Coordinate Rotation based Design Methodology for Square root and Division computation

Suresh Mopuri*, Swati Bhardwaj*, Amit Acharyya, *Member, IEEE*

***Abstract*—In this paper, we propose a low-complexity design methodology to compute square-root and division using circular CORDIC. Unlike the state of the art methods, the proposed methodology eliminates the requirement of a separate hardware for square root and division computation in the CORDIC based applications without compromising the computational speed, throughput and accuracy. The ASIC implementation of the proposed architecture has been performed using UMC $90nm$ Technology node with $1.08V$ @$1MHz$ and subsequently Xilinx Virtex-6 (XC6v1x240t) based FPGA-prototyping has been done. The performance of the proposed methodology has been compared with the reported literature and significant power consumption improvement was observed without any additional area overhead.**

***Index Terms*—Square root, Division, CORDIC.**

I. Introduction

CORDIC (COordinate Rotation DIgital Computer) has been used in many signal processing applications including Independent Component Analysis (ICA) and Under determined Blind Source Separation (UBSS) [1]-[8]. These signal processing applications very often require the computation of square root and division. For Example in UBSS, to recover the sources, it is necessary to compute inverse square root of co-variance matrix and inverse of mixing matrix [7]-[8]. The inverse of a matrix can be computed using QR factorization followed by the reciprocal operation [1]-[8]]. The inverse square root of a matrix can be computed using Eigen Value Decomposition (EVD) followed by the square root and reciprocal operation. The QR decomposition and EVD computation can be performed using the circular CORDIC for low-complexity applications [1]-[4].

Several methods are available for the implementation of square root [9]-[15] and reciprocal computation [16]-[24]. From the available literature for square root and reciprocal implementations [9]-[24], it is apparent that the circular CORDIC have not been used till now to perform the square root and division operations. The complex square root and complex division can be performed using circular CORDIC [25]-[26]. Various bio-medical applications based on Electrophysiological signals like Electrocardiography (ECG), Electroencephalography (EEG) and Electromyography (EMG), uses ICA and UBSS for pervasive health monitoring and disease diagnosis. In such applications for remote health monitoring, the monitoring devices need to be portable and battery powered. This gives rise to the necessity for low complex implementations for the underlying algorithms like ICA and UBSS [5]-[8]. Motivated by the aforementioned facts, in this paper, we introduce a low-complex design methodology for the computation of square root ($\sqrt{X}$) and division ($\frac{X}{Z}$) using circular CORDIC. Using the proposed method along with the aforementioned CORDIC based applications [1]-[8], the pre-existing circular CORDIC can be re-used for computation square root and division. The re-utilization of CORDIC reduces the area overhead in the ICA and UBSS implementations for bio-medical applications.

The rest of this paper is organized as follows: Section II provides the necessary theoretical background followed by the proposed methodology and architectural details in Section III. The Hardware complexity and timing are analyzed in Section IV-A and followed by experimental results and error analysis in Section IV-B. Finally Section V concludes the discussion.

II. Theoretical Background

The square root is computed using the derivative Newton Raphson (NR) formula i.e, $y^1 = \frac{1}{\sqrt{X}}$ is computed first and then multiplied by X i.e, $y = y^1 * X = \sqrt{X}$. The reciprocal square root function using NR method is expressed as follows [12]:

$$y^1 = \frac{x_0}{2}(3 - X * x_0^2) \tag{1}$$

where x_0 is the initial approximation. Most often, the division is computed by multiplying the dividend with the reciprocal of divisor. The reciprocal function computed using NR method can be expressed as follows [24]:

$$y = x_1(2 - x_1 X) \tag{2}$$

where x_1 is the initial approximation. The initial approximation x_i ($i = 0$ for the square root and $i = 1$ for the reciprocal function) can be computed using the following second order approximation expression

$$x_i = a_2 X^2 + a_1 X + a_0 \tag{3}$$

To compute the x_i from the above expression the input x is divided into different segments. Coefficients a_2, a_1, a_0 are varied for each input segment, $i = 0$ and $i = 1$. To compute the square root by using (1), (3) and the division by using (2), (3) on the hardware require seven and six multiplications respectively [12], [24]. The basic working principle of circular CORDIC, on the other hand, can be expressed as follows:

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} Rot_x(x_0, y_0, \theta) \\ Rot_y(x_0, y_0, \theta) \end{bmatrix} \tag{4a}$$

$$\theta = Vec_{\theta/x}(x_0, y_0, x_c/y_c) \tag{4b}$$

where x_0, y_0 and x_f, y_f are the initial and final components of the vector and θ is angle of rotation. The output of x/y component of the rotation mode CORDIC denoted as $Rot_{x/y}(.)$. $Vec_{\theta/x}(.)$ denotes the θ/x output of the vectoring mode CORDIC by equating one of the x_f/y_f co-ordinates with either of x_c/y_c respectively. The details are omitted here, [1],[5] can be referred for the same.

* S.Mopuri and S.Bhardwaj contributed equally to this work. S.Mopuri, S.Bhardwaj and A.Acharyya are with Department of Electrical Engineering, Indian Institute of Technology, Hyderabad, India-50205, e-mails:{ee13p0004, ee14resch11018 and amit_ acharyya}@iith.ac.in.

This work is partly supported by the Department of Science and Technology(DST)-Science and Engineering Research Board(SERB) (Grant: ECR/2015/00148). CAD Tools are supported under SMDP-C2S program by Ministry of Electronics and Information Technology(MeitY). A.Acharyya acknowledges Visvevaraya Young Faculty Fellowship, MeitY (Govt. of India).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2018.2878599, IEEE Transactions on Circuits and Systems II: Express Briefs

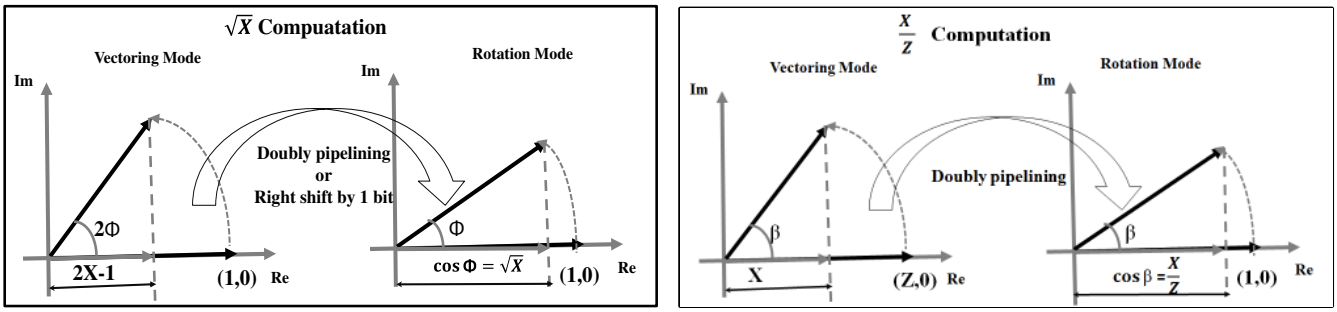

Fig. 1. Geometrical representation of Proposed methodology a) $\sqrt{X}$ computation and b) $\frac{X}{Z}$ computation

III. Proposed Methodology and Architecture

A. Square root computation

Consider a positive real number $X \leq 1$. If $X > 1$, it can be scaled down to less than or equal to 1 by performing simple shifting operation. For example, if $2^{(k-2)} < X \leq 2^k$ where k is an even number, X is scaled down to 1 by right shifting it by k bits. After $\sqrt{X}$ computation, it could be brought to the original value by shifting $\frac{k}{2}$ bits to left. As an example, consider $X = 49$ then $2^4 < X \leq 2^6$ and $k = 6$. After shifting $k = 6$ bits to right X will become 0.765625 and $\sqrt{X} = 0.875$. Thereafter $\sqrt{X}$ should be brought to its original value by shifting $\frac{k}{2} = 3$ bits to the left i.e, $\sqrt{X} = 7$.

Since $X \leq 1$, by assuming $X = cos^2\Phi$, we will get $\Phi = (p-1)\pi + (-1)^{p-1}cos^{-1}\sqrt{X}$ where p is an integer. To limit the rotations to first quadrant, we consider $p = 1$ and thus $cos\Phi = \sqrt{X}$. From the trigonometric identities,

$$cos2\Phi = 2cos^2\Phi - 1 \tag{5}$$

Since $X = cos^2\Phi$, now (5) can be written as:

$$cos2\Phi = 2X - 1. \tag{6}$$

In order to compute the $\sqrt{X}$, it is necessary to compute the angle Φ and corresponding $cos\Phi$ which can be computed by using circular CORDIC. Since X is known, consider $x_0 = 1$ and $y_0 = 0$, operating the circular CORDIC in vectoring mode until $x_c = 2X - 1$ as shown in Fig.1(a), Φ can be computed using (4b)

$$\Phi = \frac{Vec_\theta(1, 0, 2X - 1)}{2} \tag{7}$$

From (7), Φ can be computed by shifting one bit to the right. Since Φ is known, consider $x_0 = 1$ and $y_0 = 0$ to the circular CORDIC once again, operating in rotation mode as shown in Fig.1(a), $cos\Phi$ can be computed using (4a)

$$cos\Phi = \sqrt{X} = Rot_x(1, 0, \Phi) \tag{8}$$

B. Division computation

Consider two real numbers X and Z, assume that $cos\beta = \frac{X}{Z}$. Since $|cos\beta| \leq 1$, then $|X| \leq |Z|$. If $|X| > |Z|$, $|X|$ can be easily scaled down to less than or equal to $|Z|$ by performing simple shifting operation to meet the above assumption. For example, if $2^{(k-1)} \leq (|X| - |Z|) \leq 2^k$ where k is an integer, $|X|$ is scaled down to $|Z|$ shifting by k bits to the right. After $\frac{X}{Z}$ computation, it could be brought to the original value shifting by k bits to the left. As an example, consider $X = 55$ and $Z = 30$ then $|X| - |Z| = 25$, $2^4 < |X| - |Z| \leq 2^5$ and $k = 5$. After shifting by $k = 5$ bits to the right X will become 1.71875 and $\frac{X}{Z} = 0.0573$. There after $\frac{X}{Z}$ should be brought to its original value by shifting $k = 5$ bits to the left i.e, $\frac{X}{Z} = 1.833$.

In order to compute $\frac{X}{Z}$, it is necessary to compute the angle β and corresponding $cos\beta$ which can be computed by using circular CORDIC. Consider the input to the circular CORDIC as a vector $[Z, 0]$ lying on the x-axis i.e, $x_0 = Z$ and $y_0 = 0$ as shown in Fig. 1(b). Since X is known, operating the circular CORDIC in vectoring mode until $x_c = X$ as shown in Fig.1(b), $\theta = \beta$ can be computed using (4b)

$$\beta = Vec_\theta(Z, 0, X) \tag{9}$$

Since β is known from (9), considering $x_0 = 1$ and $y_0 = 0$ to the circular CORDIC, operating in rotation mode as shown in Fig.1(b), $cos\beta$ can be computed using (4a):

$$cos\beta = \frac{X}{Z} = Rot_x(1, 0, \beta) \tag{10}$$

C. Proposed Architecture

Fig.2(a) shows the architecture designed based on the proposed methodology as described in Section III-A and III-B. From (7), (8), (9), (10), Fig.1(a) and Fig.1(b), it can be noted that the angle of rotation is computed from the circular vectoring mode CORDIC (CVCORDIC) which is fed as input to the circular rotation mode CORDIC (CRCORDIC) as shown in Fig.2(b). The Doubly Pipelining (DP) can be used to minimize the latency of the proposed architecture as shown in Fig.2(c) [5]. The CORDIC performs the rotation iteratively through an angle θ_i instead of rotation directly through an angle θ, where $\theta_i = tan^{-1}(2^{-i})$. The θ can be represented in terms of θ_i as given below

$$\theta = \sum_{i=0}^{n-1} \sigma_i\theta_i; \sigma_i = \pm 1 \tag{11}$$

where σ_i is decomposition factor. Consider μ_i as micro-rotation corresponding to σ_i, where $\sigma_i = -1$ corresponds to corresponds to anti-clockwise rotation with $\mu_i = 0$ and $\sigma_i = 1$ corresponds to clockwise rotation with $\mu_i = 1$. It is apparent from (7), (8) and Fig.1(a), that for square root computation, the micro-rotations corresponding to ϕ are not directly available using existing DP CORDIC. Hence, we propose a methodology for computing micro-rotations corresponding to ϕ on-the-fly directly from micro-rotations for 2ϕ.

Consider two angles A and B, with micro-rotations A_μ, B_μ respectively. A_{μ_i}, B_{μ_i} represent the micro-rotation at i^{th} iteration. The micro-rotation AB_{μ_i} corresponds to the angle AB_μ, where $AB_\mu = \frac{A+B}{2}$ can be computed from Fig.3 and Table I. For example, if (A_{μ_i}, B_{μ_i}) = $(0, 1)$ or $(1, 0)$; AB_{μ_i}= 01 or 10, it corresponds to no rotated and the inputs will be directly passed to outputs making $x_{i+1} = x_i$, $y_{i+1} = y_i$. On

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2018.2878599, IEEE Transactions on Circuits and Systems II: Express Briefs

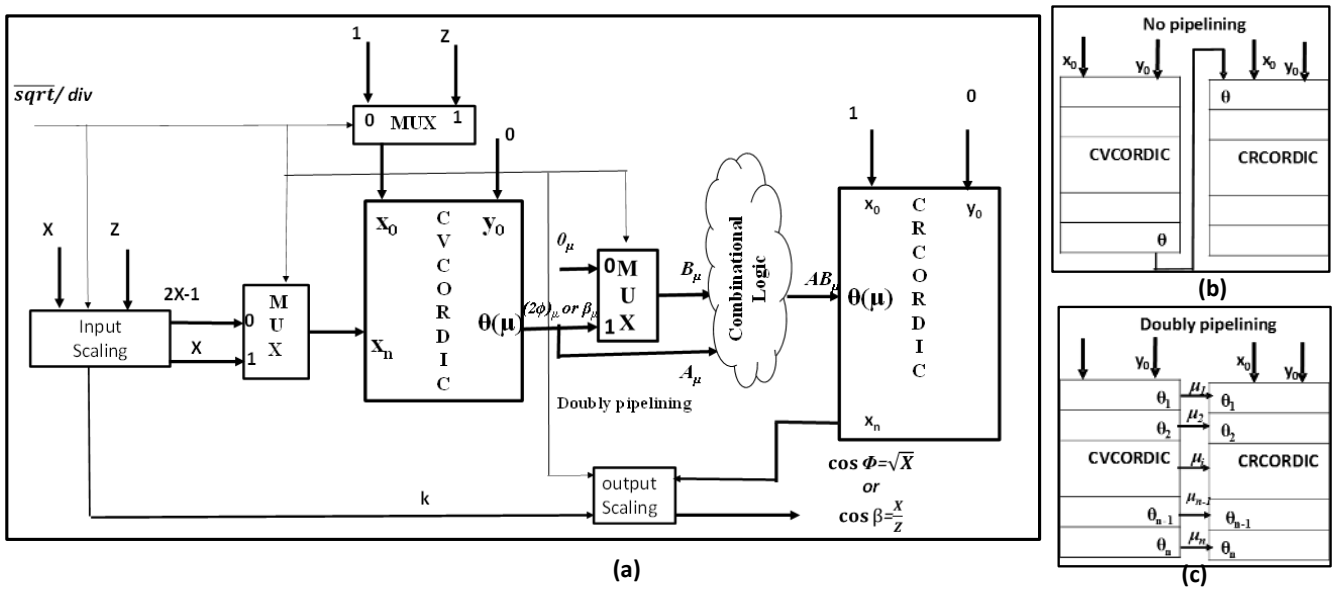

Fig. 2. = (a) Architecture for square root and division (b) With out Pipelining and (c) Doubly Pipelined architecture

the other hand if (A_{μ_i}, B_{μ_i}) = $(0,0)$ or $(1,1)$; AB_{μ_i}= 00 or 11 corresponds to anti-clock wise and clock wise rotation respectively, which is same as the usual CORDIC operation. For example, if $A = 70°$ and $B = 0°$ then for $n = 16$

TABLE I
MICRO-ROTATION TABLE FOR COMPUTATION $\frac{A+B}{2}$

A_{μ_i}	B_{μ_i}	AB_{μ_i}
0	0	00-Anti Clock wise
0	1	01-No rotation
1	0	10-No rotation
1	1	11- Clock wise

stage CORDIC using (11) the $A_\mu = 1101110111111010$ and $B_\mu = 1000101100001011$. The resultant angle $\frac{A+B}{2}$ will become $35°$ and corresponding AB_μ can be computed using Table I and Fig.3. To use DP in the square root computation, the micro-rotations for ϕ i.e, ϕ_μ can be computed from 2ϕ by considering $A = 2\phi$ and $B = 0$.

The architecture performs square root or division computation based on selection line $\overline{sqrt}/div$. When $\overline{sqrt}/div = 0$, the input vector to the CVCORDIC $[x_0, y_0] = [1, 0]$ (7). The CVCORDIC will rotate the input vector until $x_c = (2X - 1)$. The output of CVCORDIC micro-rotation μ corresponds to angle 2ϕ. The micro-rotations ϕ_μ corresponds to angle ϕ are computed by considering $A_\mu = 2\phi_\mu$, $B_\mu = 0_\mu$ = 1000101100001011 using Table I and fed as input to CRCORDIC ($\mu = AB_\mu = \phi_\mu$). The input vector to CRCORDIC is $[x_0, y_0] = [1, 0]$ (8) and the input micro-rotation $\mu = \phi_\mu$. The output of CRCORDIC is $x_n = cos\phi = \sqrt{X}$. When $\overline{sqrt}/div = 1$, the architecture performs division computation and the input vector to the CVCORDIC $[x_0, y_0] = [Z, 0]$ (9). The CVCORDIC will rotate the input vector $x_c = X$. The output of CVCORDIC micro-rotation μ corresponds to angle β which is input to the CRCORDIC. The input vector to CRCORDIC is $[x_0, y_0] = [1, 0]$ (10) and the input micro-rotation $\mu = \beta_\mu$. The outputs of CRCORDIC is $x_n = cos\beta = \frac{X}{Z}$.

IV. Experimental Results and Discussion

A. Hardware Complexity and Timing Analysis

In this subsection, we analyze the performance of the proposed architecture in terms of the hardware complexity,

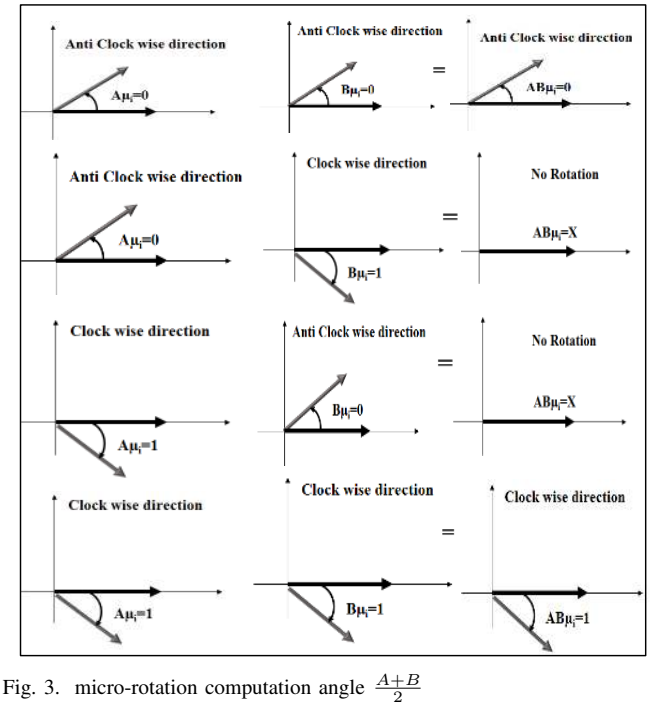

Fig. 3. micro-rotation computation angle $\frac{A+B}{2}$

latency and throughput. Throughout the analysis we keep a generalized view on number of stages in CORDIC as n and word-length as b. To provide comparison on a uniform platform considering only Ripple Carry Adder (RCA) and Conventional Array Multiplier (CAM). A b-bit RCA requires b full adders (FA), bXb CAM requires $b(b-2)$ FA plus b half adders (HA) and b^2 AND gates. In addition, one FA cell requires 24 transistors, one HA cell consist of 12 transistors and a two input AND gates consists of 6 transistors. Based on the approach presented in [5] and [26], Transistor count for the proposed architecture is expressed in terms of Transistor Count (TC) of RCA and CAM. We can calculate TC_{RCA} = $24b$, TC_{CAM} = $6b(5b-6)$. The proposed architecture consists of two CORDICs. Each CORDIC has two additions and two subtractions, the total TC involved in proposed design can be expressed as

$$TC_{proposed} = 8 * n * TC_{RCA} \quad (12)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2018.2878599, IEEE Transactions on Circuits and Systems II: Express Briefs

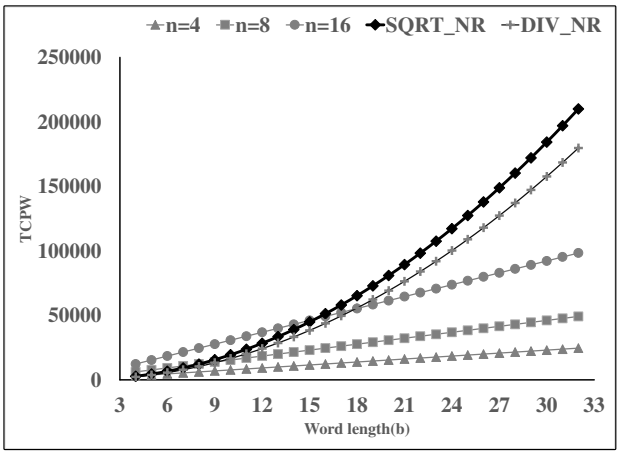

Fig. 4. Transistor count as a function of word-length

The TC involved in the NR method for square root and division implementation can be expressed as

$$TC_{Sqrt} = 7 * TC_{CAM} + 4 * TC_A \tag{13a}$$

$$TC_{Div} = 6 * TC_{CAM} + 3 * TC_A \tag{13b}$$

Fig.4 shows the TCPW (Transistor count per word-length) for the proposed design($n = 4, 8, 16$) and compared with the NR based square root and division implementations. The TSPW (Transistor saving per word-length) is defined as follows

$$TSPW = 1 - \frac{TC_{proposed}}{TC_{Sqrt} + TC_{Div}} \tag{14}$$

As can be seen from Fig.4 that the proposed design saves 49% $TSPW$ for $n = 16$ and $b = 16$ when compared with the NR based approaches. Since the CORDIC consists of n stages, the proposed architecture (with out pipelining as shown in Fig.2(b)) requires $2 * n$ clock-cycles to compute $\sqrt{X}$ and $\frac{X}{Z}$. The latency of the designed architecture is $2n$ clock cycles. But by using DP architecture the latency of the proposed design can be reduced to n clock cycles as shown in Fig.2(c). The throughput of the proposed design is100%.

B. Implementation Results and Error analysis

The proposed architecture is coded in VHDL for 16 bit input word length. The FPGA prototype for this architecture has done on the Xilinx Virtex-6 FPGA (XC6v1x240t).The ASIC implementation was done for the proposed architecture at UMC $90nm$ technology @ VDD=$1.08V$ and clock frequency @ $1MHz$ with the help of Synopsis Design Compiler (DC) and IC compiler. It is to be noted that, the proposed methodology is independent of the technology node used. The use of lower technology node to validate the proposed methodology, will further reduce the area and power consumption for the proposed technology. However, due to the unavailibility of lower technology with us, we could not demonstrate that here. The synthesis results of ASIC implementation and FPGA prototype for the proposed methodology and implementation of other architectures are shown in Table II. However, all the reported literature have been implemented with different specifications using different technology nodes and implementations platform. Thus, it is hard to make comparison on a uniform platform

Accuracy of the designed architecture based on the proposed methodology is determined by comparing outputs from FPGA with Matlab's inbuilt 'sqrt' and 'division' functions. Two set of 2048 randomly generated numbers are considered as input X and Z. The functionality is validated both for square root ($\sqrt{X}$) and division ($\frac{X}{Z}$). Mean Absolute Error (MAE) was calculated with various stages of CORDIC for different input word lengths. Fig.5(a) and Fig.5(b) are shown the variation of MAE for the square root and division with different CORDIC stages $n = 4, 8, 16$, and with word length $b = 4, 8, 16$. From Fig.5(a) and Fig.5(b), it is evident that as number of CORDIC stages increases, MAE decreases due to the improvement in resolution of CORDIC.

V. Conclusion

In this paper, a low-complexity design methodology to compute square-root and division using only circular CORDIC. Subsequently, the ASIC implementation of the proposed architecture architecture has been performed using UMC $90nm$ Technology node with $1.08V$ @$1MHz$ and FPGA-prototyping on Xilinx Virtex-6 (XC6v1x240t). The architecture implementation shows that the proposed architecture design eliminates a separate requirement of square-root and division implementation in the reported CORDIC based applications without compromising the computational speed, throughput and accuracy.

References

[1] P. 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: Regular Papers, vol. 56, no. 9, pp. 18931907, 2009.

[2] Muoz, Sergio D., and Javier Hormigo. "High-throughput FPGA implementation of QR decomposition." IEEE Transactions on Circuits and Systems II: Express Briefs vol. 62, no. 9, pp. 861-865, 2015.

[3] Chang, Robert Chen-Hao, et al. "Iterative QR Decomposition Architecture Using the Modified GramSchmidt Algorithm for MIMO Systems." IEEE Transactions on Circuits and Systems I: Regular Papers vol. 57, no. 4, pp. 1095-1102, 2010.

[4] Zhaohui Liu, K. Dickson and J. V. McCanny, "Application-specific instruction set processor for SoC implementation of modern signal processing algorithms," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 52, no. 4, pp. 755-765, 2005.

[5] A. Acharyya, K. Maharatna, B. M. Al-Hashimi and J. Reeve, "Coordinate Rotation Based Low Complexity N-D FastICA Algorithm and Architecture," in IEEE Transactions on Signal Processing, vol. 59, no. 8, pp. 3997-4011, 2011.

[6] S. Bhardwaj and A. Bhagyaraja and R. Shashank and P. Jadhav and D. Biswas and A. Acharyya and G. R. Naik, "Low Complexity Single Channel ICA Architecture Design Methodology for Pervasive Healthcare Applications", 2016 IEEE International Workshop on Signal Processing Systems (SiPS), 2016, pp. 39-44

[7] S. Mopuri, P. S. Reddy, A. Acharyya and G. R. Naik, "Fast underdetermined BSS architecture design methodology for real time applications," 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Milan, 2015, pp. 5408-5411.

[8] S. Mopuri, P. S. Reddy, K. Ch, A. S. Prasad, A. Acharyya and V. S. Ramakrishna, "Low complexity underdetermined blind source separation system architecture for emerging remote healthcare applications," 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Chicago, IL, 2014, pp. 3833-3836.

[9] Nanhe, Anuja, Gaurav Gawali, Shashank Ahire, and K. Sivasankaran. "Implementation of Fixed and Floating Point Square Root Using Nonrestoring Algorithm on FPGA." International Journal of Computer and Electrical Engineering vol. 5, pp. 533-537, 2013.

[10] Park, I. and Kim, T., Multiplier-Less and Table-Less Linear Approx-376 imation for Square and Square-Root, IEEE International Conference 377 on Computer Design, Lake Tahoe, 4-7 October 2009, 378-383.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2018.2878599, IEEE Transactions on Circuits and Systems II: Express Briefs

TABLE II
PERFORMANCE ILLUSTRATION OF PROPOSED ARCHITECTURE AND OTHER ARCHITECTURES

Design	FPGA Technology	Word Length	Logic Elements	Slice LUTs	Dedicated Registers	Max Clock Frequency (MHz)	ASIC Technology	Core Area ($sq.\mu m$)	Power consumption (mw)
Proposed (SQRT+ DIV)	Viretex-VI	16	-	5592	161	157.1	UMC $90nm$	31623.2	0.51968
SQRT[9]-I	Altera Cyclone II	8	50	-	-	-	-	-	-
SQRT[9]-II	Altera Cyclone II	8	71	-	-	-	-	-	-
SQRT[12]	Altera Cyclone II	32	147	-	88	109.89	-	-	-
SQRT[13]	Xilinx SPARTAN-3E	24	-	173	0	68.22	-	-	90.9
SQRT[14]	Altera Cyclone II	32	580	-	0	-	-	-	-
DIV[21]	-	20	-	-	-	196	TSMC $250nm$	67908	-
DIV[22]	Virtex-E	16	-	6498	-	66.7	AMIS $305nm$	29110	-
DIV[23]-I	-	32	-	-	-	-	UMC $90nm$	31911	19.1
DIV[23]-II	-	32	-	-	-	-	UMC $90nm$	90030	37.1
DIV[24]	Stratix-V	16	339	-	73	68.62	-	-	-

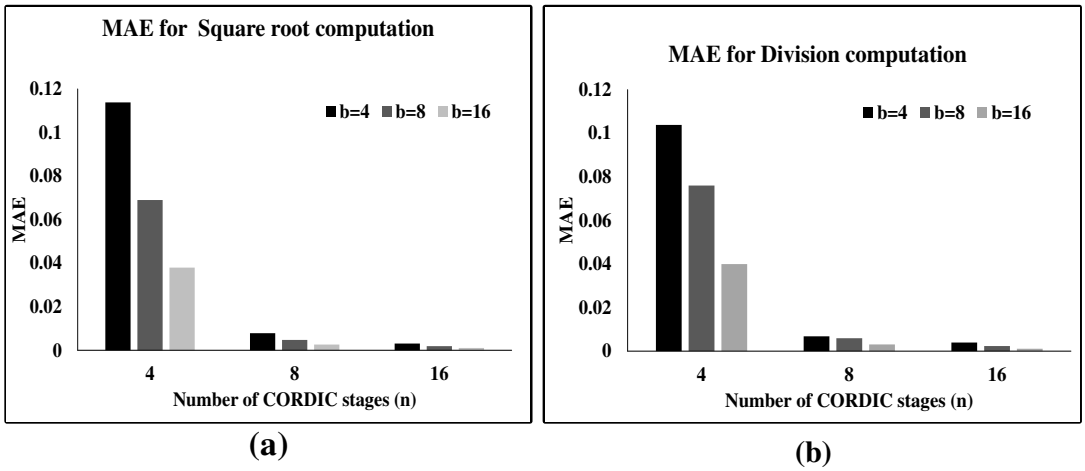

Fig. 5. MAE comparison for the proposed architecture for different input word lengths a) For square root computation, b)For division computation

[11] T. Sutikno, An efficient implementation of the non restoring square 379 root algorithm in gate level, Int. Journal of Computer Theory and 380 Engineering, vol.3, pp.46-51, February 2011.

[12] Putra, Rachmad Vidya Wicaksana. "A novel fixed-point square root algorithm and its digital hardware design." ICT for Smart Society (ICISS), 2013 International Conference on. IEEE, 2013.

[13] Kachhwal, P., & Rout, B. C. (2014, July). Novel square root algorithm and its FPGA implementation. In Signal Propagation and Computer Technology (ICSPCT), 2014 International Conference on (pp. 158-162). IEEE 2014.

[14] Putra, Rachmad Vidya Wicaksana, and Trio Adiono. "A register-free and homogenous architecture for square root algorithm." Computer, Control, Informatics and Its Applications (IC3INA), 2014 International Conference on. IEEE, 2014.

[15] Kwon, T.J. and Draper, J., Floating-Point Division and Square Root 388 Implementation Using a Taylor-Series Expansion Algorithm with Re- 389 duced Look-Up Tables, 51st Midwest Symposium on Circuits and 390 Systems, Knoxville, 10- 13 August 2008, 954-995.

[16] J-A Pineiro, S.F.Oberman, J.M. Muller, and J.D. Bruguera, "HighSpeed Function Approximation Using a Minimax Quadratic Interpolator," IEEE Trans. on Computers, Vol. 54, No. 3, pp. 304-318, Mar. 2005.

[17] F. De Dinechin and A. Tisserand, "Multipartite Table Method," IEEE Transactions on Computers, Vol. 54, No. 3, pp. 319-330, Mar. 2005.

[18] I. Kong and E. E. Swartzlander, A Goldschmidt Division Method With Faster Than Quadratic Convergence, IEEE Transactions on Very Large Scale Integration (VLSI) System, Vol. 19, No. 4, pp. 696-700, 2011.

[19] A. Habegger, A. Stahel, J. Goette and M. Jacomet, "An efficient hardware implementation for a reciprocal unit", Fifth IEEE International Symposium on Electronic Design, Test and Application, 2010, pp. 183-187

[20] D. Chen and S. B. Ko, "Design and Implementation of Decimal Reciprocal Unit," 2007 Canadian Conference on Electrical and Computer Engineering, Vancouver, BC, 2007, pp. 1094-1097.

[21] Kucukkabak, U.; Akkas, A., Design and implementation of reciprocal unit using table look-up and Newton-Raphson iteration, Digital System Design, 2004. DSD 2004. Euromicro Symposium on 31 Aug.-3 Sept. 2004 pp: 249 253

[22] Dongdong Chen, Bintian Zhou, Zhan Guo and P. Nilsson, "Design and implementation of reciprocal unit," 48th Midwest Symposium on Circuits and Systems, 2005., Covington, KY, 2005, pp. 1318-1321 Vol. 2.

[23] S. F. Hsiao, C. S. Wen and M. Y. Tsai, "Low-cost design of reciprocal function units using shared multipliers and adders for polynomial approximation and Newton Raphson interpolation," 2010 International Symposium on Next Generation Electronics, Kaohsiung, 2010, pp. 40-43.

[24] A. Rodriguez-Garcia, L. Pizano-Escalante, R. Parra-Michel, O. Longoria-Gandara and J. Cortez, "Fast fixed-point divider based on Newton-Raphson method and piecewise polynomial approximation," 2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig), Cancun, 2013, pp. 1-6.

[25] D. Wang, M. D. Ercegovac and N. Zheng, "Design of High-Throughput Fixed-Point Complex Reciprocal/Square-Root Unit," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 57, no. 8, pp. 627-631, Aug. 2010.

[26] S. Mopuri and A. Acharyya, "Low-Complexity Methodology for Complex Square-Root Computation," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 11, pp. 3255-3259, Nov. 2017.