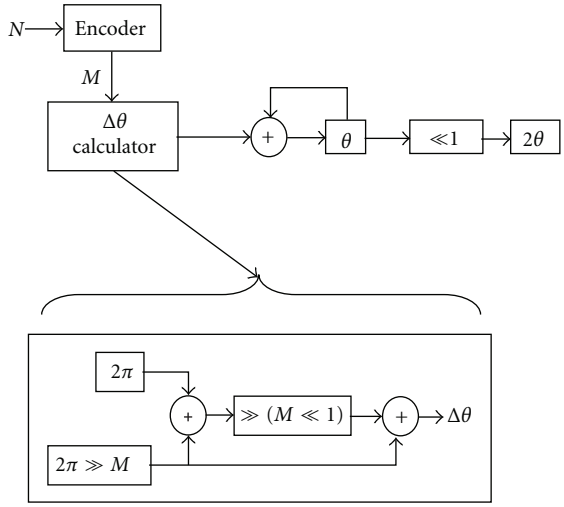

Figure 5: Theta generator unit.

The complexity of the 16 bit linear conventional CORDIC is equivalent to 512 1 bit full adders and 768 1 bit registers, approximately equal to 12288 gates. The other units like theta generator unit, FIFO, and adders required for realizing the window processor are common for the proposed as well as the existing design.

5.1.2. Latency. The throughput of all the designs is same, that is, one data sample per clock cycle, while the latency is different and is closely related to number of iterations in circular CORDIC and linear CORDIC processor when the designs are operating at the same clock frequency. The 16 bit linear CORDIC processor uses 16-stages long pipeline. The conventional circular CORDIC processor again uses 16-stages pipeline for 16 bit word length. For the same 16 bit word length, the scaling-free CORDIC [13] processor uses 12-stages long pipeline, while the ESF CORDIC [16] pipeline is 9 stages long. Therefore, the latency of existing design in [7] with conventional circular CORDIC is 32 stages, while with scaling-free is 28 stages and with ESF-CORDIC is 25 stages.

The new redesigned-CORDIC pipeline is 7 stages long (Section 4.1). The delay of the WCM unit (Section 4.2) is three adders in serial, which can be considered equivalent to three linear CORDIC iterations. Hence, the total latency of the proposed design is 10 stages, which is far less as compared

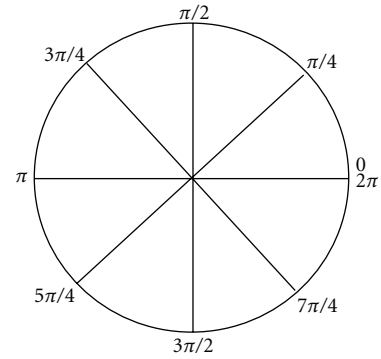

Figure 6: Mapping of coordinate space into octants.

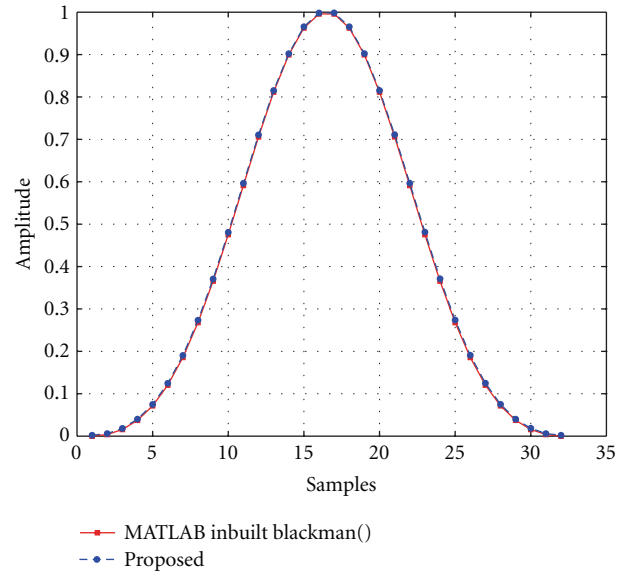

Figure 7: Comparison of proposed window implementation with MATLAB inbuilt blackman() function for $N = 32$.

to existing design using the best of the available circular CORDIC hardware.

5.1.3. Delay. The delay is the time required to generate one set of window coefficients for a window length of N when the design is operating at the maximum clock frequency. The critical path for the proposed design is the TGU. Since the existing design using the conventional circular CORDIC and the scaling-free circular CORDIC also work using the same TGU, while, the existing design using the ESF-CORDIC uses a slightly less complex TGU as compared to other designs. The TGU for the proposed design and the existing design using the conventional circular CORDIC and the scaling-free circular CORDIC generates angle in the range $[-\pi/4, \pi/4]$, while for existing design using the ESF-CORDIC the TGU generates angles in the range $[-\pi/2, \pi/2]$. Therefore, the maximum clock frequency for the existing design using ESF-CORDIC is 101.983 MHz and for other designs including the proposed design is 101.284 MHz. Figure 8 compares the delay for the four designs for various window lengths.

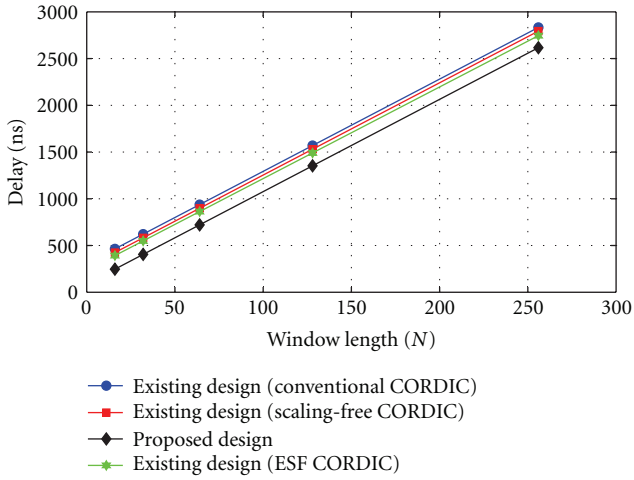

FIGURE 8: Comparison of delay of the proposed design with existing designs operating at maximum clock frequency.

6. Conclusion

In this paper, we present an area-time efficient CORDIC based processor for realizing window functions. Currently, the architecture implementing the Blackman window function, with slight modification, can be extended to other window functions as well. We also propose a circular CORDIC processor for word lengths up to 16 bits. The redesigned scale-free CORDIC processor uses third order of approximation of Taylor series to realize scale-free CORDIC iterations. However, removal of scaling factor comes with the disadvantage of complex coordinate calculations. The micro-rotation sequence generation is optimized using a priority encoder which reduces the total CORDIC processor pipeline to seven stages. A shift-add tree derived using Booth multiplication algorithm replaces the linear CORDIC processor in the original design of window architecture. The proposed Blackman window architecture saves approximately 44.2% area and drastically reduces latency with no affect on accuracy.

References

[1] K. K. Parhi, *VLSI Digital Signal Processing Systems*, John Wiley & Sons, 1999.

[2] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing Principles, Algorithms and Applications*, Prentice Hall, 3rd edition, 2006.

[3] A. M. Despain, "Fourier transform computers using CORDIC iterations," *IEEE Transactions on Computers*, vol. 23, no. 10, pp. 993–1001, 1974.

[4] T. Sansaloni, A. Pérez-Pascual, and J. Valls, "Area-efficient FPGA-based FFT processor," *Electronics Letters*, vol. 39, no. 19, pp. 1369–1370, 2003.

[5] M. Ayinala, M. Brown, and K. K. Parhi, "Pipelined parallel FFT architectures via folding transformation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 6, pp. 1068–1081, 2011.

[6] K. C. Ray and A. S. Dhar, "CORDIC-based unified VLSI architecture for implementing window functions for real time spectral analysis," *IEE Proceedings: Circuits, Devices and Systems*, vol. 153, no. 6, pp. 539–544, 2006.

[7] K. C. Ray and A. S. Dhar, "High throughput VLSI architecture for Blackman windowing in real time spectral analysis," *Journal of Computers*, vol. 3, no. 5, pp. 54–59, 2008.

[8] J. E. Volder, "The cordic trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. 8, no. 3, pp. 330–334, 1959.

[9] J. S. Walther, "A unified algorithm for elementary functions," in *Proceedings of the 38th Spring Joint Computer Conferences*, pp. 379–385, Atlantic City, NJ, USA, 1971.

[10] P. K. Meher, J. Valls, T. B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: algorithms, architectures, and applications," *IEEE Transactions on Circuits and Systems I*, vol. 56, no. 9, pp. 1893–1907, 2009.

[11] J. O. Smith III, *Spectral Audio Signal Processing*, W3K, 2011.

[12] T. B. Juang, S. F. Hsiao, and M. Y. Tsai, "Para-CORDIC: parallel CORDIC rotation algorithm," *IEEE Transactions on Circuits and Systems I*, vol. 51, no. 8, pp. 1515–1524, 2004.

[13] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 11, pp. 1463–1474, 2005.

[14] J. Vankka, *Digital Synthesizers and Transmitters for Software Radio*, Springer, Dordrecht, Netherlands, 2005.

[15] K. Kota and J. R. Cavallaro, "Numerical accuracy and hardware tradeoffs for CORDIC arithmetic for special-purpose processors," *IEEE Transactions on Computers*, vol. 42, no. 7, pp. 769–779, 1993.

[16] F. J. Jaime, M. A. Sánchez, J. Hormigo, J. Villalba, and E. L. Zapata, "Enhanced scaling-free CORDIC," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 7, pp. 1654–1662, 2010.