REDUCED Z-DATAPATH CORDIC ROTATOR

Koushik Maharatna, Karim El-Shabrawy and Bashir Al-Hashimi
University of Southampton, UK
Email: km3@ecs.soton.ac.uk

ABSTRACT

In this article we propose a novel scheme based on virtually scaling-free COordinate Rotation DIgital Computer (CORDIC) algorithm to design a hardware efficient CORDIC rotator. For predicting rotation directions, less than $1/3^{rd}$ of the elementary rotational stages require classical CORDIC iteration. The rest of the iteration directions could be computed in parallel and the corresponding z-datapath could be eliminated. A 16-bit implementation of the processor requires 0.23 mm^2 silicon area and consumes 967.8 μW power when synthesized in 0.18 μm technology.

Index Terms— CORDIC, low-power, computer arithmetic

1. INTRODUCTION

The forward rotation mode of CORDIC algorithm in circular coordinate (CORDIC rotator) [1] computes sine and cosine of a given angle and carries out complex multiplication using iterative vector rotation in a *to and fro* (two-sided) manner through a set of elementary rotation angles. In terms of hardware, these vector rotations are nothing but a series of shift-and-add operations and the resulting structure is very hardware economical. Thus it has been incorporated in several DSP and communication systems and a large volume of research work has been dedicated to improve its speed and hardware requirement [2 – 6, 9, 10].

Earlier we have proposed a low-power hardware-optimal architecture for CORDIC rotator [5] where we were able to eliminate all the arithmetic operations along the angle computation datapath (or z-datapath) considering unidirectional vector rotation. However, it is not possible to apply the same formulation to reduce hardware for the conventional *two-sided* vector rotation and for a wordlength larger than 18-bit the hardware requirement of the proposed one becomes more than the conventional CORDIC.

In this work, we propose a novel scheme to make our previously proposed architecture more hardware optimal. In essence, the adopted philosophy is very much similar to that proposed in [6] where, the target angle is partitioned into two halves. For the upper half, conventional CORDIC iterations are executed whereas, for the lower half, the CORDIC rotator operation is approximated by linear CORDIC rotator. The directions of rotation for this lower half are predicted in parallel by mapping a logic '1' and a logic '0' to +ve and –ve rotation respectively. In this way, it is possible to eliminate about $2/3^{rd}$ of the required *z*-iterations. In this work we will show that this simple mapping is not sufficient for maintaining accuracy. Consequently we propose a different mapping scheme with no additional hardware. The rest of the paper is structured as follows: in Section 2, we formulate a novel scheme for mapping a logic '1' and '0' in the representation of the target angle to +ve and –ve rotation respectively; Section 3 describes the proposed architecture; and its performance is analysed in Section 4; and conclusions are drawn in Section 5.

2. OPTIMIZATION OF Z-DATAPATH FOR TWO-SIDED ROTATION

In a scaling-free CORDIC [5] the i^{th} elementary rotational angle is described as $\alpha_i = 2^{-i}$. In this formulation, considering the direction of each elementary rotation $\sigma_i \in \{0, 1\}$, there exists a one-to-one correspondence between the position of a logic '1' in the binary representation of the target angle and the elementary CORDIC section undergoing a rotation operation. Thus the bit pattern of the target angle can be used directly as enable signals for different elementary rotational stages and there is no need for any real computation along the z-datapath. However, for two-sided rotation ($\sigma_i \in \{-1, 1\}$) there exists no such direct correspondence and thus the method adopted in [5] cannot be exploited directly to optimize hardware. Although it has been proposed earlier that if $\alpha_i = 2^{-i}$ then a mapping of logic '1' and logic '0' in the expression of the target angle to +ve and –ve rotations respectively will enable the vector to get rotated by the target angle [6], here we will show that this is not sufficient for preserving the accuracy.

Theorem1: If the target angle is represented as $\theta = \sum_{i=1}^{b} s(i)2^{-i}$ where, b is the number of fractional digits, $s(i) \in \{1, 0\}$, and the elementary CORDIC rotational angles are defined as $\alpha_i = 2^{-i}$, then if $s(i) = 1$ represents a +ve rotation and $s(i) = 0$ represents a –ve rotation, then the computed angle is given by $\theta_c = -1 + 2\theta + 2^{-b}$

Proof: Let us consider that $s(i) = 1$ for $i = j, k, l, m$ so that $\theta = 2^{-j} + 2^{-k} + 2^{-l} + 2^{-m}$. Now considering $s(i) = 1$ and 0 is mapped to +ve and –ve rotation respectively, we get

$$\theta_c = -\sum_{i=1}^{j-1} 2^{-i} + 2^{-j} - \sum_{i=j+1}^{k-1} 2^{-i} + 2^{-k} - \sum_{i=k+1}^{l-1} 2^{-i} + 2^{-l} - \sum_{i=l+1}^{m-1} 2^{-i} + 2^{-m} - \sum_{i=m+1}^{b} 2^{-i} \quad (1)$$

Using the rule of geometric progression above equation can be simplified as

$$\theta_c = -1 + 2^{-(j-1)} + 2^{-(k-1)} + 2^{-(l-1)} + 2^{-(m-1)} + 2^{-b} \quad (2)$$

Therefore, in terms of θ, equation (2) can be written as

$$\theta_c = -1 + 2\theta + 2^{-b}$$ □

This implies that in a CORDIC rotator, even though a subset of the elementary rotational angles can be expressed as $\alpha_i = 2^{-i}$, a direct mapping of 1 and 0 to the +ve and –ve rotation respectively will lead to a rotation of the vector through an angle θ_c rather than θ.

Corollary: If an angle is represented as $\theta = \sum_{i=p}^{b} s(i)2^{-i}$ where, b is the number of fractional digits, $s(i) \in \{1, 0\}$, and $\alpha_i = 2^{-i}$, $i \in \{p, p+1, \ldots, b\}$ and $p < j, k, l, m$, then if $s(i) = 1$ represents a +ve rotation and $s(i) = 0$ represents a –ve rotation, the computed angle is given by $\theta_c = -2^{-(p-1)} + 2\theta + 2^{-b}$

Proof: The proof directly follows from Theorem 1.

Theorem2: For a target angle having binary representation $\theta = \sum_{i=p}^{b} s(i)2^{-i}$ and $s(i) \in \{1, 0\}$, $p < j, k, l, m$, $s(i) = 1$ for $i = j, k, l, m$, and $s(i) = 0$ for $i \neq j, k, l, m$ the following relation holds:

$$2^{-p} - \sum_{i=p+1, \neq j,k,l,m}^{b} 2^{-i} + (2^{-j} + 2^{-k} + 2^{-l} + 2^{-m}) = 2\theta + O(2^{-b}) \quad (3)$$

Proof: From the convergence criterion of CORDIC

$$2^{-p} - \sum_{i=p+1}^{b} 2^{-i} = 0 + O(2^{-b}) \quad (4)$$

Equation (4) can be written as

$$2^{-p} - \sum_{i=p+1, \neq j,k,l,m}^{b} 2^{-i} = 2^{-j} + 2^{-k} + 2^{-l} + 2^{-m} + O(2^{-b}) \quad (5)$$

Adding $2^{-j} + 2^{-k} + 2^{-l} + 2^{-m}$ to both sides of equation (5) we get

$$2^{-p} - \sum_{i=p+1, \neq j,k,l,m}^{b} 2^{-i} + (2^{-j} + 2^{-k} + 2^{-l} + 2^{-m}) = 2(2^{-j} + 2^{-k} + 2^{-l} + 2^{-m}) + O(2^{-b}) \quad (6)$$

Now observing $2^{-j} + 2^{-k} + 2^{-l} + 2^{-m} = \theta$, equation (6) can be written as

$$2^{-p} - \sum_{i=p+1, \neq j,k,l,m}^{b} 2^{-i} + (2^{-j} + 2^{-k} + 2^{-l} + 2^{-m}) = 2\theta + O(2^{-b})$$ □

Corollary: For a –ve angle following the conditions of Theorem 2 the following condition holds:

$$-2^{-p} - \sum_{i=p+1, \neq j,k,l,m}^{b} 2^{-i} + (2^{-j} + 2^{-k} + 2^{-l} + 2^{-m}) = -2\theta + O(2^{-b})$$

Proof: The proof directly follows from Theorem 2.

The physical meaning of equation (3) is that, if the p^{th} stage (more generally, the very first stage) of elementary rotation is *always forced* to do a positive rotation and a '0' or '1' in the binary representation of the input angle represents a –ve or +ve rotation of the corresponding iteration stage respectively then effectively the input vector is rotated by an angle 2θ with an error of $O(b)$. Thus, if the binary representation of θ is shifted to its right by one bit position (i.e., yielding $\theta/2$), and the elementary rotational stages (except p^{th} stage) corresponding to the positional bit value '0' and '1' of this shifted bit pattern are treated as undergoing negative and positive rotation respectively, then the vector rotation through an angle θ is achieved.

3. ARCHITECTURE

In the scaling-free CORDIC rotator in [5] the lowest value of i has to be $i_{min} = p = \lfloor (b - 2.585) / 3 \rfloor$. Thus to ensure the convergence over the range of $[0, \pi/8]$ (which was shown sufficient to cover the entire coordinate space [5]), one needs to repeat $i=p$ elementary rotational stage by $N = \lfloor (\pi/8)/2^{-p} \rfloor$ times. As b increases the value of p increases and accordingly N increases rapidly resulting in no more advantage compared to the classical CORDIC anymore. In order to overcome this problem we propose a scheme by integrating $K = (p-2)$ classical elementary rotational stages (with $\alpha_i = \tan^{-1} 2^{-i}$) with $(b-p)$ scaling-free stages. This means that the target angle is partitioned into two halves, one half with the definition of $\alpha_i = \tan^{-1} (2^{-i})$, and the other half with $\alpha_i = 2^{-i}$. It is to be noted that the numerical value of $\pi/8$ is 0.392699. Thus the lowest value of i required to cover the convergence range will be $i = 2$ since $\tan^{-1} (2^{-2}) = 0.255341$ whereas $\tan^{-1} (2^{-1}) = 0.5436$ which is beyond the convergence range. Note that because of our deinition of α_i it is possible to apply the formulation presented in Section 2 from $i = p$ stage. The problem of determination of rotation direction for the classical stages can be tackled easily by considering pipelined operation and noting that since we consider convergence range as $[0, \pi/8]$, all the input angles are +ve and thus $i = 2$ stage will always execute +ve rotation. Thus, determination of direction of rotation for $i = 3$ stage can be carried out concurrently with $i = 2$ rotation operation.

Also note that for $(b-p)$ stages we do not need any real computation along the z-datapath and thus a significant saving of hardware is possible. To illustrate the whole design, here we describe the design of a 16-bit processor as an example. Without any loss of generality, the method described here can be extended for other wordlengths as well.

The complete processor consists of three separate units *viz.*, domain, basic pipeline and output unit.

Domain: This unit is reponsible for carrying out the domain folding operation (described in [5]) by which the vector lying in the range of $\theta \in (\pi/8, \pi/2]$ is mapped to $\phi \in [0, \pi/8]$. It also generates two 2-bit signals namely *domain* and

quad. While the *quad* signal indicates on which quadrant the original target angle lies, the *domain* signal indicates the corresponding domain in the first quadrant on which it has been folded back. The entire operation in this unit is performed in one clock cycle.

Basic pipeline: The schematic diagram of the basic pipeline is shown in Figure 1.

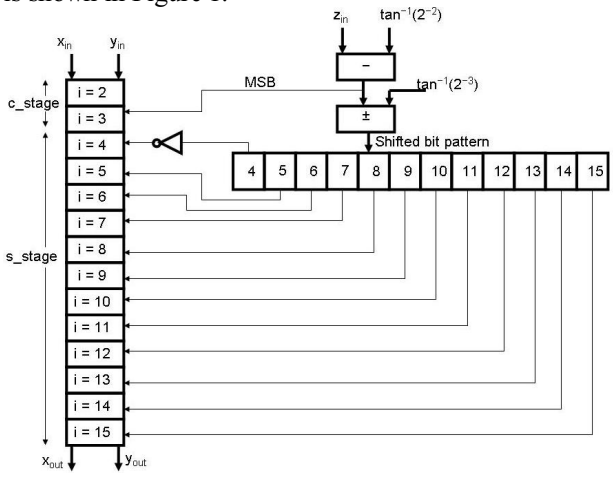

Figure 1. The schematic diagram of the basic pipeline

Since we consider a wordlength of 16-bit, the value of p is 4. Thus the scheme of parallel prediction of direction of rotation developed in Section 2 can be applied for the elementary rotational stages from i = 4 down to 15. This means we need conventional CORDIC iteration for i = 2 and 3 stages. In Figure 1 the conventional stages are denoted as c_stage and the scaling-free stages are denoted as s_stage. The total length of the pipeline is 14 stages. x_{in}, y_{in} and z_{in} are the input data generated after the domain folding operation and x_{out} and y_{out} are the data supplied to the output unit for final processing. Each of the conventional CORDIC section consists of two two-operand adder/subtractors and the corresponding z-datapath requires one adder/subtractor. Note that since i=2 stage always execute +ve rotation, the corresponding z-datapath component is actually a subtractor. After processing through the conventional CORDIC stages, data enter into the scaling-free stages. At this point the residual angle computed in the z-datapath could be either +ve or −ve and accordingly, the 4th bit position will be 0 or 1 respectively. If the residual angle is +ve then from the theory derived in Theorem 2 of Section 2, i=4 stage needs to execute a +ve rotation. Similarly, for −ve residual angle, the direction of rotation will be −ve. Thus, the logic value of 4th bit is inverted before applying to i = 4 stage. Each of the bits from 4th to 15th position of the residual angle acts as the direction of rotation for the corresponding scaling-free elementary rotational stages. Each of the scaling-free stages require four two-operand adder/subtractor. However, as shown in [5], the stages ranging from i = 7 to 15 requires two adder/subtractors each.

Output unit: This circuit is responsible for carrying out the scaling operation. In this particular case the scale factor to be multiplied is 0.9613526. This factor is coupled with the scaling of $\sqrt{2}$ adaptively when the *quad* and *domain* signals indicate that the initial poistion of the vector was in the range ($\pi/8$, $3\pi/8$]. The complete circuit is realised using a shift-and-add technique with a couple of multiplexers for bypassing some of the stages.

4. PERFORMANCE EVALUATION

Hardware requirement: Considering b fractional digits, the x/y datapath requires (b−1) stages. (p−2) and (b−p+1) stages of which are the classical and scaling-free stages respectively. Of the scaling-free rotational stages $(\lfloor(b/2)-1\rfloor)-(p-1)$ stages require four two-operand adder/subtractor whereas; $b-\lceil(b/2)-1\rceil+1$ stages require two two-operand adder/subtractors. Thus the total complexity of the x/y datapath is $[p-1+\ b-\lceil(b/2)-1\rceil\ ]\times2\ +\ [\lfloor(b/2)-1\rfloor-(p-1)]\times4$ two-operand b-bit adder. The hardware complexity of the z-datapath for rotational CORDIC is (p−2) two-operand b-bit adders. As the wordlength increases more classical CORDIC stages are required to be integrated. This number varies from 14% to 28% of the total number of required elementary stages for wordlength from 16-bit to 64-bit.

Error analysis: The error analysis of x/y datapath is done considering 16-bit implementation. We generated 80,000 random values of x, y and θ. The final errors are plotted against magnitude of the vector and θ in Figure 2.

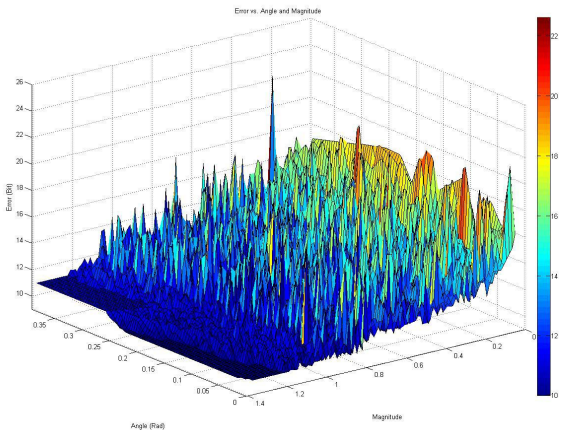

Figure 2(a). Bit position error for x output

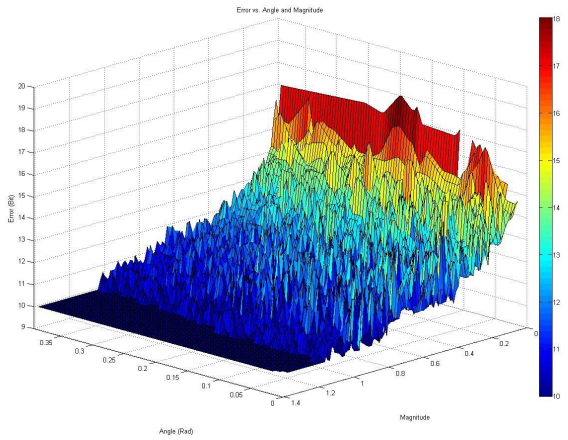

Figure 2(b). Bit position error for *y* output

As the value of *x* and *y* approach 1 (i. e., the vector magnitude approaching √2), the error becomes more significant. Also the error floor is in conformation with the error theory of classical CORDIC developed in [7, 8].

Implementation results: The 16-bit CORDIC rotator is synthesized using 0.18µm CMOS library. The synthesized area of the processor is 0.23 mm^2 of which the domain, basic pipeline and the output unit requires 0.017 mm^2, 0.135 mm^2 and 0.08 mm^2 area respectively.

Power dissipation has been analyzed using Synopsys' Prime power. At 20 MHz the processor consumes 967.8 µW power of which the basic pipeline consume 765 µW, and the domain and output unit consumes about 49.1 µW and 153 µW respectively.

Comparison: We compare the proposed architecture with the architectures reported in [5, 6, 9, 10]. Since different CORDIC architectures use different adder structures, to make a uniform comparison we consider the total number of required *b*-bit adders only (each of these *b*-bit adders could be implemented using different design approaches). The hardware comparison for different wordlengths is shown in Table 1.

Wordlength	[5]	[6]	[9]	[10]	Conventional	Proposed
16	56	38	43	40	48	36
20	88	47		54	60	45
24	244	56	66	68	72	55
28	452	66		134	84	66
32	864	75	95	196	96	77

Table1. Comparison of hardware requirement of the proposed CORDIC rotator with some other reported rotators

In deriving the numbers we have assumed that one byte of ROM cell (applicable for [10]) occupies half the area of a full adder and the area of a redundant adder is approximately equal to three times the area of a full adder. We have also excluded the scaling circuitry since it is common to all except for [5]. As the wordlength increases, the advantage of the proposed architecture becomes apparent compared to the other architectures except for [6]. However, as has been proved in Section 2, the proposed one shows better numerical accuracy compared to that of [6].

5. CONCLUSIONS

In this paper we propose a novel CORDIC rotator algorithm and architecture where scaling-free CORDIC algorithm is used in conjunction with the classical CORDIC algorithm. In this formulation, less than 30% computations are required in the *z*-datapath compared to classical CORDIC which makes it an attractive one from the hardware cost and low-power application point of view. The algorithm proposed here shows similar error characteristic to that of the classical CORDIC albeit at lower hardware cost. Synthesis results for a 16-bit processor following the proposed architectural methodology occupy a very small area and consume very low power.

REFERENCES

[1] J. S. Walther, "A Unified Algorithm for Elementary Functions", Proc. Joint Spring Comput. Conf., vol. 38, pp. 379 – 385, Jul. 1971.

[2] H. Dawid and H. Meyr, "The Differential CORDIC Algorithms: Constant Scale Factor Redundant Implementation without Correcting Iterations", IEEE Trans. Comput., vol. 45, no. 3, pp. 307 – 318, 1996.

[3] A. Troya, K. Maharatna, M. Krstic, E. Grass, U. Jagdhold and R. Kraemer, "Efficient inner-receiver design for OFDM-based WLAN systems: Algorithm and architecture", *To appear in IEEE Trans. Wireless Communication* 2007.

[4] M. Krstic, A. Troya, K. Maharatna and E. Grass, "Optimized Low-power Synchronizer Design for the IEEE 802.11a Standard", *Proc. ICASSP'03*, pp. II 333 – 336, 2003.

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

[6] S. Wang, V. Puiri and E.E. Swartzlander Jr., "Hybrid CORDIC algorithm", *IEEE Trans. Comput.*, vol. 46, no. 11, pp. 1202 – 1207, 1997.

[7] Y. H. Hu, "The quantization effects of the CORDIC algorithm", *IEEE Trans. Signal Processing.*, pp. 834 – 844, April 1992.

[8] K. Kota and J. R. Cavallaro, "Numerical accuracy and hardware tradeoffs for CORDIC arithmetic for special purpose processors", *IEEE Trans. Comput.*, vol. 42, no. 7, pp. 769 – 779, July 1993.

[9] T. B. Jung, S. F. Hsiao and M. Y. Tsai, "Para-CORDIC: parallel CORDIC rotation algorithm", *IEEE Trans. Circuits and Syst.-I*, vol. 51, no. 8, pp. 1515 – 1524, 2004.

[10] M. Kuhlman and K. K. Parhi, "P-CORDIC: A precomputation based rotation CORDIC algorithm", *EURASIP J. Appl. Signal Processing*, vol. 9, pp. 936 – 943, 2002.