LOW POWER CORDIC IP CORE IMPLEMENTATION

Ruiqi Zhang[2], *Jong Hun Han*[1], *Ahmet T. Erdogan*[1, 2] *and Tughrul Arslan*[1, 2]

[1]University of Edinburgh, School of Engineering and Electronics
Edinburgh, EH9 3JL, Scotland, United Kingdom
[2]Institute for System Level Integration, the ALBA Campus
Livingston, EH54 7EG, Scotland, United Kingdom

ABSTRACT

There is a high demand for low power and efficient implementation of complex arithmetic operations in many Digital Signal Processing (DSP) algorithms. The CORDIC algorithm is suitable to be implemented in DSP systems since its calculation for complex arithmetic is simple and elegant. However, the large number of iterations involved in CORDIC operation limits its speed performance seriously and also consumes large power. This paper presents three CORDIC IP cores which were implemented using a new CORDIC algorithm. Each of them has one or more distinctive performances in terms of power, area, speed and flexibility due to their different architectures.

1. INTRODUCTION

The COordinate Rotation DIgital Computer (CORDIC) algorithm was first introduced by Volder [1] for the computation of trigonometric functions, multiplication, division and datatype conversion, and later generalized by Walther [2]. Implementation of the algorithm can be considered as an iterative sequence of additions/subtractions and shift operations. Due to the simplicity of its involved operations, CORDIC algorithm is very well suitable for VLSI implementations and is used in various applications such as digital communication [3], adaptive signal processing [3], computer graphics [4] and robot control [5].

1.1. Review of CORDIC-based rotators

The general vector rotation transform shown in Equation (1) is used to compute trigonometric functions. It rotates a plane vector $[\,x\,,\,y\,]^T$ by angle θ to produce a new vector point with coordinates of $[\,x',\,y'\,]^T$.

$$\begin{aligned} x' &= x\cdot\cos\theta - y\cdot\sin\theta = \cos\theta\cdot(x - y\cdot\tan\theta) \\ y' &= y\cdot\cos\theta + x\cdot\sin\theta = \cos\theta\cdot(y + x\cdot\tan\theta) \end{aligned} \tag{1}$$

The conventional CORDIC algorithm for rotation mode shown in Equation (2) simplifies the operation in Equation (1) by restricting the rotation angles so that $\tan\theta = \mu_i\cdot 2^{-i}$. Equation (2) shows a basic CORDIC iteration [2, 3, 6], which describes a rotation of an intermediate plane vector $v_i = [\,x_i\,,\,y_i\,]^T$ to $v_{i+1} = [\,x_{i+1}\,,\,y_{i+1}\,]^T$.

$$\begin{aligned} x_{i+1} &= K_i\cdot(x_i - \mu_i\cdot y_i\cdot 2^{-i}) \\ y_{i+1} &= K_i\cdot(y_i + \mu_i\cdot x_i\cdot 2^{-i}) \\ z_{i+1} &= z_i - \mu_i\cdot\alpha_i \end{aligned} \tag{2}$$

where, $\alpha_i = \tan^{-1}2^{-i}$ $\mu_i\begin{cases}-1, & \text{if } z_i < 0\\ +1, & \text{if } z_i \ge 0\end{cases}$, with $i \in \{0,\ldots,n-1\}$

K_i is a scaling factor and iteration variable z_i keeps track of the unsigned rotation angle α_i. The well-known radix 2 system is used since it avoids using multiplications while implementing Equation (2). Therefore, a CORDIC iteration can be realized using shifters and adders/subtracters only. However, for a fixed-point implementation with data wordlength of W bits, no more than W CORDIC iterations need to be performed [7]. The large number of iterations limits its speed performance seriously and also consumes large power. Secondly, a scale factor operation is necessary in order to guarantee the final coordinate $[\,x_f\,,\,y_f\,]^T$ has the same norm as the initial coordinate $[\,x_0\,,\,y_0\,]^T$ [6].

The *Angle Recording* (AR) technique [8] is a useful technique in the applications where the rotation angles are known in advance such as lattice-based digital filters [9], FFT or other discrete linear transformations. Compared to the conventional CORDIC algorithm, AR technique can reduce the number of CORDIC iterations and the angle quantization error significantly. However, AR technique has no restriction on the total number of effective CORDIC iterations. Various target rotational angles may generate unequal number of effective iterations which can lead to hardware problems. *Modified Vector Rotational* CORDIC (MVR-CORDIC) algorithm [10] is based on AR technique and has two more modifications proposed. The first alteration is to allow repeat use of the elementary angles because in AR and conventional CORDIC each micro-rotation angle of α_i can be used only once. The second modification is to combat the aforementioned non-fixed effective iteration number in AR technique. *Extended Elementary-Angle Set (EEAS)*-based CORDIC algorithm [11] extends the *Elementary Angle* (α_i) *Set* formed by MVR-CORDIC algorithm and AR technique so that the quantization angle error can be reduced. However, the relaxation on the set of elementary angles is obtained at the expense of double hardware/computational complexity [11]. Although the increased complexity can be compensated by the halved maximum iteration number [11], the overall algorithm complexity is not reduced since the *greedy* search algorithm [8] needs to be applied to search for the closest elementary rotation angles. Besides, the scaling factor operation is still required in MVR-CORDIC and EEAS-CORIDC algorithms.

1.2. The new CORDIC algorithm

In this paper, our CORDIC IP cores were implemented based on a novel CORDIC algorithm presented in [12], but slightly different in the sense of the approximation for sinα as shown in Equation(3) and the avoidance of scaling factor correction. Compared to the conventional CORDIC algorithm, the new algorithm adopted in our designs also reduces the number of CORDIC iterations significantly. It can be derived from Equation (1). The sine and cosine functions can be represented using *Taylor Series Expansion*:

$$\begin{aligned} \sin\alpha &= \alpha - (3!)^{-1}\cdot\alpha^3 + (5!)^{-1}\cdot\alpha^5 + ... \approx \alpha - 2^{-4}\cdot\alpha^3 \\ \cos\alpha &= 1 - (2!)^{-1}\cdot\alpha^2 + (4!)^{-1}\cdot\alpha^4 + ... \approx 1 - 2^{-1}\cdot\alpha^2 \end{aligned} \quad (3)$$

The series up to third order are applied to Equation (1) with the correction of coefficient for sine function, i.e. $(3!)^{-1}$ is approximated to 2^{-4} instead of 2^{-3} to achieve a better performance due to series from fifth order are omitted. The new CORDIC algorithm can be summarized below:

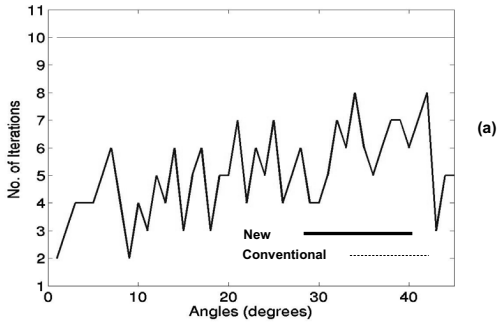
For $\alpha_j = \sin^{-1} 2^{-j} \approx 2^{-j}$, with $j \in \{2,3,...,n-1\}$:

$$\begin{aligned} x_{i+1} &= x_i\cdot\cos(\alpha_j) - y_i\cdot\sin(\alpha_j) \\ &= x_i\cdot(1-2^{-1}\cdot\alpha_j^2) - y_i\cdot(\alpha_j - 2^{-4}\cdot\alpha_j^3) \\ &= x_i\cdot(1-2^{-2j-1}) - y_i\cdot(2^{-j} - 2^{-3j-4}) \\ y_{i+1} &= y_i\cdot\cos(\alpha_j) + x_i\cdot\sin(\alpha_j) \\ &= y_i\cdot(1-2^{-1}\cdot\alpha_j^2) + x_i\cdot(\alpha_j - 2^{-4}\cdot\alpha_j^3) \\ &= y_i\cdot(1-2^{-2j-1}) + x_i\cdot(2^{-j} - 2^{-3j-4}) \\ z_{i+1} &= z_i - \mu_i\cdot\alpha_j \\ &= z_i - \mu_i\cdot 2^{-j} \end{aligned} \quad (4)$$

with $\mu_i \begin{cases} 0, & \text{if } z_i < 0 \\ 1, & \text{if } z_i \geq 0 \end{cases}$, and $i \in \{0,1,...,n-1\}$.

CORDIC iterations can be skipped (not performed) when $\mu_i = 0$, i.e. $x_{i+1} = x_i$, $y_{i+1} = y_i$. Therefore, power consumption can be reduced significantly with the reduced number of CORDIC iterations. Note that j starts with 2 since the quantization error ($\alpha_j = \sin^{-1} 2^{-j} \approx 2^{-j}$) of α_0 and α_1 are large.

We applied both the conventional and the new CORDIC algorithms to angles from 0 to 45° (rather than 0 to 22.5° in [12]) and simulated them in Matlab. For both algorithms, the maximum iteration number was set to ten. Results of the number of iterations to find the end point vector value and the error of the end point vector value at each angle are shown in Figure 1 (a) and (b) respectively.



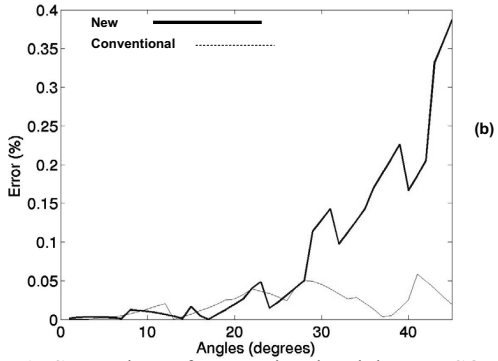


Figure 1. Comparison of conventional and the new CORDIC algorithm for (a) number of iterations (b) error

Clearly, the number of CORDIC iterations is reduced with the new CORDIC algorithm and the error is comparable to that of the conventional CORDIC algorithm. Results of other angles on the coordinate rather than angles from 0 to 45° can be obtained by using 'domain folding' technique [12].

2. IMPLEMENTATION OF OUR CORDIC IP CORES

2.1. Un-folded architecture with pipeline stages

Figure 2 shows an un-folded CORDIC architecture with four pipeline stages. Ten "slot" blocks are used to achieve high calculation precision, i.e. maximum ten CORDIC iterations for one input data. Using pipeline-register stages in the CORDIC IP core reduces the critical path of the design. Besides, increasing the level of pipelining also has the effect of reducing the logic depth and hence the power contributed due to hazards and critical races is reduced as well [13].

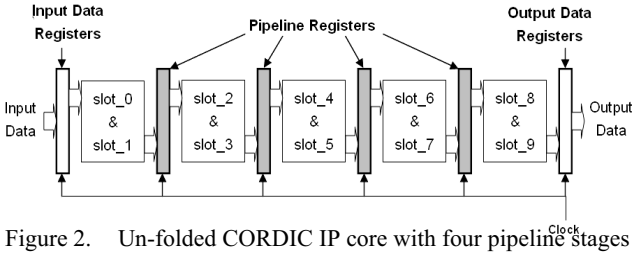


Figure 2. Un-folded CORDIC IP core with four pipeline stages

Each 'slot' block implements one CORDIC iteration based on Equation (4) as shown in Figure 3. Block 'x-rotate' and 'y-rotate' implement x_{i+1} and y_{i+1} in Equation (4) respectively. Block 'angle-compute' implements z_{i+1} and outputs μ_i are used to steer the operation of 'x-rotate' and 'y-rotate' blocks. In the new CORDIC algorithm, the micro-rotation angles are allowed to be used repeatedly. Besides, only the start rotation angle α_0 needs to be either stored in a register or obtained from the input, other angles in the micro-rotation sequence ($\alpha_j = 2^{-j}$) can be obtained easily using a shifter. The conventional CORDIC algorithm usually uses the Table-Look-Up method [3] to obtain each required micro-rotation angle α_i. Compared with the method adopted in the new CORDIC algorithm, the traditional way is more complex and its speed performance is also limited by the memory accessing time.

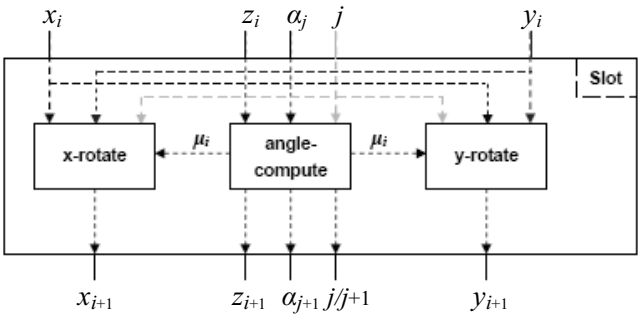


Figure 3. Block diagram of block 'slot'

2.2. Recursive architecture

Physical capacitance can be kept at a minimum by using less logic, smaller devices, fewer and shorter wires [4]. Example techniques for reducing the active area include resource sharing, logic minimization and gate sizing. We have also implemented the recursive/folded architecture with the new CORDIC algorithm. The successive operations are implemented on this architecture as shown in Figure 4. Control signal 'En' is used to control the input/recursive and output data of the recursive CORDIC IP core. The total CORDIC rotation number is set to ten in order to achieve the same function and precision as the un-folded CORDIC IP cores.

The area of the recursive architecture is evidently much less than the un-folded CORDIC IP core and therefore the physical capacitance are minimized. Besides, only one 'slot' block is switching during each clock cycle, therefore the switching activity and delay between the input and output registers of this architecture can also be reduced. This means the clock frequency is increased while the power consumption is reduced. The penalty paid is its low throughput since it takes ten clock cycles to process one input data.

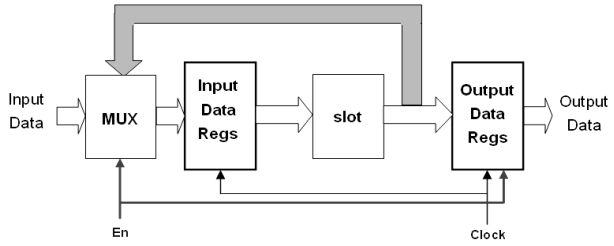


Figure 4. Folded/recursive CORDIC IP core

2.3. Integrated architecture

Beside the un-folded and recursive CORDIC IP cores, we have also implemented another CORDIC IP core which integrates both architectures. It is based on the un-folded architecture with four pipeline stages and uses its input/output registers and one of the 'slot' blocks to realize the recursive architecture as shown in Figure 5. A mode selection input signal has been added to the integrated core. The core operates with un-folded pipelining architecture when it is one; otherwise, recursive architecture is selected. Therefore, this integrated core is more flexible than the pure un-folded architecture with pipeline registers or pure recursive architecture.

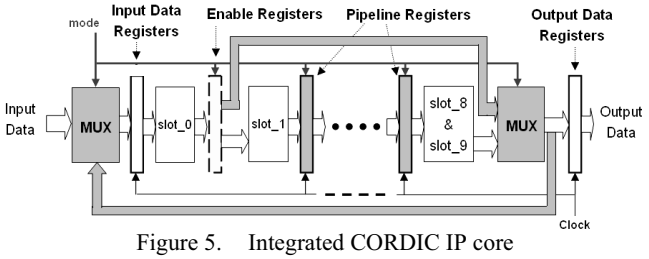


Figure 5. Integrated CORDIC IP core

3. IMPLEMENTATION OF CONVENTIONAL CORDIC IP CORES

In this paper, we have also implemented two CORDIC IP cores using the conventional CORDIC algorithm which was shown in Equation (1) with the same unfolded pipelining and recursive architectures as our CORDIC IP cores respectively (See Figure 2 and Figure 4). The scaling factor K_i is given in Equation (5) [4]:

$$K_i = \cos \alpha_i \quad \text{with} \quad i \in \{0,1,2,..., n-1\} \qquad (5)$$

The 'slot' block of the conventional CORDIC IP cores implements one CORDIC iteration based on Equation (1). Block 'x-rotate' and 'y-rotate' implement x_{i+1} and y_{i+1} in Equation (1) respectively. The only difference from Figure 2 is that two outputs of block 'angle-compute' are $\mu_{i+1} \cdot \alpha_{i+1}$ and i+1 instead of α_{j+1} and j/j+1 respectively. Each micro-rotation angle is allowed to be used only once. The scaling factor K_i and the micro-rotation angle α_i are obtained using Table-Look-Up method [3].

4. SIMULATION RESULTS AND POWER ANALYSIS

The CORDIC IP cores were designed with Verilog HDL and synthesized to UMC 0.18um CMOS standard cell technology library with Synopsys Design Compiler. Synopsys DesignPower was used for power analysis after the gate level simulation. Data wordlength for all cores was set to 32 bits.

4.1. Results of un-folded pipelining CORDIC IP cores

Table I shows the results of unfolded pipelining CORDIC IP cores using both the conventional and the new CORDIC algorithms when operated at their maximum speed. Obviously, due to eliminating the scaling factor correction and hence avoid using multipliers, our core has its area, power reduced by around 60% and 88% respectively and speed increased by around 164% compare to the conventional CORDIC IP core.

Table I. Performances of un-folded pipelining CORDIC IP cores

	Area (mm^2)	Power (mw)	Speed (MHz)	Throughputs
Our	0.881	18.152	111	1 data / 9ns
Conventional	2.184	153.83	42	1 data/ 24ns

4.2. Results of recursive CORDIC IP cores

Results of recursive CORDIC cores using both conventional and the new CORDIC algorithms when operated at their maximum speed are shown in Table II. Our core has its area, power reduced

by around 51% and 86% respectively and speed increased by around 117% compare to the conventional core.

Table II. Performances of recursive CORDIC IP cores

	Area (mm^2)	Power (mw)	Speed (MHz)	Throughputs
Our	0.114	3.259	167	1 data / 60ns
Conventional	0.234	23.23	77	1 data / 130ns

4.3. Comparison between our integrated CORDIC IP core and our other cores

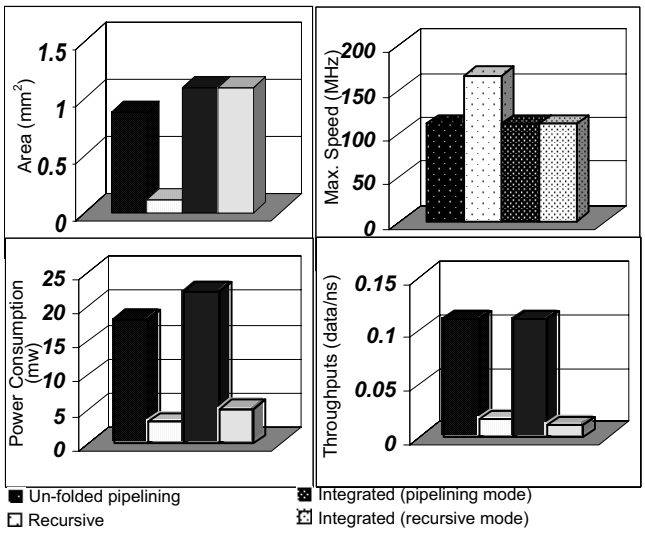


Figure 6. Performances of our different CORDIC IP cores

Compared to the pure un-folded pipelining architecture, the integrated core has the same speed and throughput in pipelining mode but has the power increased by around 22%. Its throughput reduced by 50% and the power consumption increased by around 58% compared to the pure recursive architecture. However, this integrated core provides more flexibility than pure un-folded pipelining architecture or pure recursive architecture. When applications require high throughput, pipeline mode of the integrated core can be selected to achieve the target. Otherwise, the recursive mode can be chosen to realize low power consumption. Moreover, for those applications with power management unit, this integrated CORDIC IP core can be used in pipeline mode for high throughput and switch to the recursive mode when the system does not require to have high throughput. Figure 6 shows the performances of these cores as a summary.

5. CONCLUSION

This paper has presented the implementation of several CORDIC IP cores with different architectures based on a novel CORDIC algorithm [12] which eliminates the operation of scaling factor correction as well as reduces the number of CORDIC iterations significantly. Compared to the conventional CORDIC IP cores, those cores implemented using the new algorithm have significant advantages with both the un-folded pipelining and recursive architectures in terms of power, area and speed.

The un-folded CORDIC IP core with pipeline stages has the highest throughput hence is suitable for high throughput applications where area is not constrained. The recursive core is suitable for low throughput applications which require low power and small area. The integrated core provides the flexibility for choosing either the un-folded pipelining architecture or the recursive architecture.

REFERENCES

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Computers*, vol. EC-8, no. 3, pp.330-34, Sep. 1959.

[2] J. S. Walther, "A unified algorithm for elementary functions," in *AFIPS Spring Joint Computer Conference*, vol. 38, pp. 379-85, 1971.

[3] H. Dawid, H. Meyr, "Chapter 24 CORDIC Algorithms and Architectures," Available from: http://www.google.com/url?sa=U&start=1&q=http://www-cad.eecs.berkeley.edu/~newton/Classes/EE290sp99/lectures/ee290aSp996_1/cordic_chap24.ps&e=9777 [Accessed 8 October 2005]

[4] Tso Bing Juang, Shen Fu Hsiao, "Low power and fast CORDIC processor for vector rotation," *Circuits and Systems, 1999. 42nd Midwest Symposium*, vol. 1, pp. 81 – 83, 8-11 Aug. 1999.

[5] R. G. Harber, X. Hu, J. Li, and S. C. Bass, "The application of bit-serial CORDIC computational units to the design of inverse kinematics processors," in *Proc. 1988 IEEE Int. Conf. Robot. Automat.*, vol. 2, pp. 1152 – 1157, 1988.

[6] Y. H. Hu, "CORDIC-based VLSI Architectures for Digital Signal Processing," *IEEE Signal Processing Magazine*, pp. 16-35, July 1992.

[7] Y. J. He, Z. H. Cai, and A. Y. Wu, "COST-EFFICIENT DIGITAL IP DESIGN OF A HIGH-PERFORMANCE EEAS-CORDIC-BASED VECTOR ROTATOR," MS. Thesis, National Central University, Taiwan, June 2001.

[8] Y. H. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithm implementation," *IEEE Trans. Computers*, vol. 42, pp. 99–102, Jan. 1993.

[9] A. Madisetti, A. Kwentus, and A. J. Willson, "A sine/cosine direct digital frequency synthesizer using an angle rotation algorithm," in *IEEE International Solid-State Circuits Conference, 1995*. Digest of Technical Papers. 41st ISSCC, pp. 262-263, 1995.

[10] C. S. Wu and A. Y. Wu, "Modified vector rotational CORDIC (MVR-CORDIC) algorithm and architecture," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 548–561, June 2001.

[11] A. Y. Wu, C. S. Wu, "A unified view for vector rotational CORDIC algorithms and architecture based on angle quantization approach," *IEEE Trans. Circuits Syst.*, vol. 49, pp. 1442 – 1456, Oct. 2002.

[12] K. Maharatna, A. Troya , S. Banerjee, E. Grass, "Virtually scaling-free adaptive CORDIC rotator," *Computers and Digital Techniques*, IEE Proceedings, Vol. 151, Issue 6, pp. 448 – 456, 18 Nov. 2004.

[13] A. Chandrakasan, S. Sheng, R. W. Brodersen, "Low-Power CMOS Design," *IEEE Journal of Solid-State Circuits*, vol. 27, no.4, pp. 472-484, April 1992.