Transactions Briefs

Concept, Design, and Implementation of Reconfigurable CORDIC

Supriya Aggarwal, Pramod K. Meher, and Kavita Khare

***Abstract*— This brief presents the key concept, design strategy, and implementation of reconfigurable coordinate rotation digital computer (CORDIC) architectures that can be configured to operate either for circular or for hyperbolic trajectories in rotation as well as vectoring-modes. It can, therefore, be used to perform all the functions of both circular and hyperbolic CORDIC. We propose three reconfigurable CORDIC designs: 1) a reconfigurable rotation-mode CORDIC that operates either for circular or for hyperbolic trajectory; 2) a reconfigurable vectoring-mode CORDIC for circular and hyperbolic trajectories; and 3) a generalized reconfigurable CORDIC that can operate in any of the modes for both circular and hyperbolic trajectories. The reconfigurable CORDIC can perform the computation of various trigonometric and exponential functions, logarithms, square-root, and so on of circular and hyperbolic CORDIC using either rotation-mode or vectoring-mode CORDIC in one single circuit. It can be used in digital synchronizers, graphics processors, scientific calculators, and so on. It offers substantial saving of area complexity over the conventional design for reconfigurable applications.**

***Index Terms*— Circular trigonometry, coordinate rotation digital computer (CORDIC), hyperbolic trigonometry, reconfigurable CORDIC.**

I. Introduction

The coordinate rotation digital computer (CORDIC) algorithm involves a simple shift-add iterative procedure to perform several computing tasks by operating in either rotation-mode or vectoring-mode following any one among linear, hyperbolic, and circular trajectories [1]. Applications such as singular value decomposition, eigenvalue estimations, QR decomposition, phase and frequency estimations, synchronization in digital receivers, 3-D graphics processor, and interpolators require the CORDIC to operate in both rotation and vectoring-modes. The 3-D structures such as hyperboloids, paraboloids, and ellipsoids require the CORDIC to be operated in both circular and hyperbolic trajectories. The hardware implementation of these applications requires more than one CORDIC processor operating in different modes and different trajectories. A reconfigurable CORDIC, which can operate in rotation and vectoring-modes, for both circular and hyperbolic trajectories can replace multiple CORDIC processors, and would be highly useful for such applications. A reconfigurable CORDIC can be utilized for a variety of applications in communication systems, signal processing, 3-D graphics, robotics apart from general scientific calculations, and waveform generations.

In the last five decades, several algorithms have been proposed for area-delay-efficient and power-efficient implementation of CORDIC algorithms, either for circular trajectory [2]–[7] or for hyperbolic trajectory [8]–[10]. But, we do not find any systematic study on design and implementation of reconfigurable CORDIC in the existing literature. A basic design of reconfigurable CORDIC based on a unified CORDIC algorithm [11] has been proposed recently [12]. The reconfigurable design of [12] is found to involve high reconfiguration overhead and results in low hardware utilization efficiency. Therefore, in this brief, we present a methodology for the design of reconfigurable CORDIC to be used for rotation-mode and vectoring-mode in circular and hyperbolic trajectories.

The rest of this brief is structured as follows. Section II deals with a brief overview of the CORDIC algorithm. In Section III, we explore the possibility and difficulties in the design and implementation of reconfigurable CORDIC. The design strategy for a reconfigurable CORDIC is presented in Section III-B. We have derived the proposed reconfigurable CORDIC architectures in Section IV. The field-programmable gate array (FPGA) and application-specified integrated circuit (ASIC) implementations along with complexity and performance considerations of reconfigurable CORDIC are discussed in Section V.

II. Unified CORDIC Algorithm

The rotation-mode CORDIC determines the coordinates of any given vector after rotation through a given angle, while in the vectoring-mode it computes the magnitude and phase of the vector. The unified algorithm for linear and hyperbolic CORDICs is an extension of the basic CORDIC algorithm for circular trajectory. It is based on the generalized principle proposed in [11] to include hyperbolic and linear trajectories along with the original circular trajectory of operation. A variable m is introduced to modify the rotation matrix and elementary angles as

$$\mathbf{R_i} = K_i \cdot \begin{bmatrix} 1 & -m \cdot \mu_i \cdot 2^{-i} \\ \mu_i \cdot 2^{-i} & 1 \end{bmatrix} \tag{1}$$

$$\alpha_i = \left(\frac{1}{\sqrt{m}}\right) \tan^{-1}(\sqrt{m} \cdot 2^{-i}), \quad K_i = \frac{1}{\sqrt{1 + m \cdot 2^{-2i}}} \tag{2}$$

where $m = 1$ for circular, $m = 0$ for linear, and $m = -1$ for hyperbolic trajectories, respectively.

To guarantee the convergence in hyperbolic mode, the iterations $i = 4, 13, 40, \ldots$ need to be executed twice. It is shown that the scale-factor converges to a constant $K = 1.2075$ (0.60725) for hyperbolic (circular) trajectory [1]. Consequently, we have different scale-factors for the circular and hyperbolic trajectories. The unified CORDIC algorithm supports a range of convergence (RoC) of [−99°, 99°] for circular trajectory, while the RoC for hyperbolic trajectory is $|\theta| \leq 1.1182$ radians (nearly 64°).

A. Realization of Vectoring-Mode CORDIC

In vectoring-mode, the phase and magnitude of any given vector are computed by aligning the vector along x-axis. This is achieved by performing microrotations in the direction, which drives the y-coordinate to zero. Equations (1) and (2) are extended to vectoring-mode by changing the control variable that determines the direction. In rotation-mode, the residual angle θ_i acts as the control variable for determining the direction of microrotations, while in vectoring-mode, the control variable is y-coordinate. If y_i is positive, the direction of microrotation is clockwise, while it is anticlockwise if y_i is negative.

Manuscript received November 21, 2014; revised March 23, 2015 and May 11, 2015; accepted June 9, 2015. Date of publication July 8, 2015; date of current version March 18, 2016.

S. Aggarwal and K. Khare are with the Department of Electronics and Communication Engineering, National Institute of Technology, Bhopal 462003, India (e-mail: sups.aggarwal@gmail.com; kavita_khare1@yahoo.co.in).

P. K. Meher is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: aspkmeher@ntu.edu.sg).

Digital Object Identifier 10.1109/TVLSI.2015.2445855

III. Design Exploration of a Reconfigurable CORDIC

To design a reconfigurable CORDIC architecture with minimum reconfiguration overhead, we need to maximize the sharing of common hardware circuit in different configurations. Therefore, to explore the possibility of reconfigurable CORDIC, we examine, here, the commonalities in three main issues of CORDIC implementation, namely: 1) the coordinate-rotation matrix; 2) selection of elementary angles; and 3) direction of microrotations.

A. Reference Reconfigurable CORDIC

A basic design for reconfigurable CORDIC based on unified CORDIC algorithm was proposed in [12]. The major concern with the design of conventional reconfigurable architecture is the incompatibility in RoC of circular and hyperbolic trajectories. The RoC of circular CORDIC is [−99°, 99°], while that of hyperbolic CORDIC is given by $|\theta| \leq 1.1182$ radians. This limits the maximum angle of rotation of the reconfigurable design to 64°. The incompatible RoC of circular and hyperbolic CORDICs makes it difficult to implement them in the same circuit to perform rotation through [−180°, 180°].

Another major issue with the conventional reconfigurable CORDIC is scaling. We need to have two different scaling circuits for circular and hyperbolic CORDIC, and select the output from one of the scaling circuits depending on the selection of trajectory of operation.

B. Design Strategy for Proposed Reconfigurable CORDIC

As discussed in Section III-A, the circular and hyperbolic CORDICs require two different scaling circuits, which is quite costly. Therefore, it is necessary to use a scale-free implementation in the reconfigurable CORDIC. Here, we discuss the scaling-free CORDIC and its limitations, followed by the discussions on our design strategy for a reconfigurable CORDIC.

1) Scaling-Free CORDIC Algorithm and Its Limitations: The scaling-free CORDIC [2] employs second-order Taylor series approximation, where the rotation matrix is given by

$$\mathbf{R_i} = \begin{bmatrix} 1 - 2^{-(2i+1)} & -2^{-i} \\ 2^{-i} & 1 - 2^{-(2i+1)} \end{bmatrix}. \tag{3}$$

This approximation imposes a restriction on the basic-shift[1] $i = \lfloor (b - 2.585/3) \rfloor$. For 16-bit applications, the basic-shift is $i = 4$, which reduces the RoC to 7.16°, which can be extended to 22.5° using multiple iterations corresponding to the basic-shift $i = 4$. This is a major drawback, which limits the applicability of this algorithm.

Moreover, the algorithms in [2] and [3] focus only on circular rotation-mode, which cannot be directly extended to hyperbolic CORDIC, since the second order of approximation of Taylor series expansion of hyperbolic functions results in a very low RoC (nearly 22.5°). Due to the lack of symmetry in hyperbolic functions, the RoC cannot be extended to the entire coordinate space.

2) Reconfigurability of Rotation-Mode CORDIC: In [7] and [10], scaling-free algorithms for circular and hyperbolic trajectories are proposed. Moreover, in both the scaling-free algorithms, third order of approximation of Taylor series is used to derive the CORDIC rotation-matrices, as

$$\mathbf{R_{ci}} = \begin{bmatrix} 1 - 2^{-(2s_i+1)} & -(2^{-s_i} - 2^{-(3s_i+3)}) \\ 2^{-s_i} - 2^{-(3s_i+3)} & 1 - 2^{-(2s_i+1)} \end{bmatrix} \tag{4a}$$

$$\mathbf{R_{hi}} = \begin{bmatrix} 1 + 2^{-(2s_i+1)} & 2^{-s_i} + 2^{-(3s_i+2)} \\ 2^{-s_i} + 2^{-(3s_i+2)} & 1 + 2^{-(2s_i+1)} \end{bmatrix}. \tag{4b}$$

[1]The minimum possible permissible shifts in the CORDIC iteration have been termed as basic-shift, which is equal to the number of right shifts in the first CORDIC iteration.

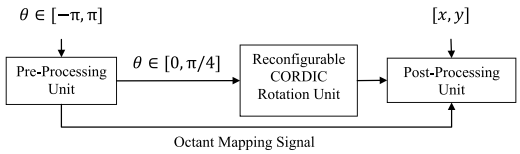

Fig. 1. Proposed reconfigurable rotation-mode CORDIC processor.

Note that the same set of elementary angles is used for both circular and hyperbolic rotation-modes. This is a big advantage to derive the reconfigurable CORDIC, since no differentiation is required to identify the microrotations according to the trajectories. For circular and hyperbolic trajectories, the elementary angles are redefined as

$$\alpha_i = 2^{-s_i} \tag{5}$$

where s_i is the number of shifts for the ith iteration.

The RoC for both the trajectories is compatible and extends to the entire coordinate space. The design for rotation-mode CORDIC with slight modification can be extended to support vectoring-mode as discussed below.

3) Reconfigurability of Vectoring-Mode CORDIC: To realize a vectoring-mode CORDIC, all the microrotations will be performed in the clockwise direction for both the circular and hyperbolic trajectories. The rotation matrices are given by

$$\mathbf{R_{ci}} = \begin{bmatrix} 1 - 2^{-(2s_i+1)} & 2^{-s_i} - 2^{-(3s_i+3)} \\ -(2^{-s_i} - 2^{-(3s_i+3)}) & 1 - 2^{-(2s_i+1)} \end{bmatrix} \tag{6a}$$

$$\mathbf{R_{hi}} = \begin{bmatrix} 1 + 2^{-(2s_i+1)} & -(2^{-s_i} + 2^{-(3s_i+2)}) \\ -(2^{-s_i} + 2^{-(3s_i+2)}) & 1 + 2^{-(2s_i+1)} \end{bmatrix} \tag{6b}$$

where s_i is the shift-index ith iteration.

The sign-bit of the y-coordinate over successive iterations determines the angle of rotation θ. For vectoring-mode, the maximum angle of rotation that can be computed lies in the range $[0, \pi/4]$. However, this range can be extended to the entire coordinate space using octant wave symmetry of sine and cosine functions for circular trajectory.

IV. Proposed Reconfigurable CORDIC

As seen in Sections III-B2 and III-B3, the coordinate calculation matrices for circular and hyperbolic CORDICs differ by the sign of operands, and to realize that additions are to be replaced by subtractions and vice-versa. This can be easily realized by a reconfigurable add/subtract circuit. In both cases, the basic-shift could be either 2 or 3, but the number of microrotations vary with the mode of operation. Besides, each case will have its own circuit to enable the extension of RoC. Based on these observations, we design three reconfigurable CORDIC architectures: 1) rotation-mode reconfigurable CORDIC; 2) vectoring-mode reconfigurable CORDIC; and 3) generalized reconfigurable CORDIC.

A. Rotation-Mode Reconfigurable CORDIC

The proposed design for reconfigurable rotation-mode CORDIC (shown in Fig. 1) consists of three parts: 1) preprocessing unit; 2) reconfigurable CORDIC rotation unit; and 3) postprocessing unit. The preprocessing unit ensures that the input rotation angle to the CORDIC processing structure always lies in the range $[0, \pi/4]$, as the maximum rotation angle that can be handled by microrotation sequence generator is $\pi/4$. The postprocessing unit is required only for circular trajectory to swap/complement the sine/cosine values depending on the octant of the rotation angle. The user can control the trajectory of the reconfigurable CORDIC by changing a 1-bit signal T. The rotation matrix for reconfigurable rotation-mode CORDIC is obtained after unifying the rotation matrices of circular

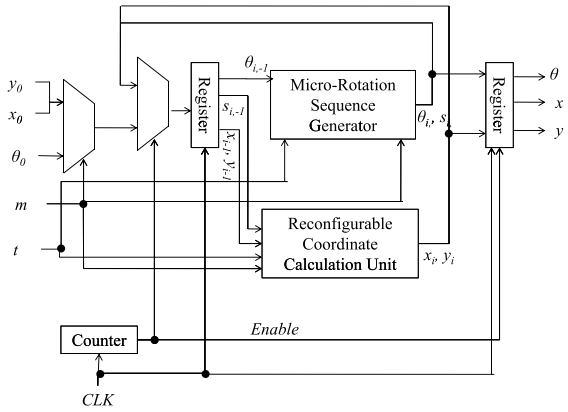

Fig. 2. Structure of the proposed reconfigurable recursive CORDIC architectures.

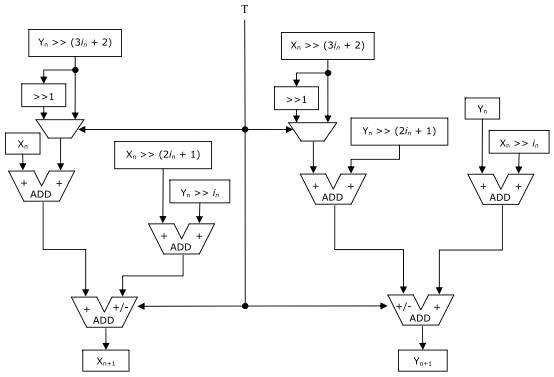

Fig. 3. RCCU for recursive design.

and hyperbolic case given by (4a) and (4b), respectively, as

$$\mathbf{R_i} = \begin{bmatrix} 1 \pm 2^{-(2s_i+1)} & \pm(2^{-s_i} \pm 2^{-(3s_i+2+T)}) \\ 2^{-s_i} \pm 2^{-(3s_i+2+T)} & 1 \pm 2^{-(2s_i+1)} \end{bmatrix}$$

where

$$T = \begin{cases} 0 & \text{for hyperbolic} \\ 1 & \text{for circular.} \end{cases} \tag{7}$$

1) Proposed Recursive Architecture: The recursive architecture (shown in Fig. 2) uses a single CORDIC microrotator to perform all the CORDIC iterations. The circular CORDIC of [7] requires one iteration less than the hyperbolic CORDIC of [10], but here we realize the architecture for the same number of iterations (eight for $s_{\text{basic}} = 2$ and eleven for $s_{\text{basic}} = 3$) for both circular and hyperbolic trajectories. The reconfigurable coordinate calculation unit (RCCU) is shown in Fig. 3.

2) Proposed Pipelined Architecture: Fig. 4 shows the reconfigurable CORDIC rotation unit for basic-shift 2. The shift-index s_i is fixed in every RCCU, and hence the shifters are hardwired and do not involve high complexity barrel-shifters. The implementation of RCCUs varies according to the basic-shift s_i. With slight modifications, the pipeline can be extended for basic-shift 3.

B. Reconfigurable Vectoring-Mode CORDIC

The reconfigurable rotation matrix for vectoring mode is obtained by unifying (6a) and (6b), as

$$\mathbf{R_i} = \begin{bmatrix} 1 \pm 2^{-(2s_i+1)} & \mp(2^{-s_i} \pm 2^{-(3s_i+2+T)}) \\ -(2^{-s_i} \pm 2^{-(3s_i+2+T)}) & 1 \pm 2^{-(2s_i+1)} \end{bmatrix}$$

where

$$T = \begin{cases} 0, & \text{for hyperbolic} \\ 1, & \text{for circular.} \end{cases} \tag{8}$$

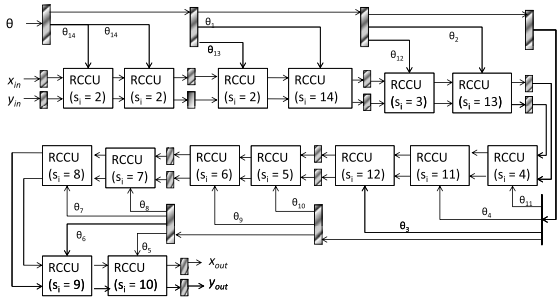

Fig. 4. Reconfigurable rotation-mode CORDIC unit for basic-shift 2.

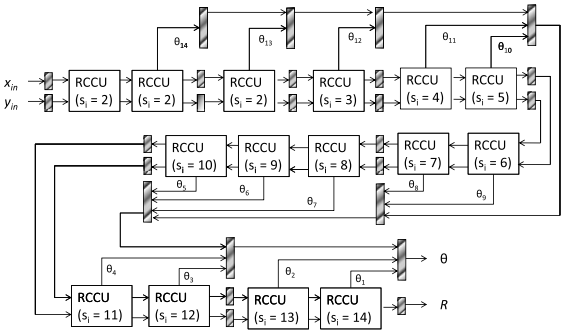

Fig. 5. Proposed pipeline reconfigurable vectoring-mode CORDIC unit for $s_{\text{basic}} = 2$.

By changing the implementation of the RCCU to implement (8), the recursive architecture of Fig. 2 can be used to realize CORDIC iterations for vectoring-mode. The rollover counter value is 15 for $s_{\text{basic}} = 2$, and 17 for $s_{\text{basic}} = 3$.

The pipelined architecture of vectoring-mode reconfigurable CORDIC consists of eight stages for $s_{\text{basic}} = 2$, as shown in Fig. 5. Similar to reconfigurable rotation-mode CORDIC, for increasing shift-indices, the implementation of RCCUs is simplified for reconfigurable vectoring-mode CORDIC as well. The input coordinates $[x'_{\text{in}}, y'_{\text{in}}]$ are first preprocessed to obtain coordinates $[x_{\text{in}}, y_{\text{in}}]$ and octant mapping signals. The coordinates $[x_{\text{in}}, y_{\text{in}}]$ are input to the vectoring-mode CORDIC pipeline to generate an angle $\theta \in [0, \pi/4]$. The rotation angle θ generated by the vectoring-mode CORDIC pipeline is mapped to the desired octant using the octant mapping signals generated by the preprocessing unit. Therefore, the RoC supported by the proposed vectoring-mode reconfigurable CORDIC is $[-\pi, \pi]$.

C. Proposed Generalized Reconfigurable CORDIC

The generalized reconfigurable CORDIC can operate either in vectoring-mode or in rotation-mode for both circular and hyperbolic trajectories. The user can select the trajectory of operation using a single bit signal T ($T = 1$ for circular and $T = 0$ for hyperbolic). Another single bit signal M is used to control the mode of operation ($M = 0$ for rotation-mode and $M = 1$ for vectoring-mode). The recursive architecture of the proposed generalized reconfigurable CORDIC is implemented by combining the CORDIC microrotators for both rotation-mode and vectoring-mode CORDICs, as shown in Fig. 6. The throughput of the proposed recursive generalized reconfigurable CORDIC is the same as that of the recursive reconfigurable vectoring-mode CORDIC.

The block diagram for pipelined generalized reconfigurable CORDIC using basic-shift $s_{\text{basic}} = 2$ is shown in Fig. 7. It can be easily extended to basic-shift $s_{\text{basic}} = 3$ as is done for reconfigurable rotation-mode and vectoring-mode CORDICs.

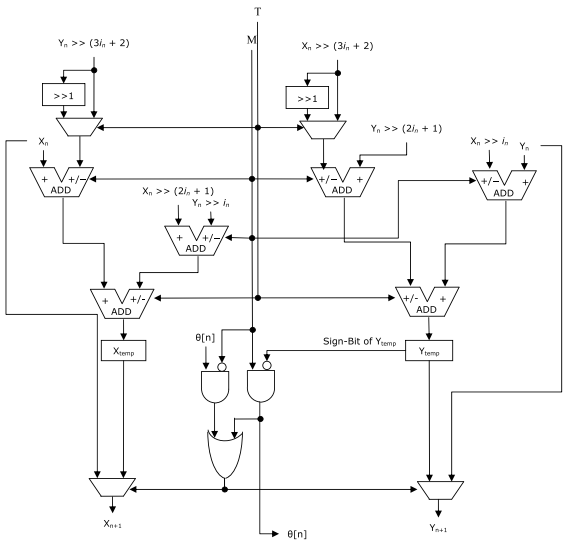

Fig. 6. Structure of CORDIC microrotator for the proposed recursive generalized reconfigurable CORDIC.

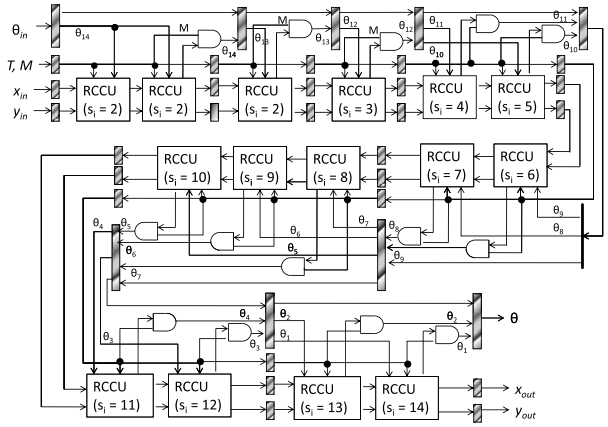

Fig. 7. Proposed pipeline generalized reconfigurable CORDIC unit for $s_{\text{basic}} = 2$.

TABLE I
COMPARISON OF RECONFIGURABLE PIPELINED CORDIC ARCHITECTURE IMPLEMENTED IN VIRTEX 5 XC5VLX50-2FF153

Design	NoSR[1]	LUTs[1]	MOF[1]	NoPS[1]
RRMC [12]	879	1363	398.891	18
RVMC [12]	876	1368	398.891	18
GRC [12]	898	1460	323.761	18
Proposed RRMC ($s_{basic} = 2$)	180	1095	202.065	6
Proposed RRMC ($s_{basic} = 3$)	205	1301	202.065	7
Proposed RVMC ($s_{basic} = 2$)	314	1309	153.149	8
Proposed RVMC ($s_{basic} = 3$)	353	1531	153.149	9
Proposed GRC ($s_{basic} = 2$)	328	2075	144.786	8
Proposed GRC ($s_{basic} = 3$)	369	2161	142.147	9

[1] NoSR = Number of Slice Registers, LUTs = Number of Slice Look-Up-Tables, MOF = Maximum Operating Frequency in MHz. NoPS = No. of pipelining stages

V. COMPLEXITY AND PERFORMANCE CONSIDERATIONS

The reconfigurable CORDIC architectures are coded in Verilog and synthesized using the Xilinx ISE and Synopsis Design Compiler using UMC 180-nm CMOS library, for the FPGA and ASIC implementations, respectively. We compare, here, the area and time complexities of the proposed reconfigurable architectures with the reference design [12]. Tables I and II compare the FPGA and ASIC

TABLE II
COMPARISON OF SYNTHESIS RESULTS OF RECONFIGURABLE PIPELINED CORDIC ARCHITECTURE FOR ASIC IMPLEMENTATION

Design	Area	Minimum Clock Period	Dynamic Power
RRMC [12]	10832.3	0.91	186.88
RVMC [12]	13529.4	0.91	190.35
GRC [12]	13622.5	0.91	300.16
Proposed RRMC ($s_{basic} = 2$)	8202.7	0.79	91.61
Proposed RRMC ($s_{basic} = 3$)	8504.6	0.79	101.25
Proposed RVMC ($s_{basic} = 2$)	7458.8	0.79	147.37
Proposed RVMC ($s_{basic} = 3$)	8391.3	0.79	162.64
Proposed GRC ($s_{basic} = 2$)	10594.8	0.79	253.16
Proposed GRC ($s_{basic} = 3$)	11897.1	0.79	272.82

TABLE III
COMPARISON OF RECONFIGURABLE RECURSIVE CORDIC ARCHITECTURE IMPLEMENTED IN VIRTEX 5 XC5VLX50-2FF153

Design	NoSR	LUTs	MOF	WCI[1]
RRRMC [12]	100	254	268.966	18
RRVMC [12]	98	255	268.966	18
RGRC [12]	115	312	253.27	18
Proposed RRMC ($s_{basic} = 2$)	62	300	196.608	8
Proposed RRMC ($s_{basic} = 3$)	64	265	197.556	11
Proposed RVMC ($s_{basic} = 2$)	66	293	196.905	15
Proposed RVMC ($s_{basic} = 3$)	67	269	185.986	17
Proposed GRC ($s_{basic} = 2$)	66	409	158.787	15
Proposed GRC ($s_{basic} = 3$)	85	387	168.791	17

[1] WCI = No. of worst-case iterations

TABLE IV
COMPARISON OF SYNTHESIS RESULTS OF RECONFIGURABLE RECURSIVE CORDIC ARCHITECTURE FOR ASIC IMPLEMENTATION

Design	Area	Minimum Clock Period	Dynamic Power
RRRMC [12]	3182.7	1.1	47.57
RRVMC [12]	3235.4	1.1	48.51
RGRC [12]	3427.2	1.2	55.23
Proposed RRMC ($s_{basic} = 2$)	1696.1	1.1	10.64
Proposed RRMC ($s_{basic} = 3$)	1627.9	1.1	9.51
Proposed RVMC ($s_{basic} = 2$)	1639.6	0.84	13.68
Proposed RVMC ($s_{basic} = 3$)	1565.2	0.8	13.02
Proposed GRC ($s_{basic} = 2$)	1954.8	0.9	28.94
Proposed GRC ($s_{basic} = 3$)	1924.9	0.9	26.33

implementations of the proposed reconfigurable CORDIC with the reference design of the conventional reconfigurable CORDIC for pipeline architectures, respectively, while the recursive architectures are compared in Tables III and IV for FPGA and ASIC implementations, respectively. The complexity of reference generalized reconfigurable CORDIC [12] requires 1456 1-bit full-adders and 864 1-bit registers, equivalent to 16 560 2-input NAND gates, while the proposed generalized reconfigurable design requires 1180 1-bit full-adders and 384 1-bit registers, equivalent to 12156 2-input NAND gates. For the proposed generalized reconfigurable CORDIC, the RCCU for $s_i = 2, 3, 4$ requires six 16-bit add/sub units and four 2×1 16-bit multiplexers; RCCU for $s_i = 5, 6, 7$ requires four 16-bit add/sub units and two 2×1 16-bit multiplexers; and RCCU for $s_i = 8$–14 requires two 16-bit add/sub units and two 2×1 16-bit multiplexers. Since the complete pipeline is eight stages long, we require eight pipeline registers of 16-bit width on each x, y, and z datapaths. Note that the accuracy of the

proposed reconfigurable CORDIC is the same as that of the scaling-free algorithms of [7] and [10].

VI. Conclusion

In this brief, for the first time a systematic design method for reconfigurable CORDIC is proposed to let a CORDIC function in different modes and different trajectories of operations. The proposed reconfigurable CORDIC architectures can be used in a variety of applications, such as synchronizers, waveform generators, low-cost scientific calculators, and so on. Approximately 60% of the area is saved by the proposed rotation or vectoring-mode reconfigurable CORDIC designs over the reference recursive reconfigurable CORDIC, without any effect on the maximum operating frequency. On the other hand, the proposed pipelined rotation and vectoring-mode reconfigurable CORDIC designs save 30%–50% area compared with the reference reconfigurable design, with nearly the same maximum operating frequency.

References

[1] 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.

[2] 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.

[3] 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.

[4] L. Vachhani, K. Sridharan, and P. K. Meher, "Efficient CORDIC algorithms and architectures for low area and high throughput implementation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 1, pp. 61–65, Jan. 2009.

[5] 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.

[6] 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.

[7] S. Aggarwal, P. K. Meher, and K. Khare, "Area-time efficient scaling-free CORDIC using generalized micro-rotation selection," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 8, pp. 1542–1546, Aug. 2012.

[8] B. Gisuthan and T. Srikanthan, "Flat CORDIC: A unified architecture for high-speed generation of trigonometric and hyperbolic functions," in *Proc. 43rd IEEE Midwest Symp. Circuits Syst.*, Lansing, MI, USA, Aug. 2000, pp. 1414–1417.

[9] D. R. Llamocca-Obregón and C. P. Agurto-Ríos, "A fixed-point implementation of the expanded hyperbolic CORDIC algorithm," *Latin Amer. Appl. Res.*, vol. 37, no. 1, pp. 83–91, 2007.

[10] S. Aggarwal, P. K. Meher, and K. Khare, "Scale-free hyperbolic CORDIC processor and its application to waveform generation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 2, pp. 314–326, Feb. 2013.

[11] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. 38th Spring Joint Comput. Conf.*, Atlantic City, NJ, USA, 1971, pp. 379–385.

[12] S. Aggarwal and P. K. Meher, "Reconfigurable CORDIC architectures for multi-mode and multi-trajectory operations," in *Proc. IEEE ISCAS*, Jun. 2014, pp. 2490–2494.