An Area Efficient Composed CORDIC Architecture

Francisco AGUIRRE-RAMOS, Alicia MORALES-REYES, Rene CUMPLIDO, Claudia FEREGRINO-URIBE
Instituto Nacional de Astrofisica, Optica y Electronica, 72840, Puebla, Mexico
a.morales@inaoep.mx

***Abstract*—This article presents a composed architecture for the CORDIC algorithm. CORDIC is a widely used technique to calculate basic trigonometric functions using only additions and shifts. This composed architecture combines an initial coarse stage to approximate sine and cosine functions, and a second stage to finely tune those values while CORDIC operates on rotation mode. Both stages contribute to shorten the algorithmic steps required to fully execute the CORDIC algorithm. For comparison purposes, the Xilinx CORDIC logiCORE IP and previously reported research are used. The proposed architecture aims at reducing hardware resources usage as its key objective.**

***Index Terms*—digital systems, computer architecture, field programmable gate arrays, signal processing, circuit optimization.**

I. Introduction

Complex digital systems use trigonometric functions as a fundamental component. These functions are widely used in many areas including digital image and signal processing, cryptography and watermarking. Several approaches have been developed to calculate trigonometric functions, most of them based on polynomial or rational approximations which are not easily mapped into hardware architectures.

CORDIC (Coordinate Rotation Digital Computer) is an iterative algorithm designed to calculate trigonometric functions using basic addition and shift operations, characteristic that makes it suitable for hardware architectures design [1,2]. The algorithm can be configured to operate in vectoring or rotation mode in several coordinate systems, providing the possibility to calculate hyperbolic functions. In rotation mode, a vector is iteratively rotated by an angle, to calculate a final vector corresponding to the sine and cosine functions of an input angle. A new vector is obtained every iteration, after a small rotation (micro-rotation). Each rotation has a specific direction which is calculated every step based on the sign of an angular variable. In vectoring mode, the algorithm follows similar steps but it also allows calculating divisions and logarithmic functions.

At an application level, CORDIC is widely used in the signal processing arena [3-6]. In [3], CORDIC compensation is implemented instead of using multiplications, helping to significantly reduce hardware complexity. In [4], CORDIC algorithm is included in an efficient filtering design where power of two coefficients are calculated. CORDIC algorithm is applied directly in the implementation of a Givens rotation module used in [5], improving computational time and decreasing complexity. In [6] an adaptive CORDIC rotator with constant scaling factor is proposed aiming to reduce resources usage.

Researchers have focused on improving the CORDIC algorithmic core and its implementation. Algorithmic improvements commonly consist in reducing the number of rotations [7-8], modifying the scaling factor [9-10] or changing the determination of rotations direction [8,11,12]. On the other hand, most popular improvements in the hardware arena are related to operational frequency, throughput and occupied area [13].

In this paper, a composed architecture is proposed, it consists in combining a Lookup Table (LUT) with a rotation prediction and a CORDIC pipeline modules. The algorithmic approach takes an input angle and obtains a coarse initial approach of its sine and cosine functions from a LUT. The remaining rotations are algorithmically predicted, and the final result is approximated with the CORDIC pipeline module. Results show a reduction in the number of rotations and in hardware resources usage per iteration.

The paper is organized as follows. In Section III the CORDIC algorithm is presented including specifics for the proposed architectural approach. Section IV introduces the proposed composed hardware architecture design followed by obtained results in Section V. Section VI presents conclusions of this research.

II. Related Work

Related work is extensive and varied, among the approaches aiming to improve CORDIC's algorithmic performance, in [7] Lakshmi et al. proposed a pipelined architecture for radix-4 CORDIC rotations. This VLSI approach refines previously proposed radix-2 techniques in terms of latency and throughput by using redundant arithmetic and higher radix techniques. Originally, radix-4 rotations for a second stage of small rotations were used to accelerate radix-2 CORDIC algorithm [9]. Lee et al. approach was modified to use radix-4 for the entire set of rotations reducing the number of iterations and the hardware resources usage. In [8], a modification of the angle recoding method that avoids an increase in cycle time and allows an arbitrary input angle is presented. This approach calculates in one step all angle constants by comparing the input angle to several adjacent rotation ranges. The lower the number of comparative ranges the lower the number of cycles during the iterative process. In [11], authors follow a similar approach to improve CORDIC by reducing the number of required micro-rotations when large bit-width (64-bit) input

This work was supported partially by the Mexican National Council for Science and Technology (CONACYT) through grant number 261243.

Digital Object Identifier 10.4316/AECE.2014.02019

angles are calculated. The improvement comes from recoding two bits of the input angle concurrently leading to a reduction of 21% in area/delay. In [10] an adaptive approach is proposed which executes necessary iterations with a 50% reduction while maintaining a constant scaling factor. A reduction in hardware resources usage is also reported.

Most popular improvements in the hardware arena are related to operational frequency, throughput and occupied area. In [14] an analysis of standard CORDIC implementations is carried out to reduce the interconnections delay. Several reconfigurable platforms are used for implementing three pre-computing sign methods: Para-CORDIC [12], P-CORDIC [15] and Flat-CORDIC [16]. Results showed P-CORDIC performs better in newer devices and Flat and Para-CORDIC in older FPGAs devices.

III. CORDIC BASIS

CORDIC's algorithmic approach performs vector rotations by arbitrary angles using shifts and additions. The algorithm is based on a general rotation transformation with angles restricted to: $\tan\theta = \pm 2^{-i}$, reducing multiplications to shift operations. Thus, arbitrary angles are obtained while applying a series of micro-rotations. Basic CORDIC equations are shown in (1) and (2):

$$x_{i+1} = k_i\left(x_i - y_i \cdot \sigma_i \cdot 2^{-i}\right) \quad (1)$$

$$y_{i+1} = k_i\left(y_i + x_i \cdot \sigma_i \cdot 2^{-i}\right) \quad (2)$$

where $k_i = 1/\sqrt{1+2^{-2i}}$ and $\sigma_i = \pm 1$; k_i approaches a constant and can be applied at any stage of the process. The necessary micro-rotations to calculate sine and cosine functions are defined by a sequence of directions represented by a decision vector. A set of decision vectors can be stored in a LUT or can be integrated into the system through an extra adder-subtracter that accumulates rotation angles at every iteration. Having an angle accumulator requires the extra equation (3).

$$z_{i+1} = z_i - \sigma_i \cdot tan^{-1}\left(2^{-i}\right) \quad (3)$$

This equation is eliminated in the proposed architecture by pre-calculating rotations directions.

IV. ARCHITECTURE DESIGN

Overall, the proposed hybrid CORDIC architecture consists of three main stages, see Figure 1. During the first stage, sine ($sin(\theta)$) and cosine ($cos(\theta)$) functions are roughly approximated by obtaining an initial estimation from a Lookup Table, the input value for this stage is the input angle (θ). On a second stage, rotations directions are obtained by a module implementing the P-CORDIC algorithm [15]. The pre-calculated rotations are then applied to determine the final trigonometric values using a CORDIC pipeline module.

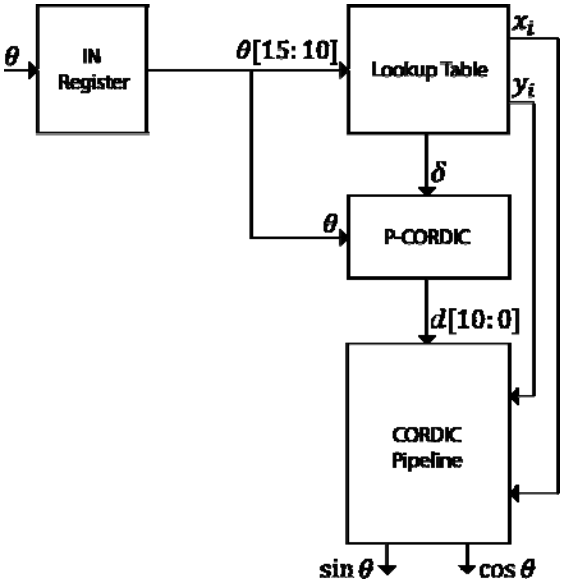

Figure 1. Hybrid CORDIC architecture blocks diagram

A. LUT Module

A LUT is implemented within the hybrid architecture for an initial approximation of the input angle's trigonometric functions. Precision for an unsigned output angle is 19-bit. The LUT stores x_i and y_i values. In the proposed design $i = 5$, thus 51 angles within 0 and $\pi/4$ radians, and their corresponding δ values for rotations prediction are stored. The input angle's 6-msb (most significant bits) are used for memory addressing to retrieve x_i, y_i and δ.

The proposed hybrid architecture shown in Figure 1 requires an input angle within $0 \le \theta \le \pi/4$ with 16-bit precision format, for a resulting output angle with 14-bit precision format for its fractional part.

B. Rotations Prediction

The second stage in the proposed architecture performs the P-CORDIC algorithm, which helps to speed up CORDIC computation by predicting the sequence of directions for all rotations to perform. Rotations directions are calculated in equation (4) by adding the input angle (θ), a constant (stored in the current stage) and an adjustment variable (δ) which is calculated previously and stored in the LUT module.

$$d = \frac{\theta}{2} + 1 - \frac{1}{2}\sum_{i=0}^{\infty}\epsilon_i + sign(\theta)\epsilon_0 + \delta \quad (4)$$

where $\epsilon_i = 2^{-i} - \tan^{-1}(2^{-i})$ and $\delta = \sum_{i=0}^{\infty} d_i\epsilon_i$; d_i-bit calculates its corresponding rotation (σ_i) according to equation (5).

$$\sigma_i = 2d_i - 1 \quad (5)$$

In Figure 2(a) a detailed view of the P-CORDIC module is drawn. Removing z datapath from the CORDIC algorithm in the proposed architecture is achieved by the P-CORDIC algorithm. Since the sequence of rotations directions to perform is known in advance, it is not necessary to verify z_i sign in every iteration, thus eliminating z datapath.

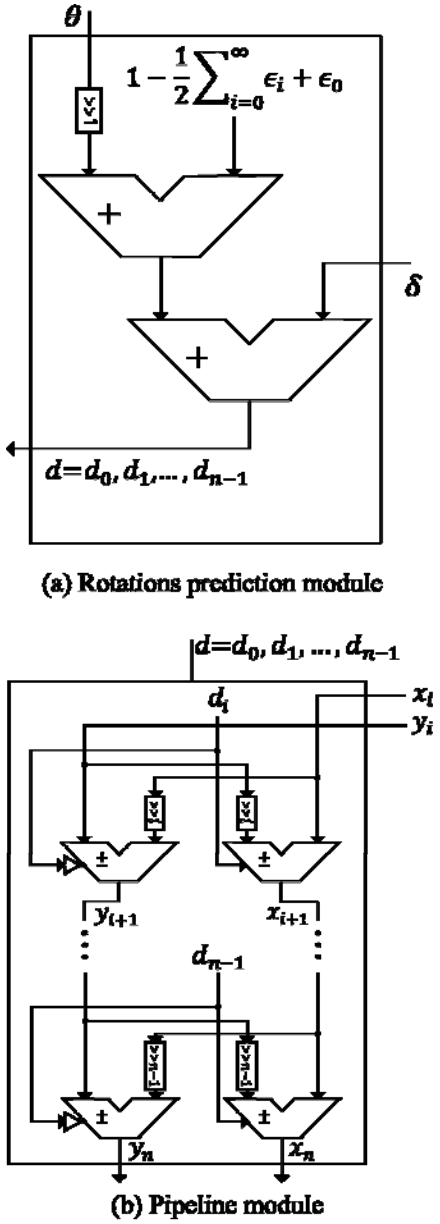

Figure 2. Hybrid CORDIC architecture submodules

C. CORDIC Pipeline

The final stage in the proposed architecture carries out a pipeline for the CORDIC algorithm; details are shown in Figure 2(b). This implementation only uses x and y datapaths, every rotation direction is obtained from the previous module and x_i and y_i values are retrieved from the LUT, where previously computed approximations are stored. In order to reduce the truncation error in every iteration, x and y datapaths are 21-bit wide, enough to achieve 14-bit final precision for trigonometric functions fractional part.

V. RESULTS DISCUSSION

FPGA devices are the chosen implementation platform due to its proven advantages, such as fast prototyping and advanced reconfigurability. The design was coded using VHDL and the tools used to implement the architecture were Mentor's ModelSim, Xilinx System Generator, Matlab/Simulink and Xilinx's ISE 13.2.

Previously proposed approaches and the CORDIC logiCORE IP by Xilinx are used as reference [13], [17-18]. Table 1 shows results in terms of hardware resources and maximum operational frequency. The proposed architecture and the CORDIC logiCORE IP are synthesized for Xilinx Spartan 3 (XC3S50-5) and 6 (XC 6SLX45-2) for direct comparison. However, the approach presented in [13] is synthesized for Spartan 2E not available on Xilinx's ISE 13.2. There is a significant reduction in the used logical resources achieved by the proposed architecture on a Xilinx Spartan 3 of 49% and 26% slices, and 75% and 65% FF for the logiCORE IP [18] and the approach reported in [17] respectively. A reasonable operational frequency, limited by the LUT access latency, is maintained. On the other hand, comparing both performance parameters on a Spartan 6, the logiCORE IP [18] increases in 40% and 70% the number of occupied slices and FF with respect to the proposed approach. In terms of frequency, the logiCORE IP achieves higher frequencies; however, its hardware resources usage is significantly greater.

An example of an application domain suitable for the proposed CORDIC architecture would be a hard real-time problem such as the GPS attitude determination for vehicles navigation [19]. In [20], a high performance hardware architecture is proposed to tackle this problem. The most expensive sub-module has as its core the CORDIC algorithm taking 116 clock cycles to calculate the attitude parameters. Throughput reported is approximately 48.2 μs. The proposed architecture would take 72 clock cycles to calculate the attitude parameters, improving the overall throughput to 30.6 μs, considering the same implementation technology used in [20]. In the next section, conclusions of this research are presented.

TABLE 1. AREA AND SPEED RESULTS

Device	Slices	FF	Max. Freq. (MHz)
Spartan 2E [13]	231	-	58.37
Spartan 3 [17]	373	723	198.27
Spartan 3 logiCORE IP [18]	541	995	180.72
Spartan 3, our approach	**276**	**248**	**83.99**
Spartan 6 logiCORE IP [18]	286	995	284.56
Spartan 6, our approach	**172**	**356**	**110.40**

VI. CONCLUSIONS

An efficient and novel hybrid architecture for the CORDIC algorithm was described. The design strategy is the combination of a LUT to obtain a coarse initial approach of the basic trigonometric functions and the elimination of the z datapath by predicting the sequence of directions for all rotations to perform. It provides several advantages such as a reduction in the number of necessary rotations as well as usage of hardware resources per iteration, while offering significant throughput and a reduction in the occupied area.

Implementation results show that the architecture offers a good balance between high performance and low area complexity. The highly efficient resources usage achieved by the proposed architecture makes it suitable for low

precision systems with limited resources such as mobile devices.

REFERENCES

[1] J. E. Volder, "The CORDIC Trigonometric Computing Technique," IEEE Transactions on Electronic Computers, vol. EC-8, no. 3, pp. 330–334, Sep. 1959. Available: http://dx.doi.org/10.1109/TEC.1959.5222693

[2] J.S. Walther, "A unified algorithm for elementary functions", in Proc. AFIPS Conf., vol. 38, 1971, pp. 385-389. Available: http://dx.doi.org/10.1145/1478786.1478840

[3] C.-C. Sun, P. Donner, and J. Götze, "VLSI implementation of a configurable IP Core for quantized discrete cosine and integer transforms," International Journal of Circuit Theory and Applications, vol. 40, no. 11, pp. 1107–1126, Nov. 2012. Available: http://dx.doi.org/10.1002/cta.774

[4] J.-H. Lee, T.-H. Cheng, and H.-C. Chen, "Design of IIR linear-phase nonuniform-division filter banks with signed powers-of-two coefficients," International Journal of Circuit Theory and Applications, vol. 37, no. 7, pp. 811–834, Sep. 2009. Available: http://dx.doi.org/10.1002/cta.501

[5] M. Abo-Zahhad and M. F. Fahmy, "Synthesis of low-sensitivity orthogonal digital filters," International Journal of Circuit Theory and Applications, vol. 25, no. 6, pp. 503–520, Nov. 1997. Available: http://dx.doi.org/10.1002/(SICI)1097-007X(199711/12)25:6<503::AID-CTA972>3.0.CO;2-#

[6] Maharatna, K., Banerjee, S., Grass, E., Krstic, M., & Troya, A. (2005). Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture. IEEE Transcations on Circuits and Systems Video Technology, 11(11), 1463–1474. Available: http://dx.doi.org/ 10.1109/TCSVT.2005.856908

[7] B. Lakshmi and a. S. Dhar, "VLSI architecture for low latency radix-4 CORDIC," Computers & Electrical Engineering, vol. 37, no. 6, pp. 1032–1042, Nov. 2011. Available: http://dx.doi.org/10.1109/TCSVT.2005.856908.

[8] T. K. Rodrigues and E. E. Swartzlander Jr., "Adaptive CORDIC: Using Parallel Angle Recoding to Accelerate Rotations," IEEE Transactions on Computers, vol. 59, no. 4, pp. 522–531, Apr. 2010. Available: http://doi.ieeecomputersociety.org/10.1109/TC.2009.190.

[9] J.-A. Lee and T. Lang, "Constant-factor redundant CORDIC for angle calculation and rotation," Computers, IEEE Transactions on, vol. 41, no. 8, pp. 1016–1025, 1992. Available: http://dx.doi.org/10.1109/12.156544.

[10] 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, Nov. 2005. Available: http://dx.doi.org/10.1109/TCSVT.2005.856908

[11] T. Juang, "Low Latency Angle Recoding Methods for the Higher Bit-Width Parallel CORDIC Rotator Implementations," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 55, no. 11, pp. 1139–1143, Nov. 2008. Available: http://dx.doi.org/10.1109/TCSII.2008.2002566.

[12] T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai, "Para-CORDIC: Parallel CORDIC Rotation Algorithm," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 8, pp. 1515–1524, Aug. 2004. Available: http://dx.doi.org/10.1109/TCSI.2004.832734.

[13] S. Aggarwal and K. Khare, "Leading One Detection Hyperbolic CORDIC with Enhanced Range of Convergence," Journal of Signal Processing Systems, Feb. 2012. Available: http://dx.doi.org/10.1007/s11265-012-0658-6.

[14] D.-M. Ross, S. Miller, M. Sima, and M. McGuire, "Exploration of sign precomputation-based CORDIC in reconfigurable systems," in 2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), 2011, pp. 2186–2191. Available: http://dx.doi.org/10.1109/ACSSC.2011.6190419.

[15] M. Kuhlmann and K. K. Parhi, "P-CORDIC : A Precomputation Based Rotation," EURASIP Journal on Applied Signal Processing, vol. 2002, no. 1, pp. 936–943, 2002. Available: http://dx.doi.org/10.1155/S1110865702205028.

[16] B. Gisuthan, "Flat CORDIC: a unified architecture for high-speed generation of trigonometric and hyperbolic functions," in Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems, 2000, 2000, pp. 1414–1417. Available: http://dx.doi.org/10.1109/MWSCAS.2000.951478.

[17] M. S. Sinith and K. Jismi, "A comparison of pipelined parallel and iterative CORDIC design on FPGA," in 2010 5th International Conference on Industrial and Information Systems, 2010, no. i, pp. 239–243. Available: http://dx.doi.org/10.1109/ICIINFS.2010.5578702.

[18] Xilinx Inc., "LogiCORE IP CORDIC v4.0. Product Specification, DS249," 2011.

[19] J. Xu, T. Arslan, D. Wan, and Q. Wang, "GPS attitude determination using a genetic algorithm," in Evolutionary Computation, Proceedings of the 2002 Congress on, 2002, vol. 1, pp. 998–1002. Available: http://dx.doi.org/10.1109/CEC.2002.1007061.

[20] E. F. Stefatos and T. Arslan, "High-performance adaptive GPS attitude determination VLSI architecture," in Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on, 2004, pp. 233–238. Available: http://dx.doi.org/10.1109/SIPS.2004.1363055.