A Multi-mode Coordinate Rotation Digital Computer (CORDIC)

Lifan Niu, Xiaoling Jia, Jun Wu, and Zhifeng Zhang(✉)

College of Electronics and Information Engineering,
Tongji University, Shanghai 201804, China
{1433332,jia_xiaoling,wujun,zhangzf}@tongji.edu.cn

Abstract. This paper presents a 24-bit fixed-point multi-mode Coordinate Rotation Digital Computer (CORDIC) engine for VLSI implementation of Independent Component Analysis (ICA). Three different modes are integrated for computing sine/cosine, arc tangent and square root to save system resource. We describe the design method for deciding iteration time and fixed-point bits, and present the architecture of a pipelined VLSI implementation. An approximation method is proposed to decrease the data to be pre-stored. The CORDIC engine is designed and implemented with SMIC 65 nm CMOS technology. The performance and computation results of this engine are shown to be very high-accurate and area-efficient.

Keywords: CORDIC · ICA · Sin · Cos · Arctan · Square root

1 Introduction

The Coordinate Rotation Digital Computer (CORDIC) algorithm is an iterative algorithm for computing general vector rotation. It was first brought up by Volder [7], and then it was refined and improved by Walther [8]. CORDIC can compute the trigonometric function, hyperbolic function, logarithm, exponential and square root with only adds and shifts. Therefore, it is suitable for hardware implementation and has applied in many areas including signal processor, communication system and mathematic co-processor.

Independent Component Analysis (ICA) is a widely used algorithm for blind source separation in signal processing. For very large scale integration (VLSI) implementation of ICA, trigonometric and square root functions are necessary [1,4,6], for which CORDIC is a perfect solution. The work in this paper integrates the computation of trigonometric and square root functions in one CORDIC engine, and improves the algorithm specially for hardware implementation. Finally we implement a 24-bit fixed-point multi-mode CORDIC that can compute arc tangent, sine/cosine and square root in one engine with different modes.

The remainder of this paper is organized as follows: Sect. 2 introduces the basic principle of CORDIC. Section 2.1 introduces the improved algorithm

Q. Chen et al. (Eds.): ChinaCom 2016, Part II, LNICST 210, pp. 345–354, 2018.
DOI: 10.1007/978-3-319-66628-0_33

and design methodology. Section 4 introduces hardware architecture. Section 5 presents the timing and area results of the CORDIC engine implementation. Section 6 is the conclusion.

2 Introduction

2.1 Overview

The basic idea of CORDIC is to approach a rotation angle by swinging a series of fixed angles. CORDIC executes a rotation in each iteration. As shown in Fig. 1, to rotate vector (x_i,y_i) by θ to get the new vector (x_j,y_j):

$$\begin{aligned} \mathrm{x}_j &= r\cos(\alpha+\theta) = x_i\cos\theta - y_i\sin\theta \\ y_j &= r\sin(\alpha+\theta) = y_i\cos\theta + x_i\sin\theta \end{aligned} \tag{1}$$

Split θ into N smaller rotation angles, for the n_{th} rotation:

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \cos\theta_n \begin{pmatrix} 1 & -\tan\theta_n \\ \tan\theta_n & 1 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} \tag{2}$$

For clarity, a new variable z_n is used to calculate the residue angle to be rotated. Then the final form of the rotation process is shown in (3) and (4):

$$\begin{pmatrix} x_N \\ y_N \end{pmatrix} = \prod_{n=1}^{N} \cos(m^{1/2}\theta_i) \begin{pmatrix} 1 & -m^{1/2}d_n\tan(m^{1/2}\theta_i) \\ m^{1/2}d_n\tan(m^{1/2}\theta_i) & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{3}$$

$$z_N = z_0 + \sum_{i=1}^{N} d_i\theta_i \tag{4}$$

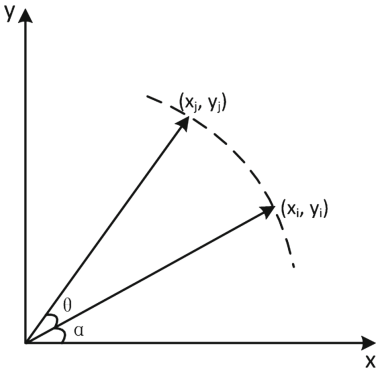

Fig. 1. The coordinate rotation

In rotation mode, z_n is forced to approach zero, while in vector mode y_n is forced to approach zero.

For the convenience of hardware implementation, each rotation angle is chosen to be related with 2^{-n} as (5). With this constraint the complicated computation of arc tangent can be replaced by simple bit-shifting [3].

$$\theta_n = \begin{cases} \arctan(2^{-n}) & m = 1 \\ 2^{-n} & m = 0 \\ \arctan h(2^{-n}) & m = -1 \end{cases} \tag{5}$$

CORDIC executes circular rotation when $m = 1$, hyperbolic rotation when $m = -1$ and linear rotation when $m = 0$.

After times of iteration, $\cos\theta$ in (3) becomes a constant, which is defined as the correction factor K:

$$K = \begin{cases} \prod_{n=0}^{\infty} \sqrt{\frac{1}{1+2^{-2n}}} \approx 0.6072 & m = 1 \\ 1 & m = 0 \\ \prod_{n=1}^{\infty} \sqrt{\frac{1}{1-2^{-2n}}} \approx 1. & m = -1 \end{cases} \tag{6}$$

Correction factor can be multiplied to the final result after the last iteration, so the iteration can be simplified as:

$$\begin{cases} x_{n+1} = x_n - m d_n 2^{-n} y_n \\ y_{n+1} = y_n + d_n 2^{-n} x_n \\ z_{n+1} = z_n - d_n \theta_n \end{cases} \tag{7}$$

Different functions are fulfilled with different choices of m and d. This work includes three different calculating modes: sine/cosine, arc tangent and square root. The details of parameter settings, input and output for these three modes are shown in Table 1.

Table 1. Initialization for different modes

Mode	m	d	Input	Output
Sin/cos	1	$sign(z_n)$	$x_0 = 1/k$, $y_0 = 0$, $z_0 = \theta$	$x_n = cos\theta$ $y_n = sin\theta$
Arctan	1	$-sign(y_n)$	$x_0 = 1$, $y_0 = x$, $z_0 = 0$	$z_n = arctanx$
Square root	-1	$sign(z_n)$	$x_0 = x + 1/4$, $y_0 = x - 1/4$, $z_0 = 0$	$x_n = \sqrt{x}$

3 Improvement of Algorithm and Design Methodology

3.1 Algorithm Improvement

As shown in Eq. (6), the value of θ_n is changing in each iteration. These values need to be pre-stored in implementation, which consume more area. To save

hardware resource, we optimize the algorithm by an approximate method. We know $\arctan(x_n)$ can be expanded with Taylor series:

$$\arctan(x_n) = x_n - \frac{x_n^3}{3} + \frac{x_n^5}{5} - \frac{x_n^7}{7} + \frac{x_n^9}{9} \cdots \tag{8}$$

When x_n equals to 2^{-n}, Eq. (8) can be further simplified:

$$\arctan(2^{-n}) = 2^{-n} - \frac{2^{-3n}}{3} + \frac{2^{-5n}}{5} - \frac{2^{-7n}}{7} + \frac{2^{-9n}}{9} \cdots \tag{9}$$

The value of $\arctan(2^{-n})$ is gradually approaching 2^{-n} as it iterates more times. When the number of iteration time is large enough, we could replace arctan with the first term 2^{-n} only, thus an operation of simply shifting could be used to save area and enhance computing speed as well.

As shown in Table 1, the calculation of square root ($m = -1$) is not required to output z_n, while the other two modes is irrelevant arctanh, thus the calculation of arctanh is not needed, only arctan ($m = 1$) is needed to be implement as above.

3.2 Algorithm Implementation

CORDIC fulfills its function by making a specified parameter approach to zero, z_n is forced to zero for rotation mode, while y_n is forced to zero for vector mode. In VLSI implementation, it is difficult to constantly judge whether the specified parameter has approached to zero during each iteration process, we need to fix the iteration times. An appropriate planning of iteration times could achieve the balance between the consumption of resource and calculation accuracy.

Figure 2 shows the relation between iteration times and absolute error. The result shows that as iteration times increase, the calculation accuracy is higher logarithmically. Square root requires less iteration times than the other two modes. The calculation accuracy of square root reaches around 10^{-6} when iterating 9 times, while the iteration number is required to be 17 for sin/cos and

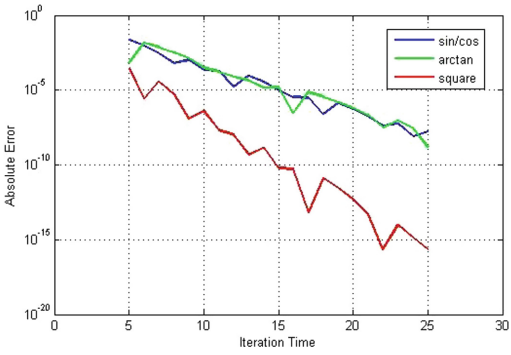

Fig. 2. The relation between iteration time and accuracy

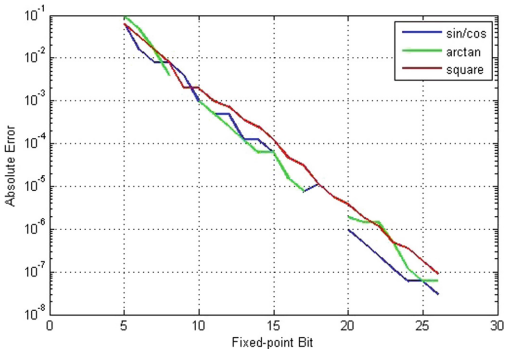

Fig. 3. The relation between fixed-point bit and accuracy

arctan modes. Therefore we choose 17 for sin/cos and arctan modes, and 9 times iteration for square root mode.

The hardware design uses fixed-point instead of floating-point. In the process of converting float-point number to fix-point number, proper bit-width is important. As shown in Fig. 3, the absolute error reaches around 10^{-6} when the fixed-point bit is around 20. Therefore we choose 24 bit fixed points with 20-bit decimal places.

4 Architecture Design

4.1 Pre-processing for Square Root

For arctan mode, the output is in $[-\pi/2, \pi/2]$ [2], and for sin/cos mode, the available interval of input is $[-\pi/2, \pi/2]$ [5], which does not need any initialization. While the simulation results in Fig. 4 show that the input range of square root needs to be $[1.1, 8.1]$, otherwise accuracy gets worse rapidly. So for square root mode, a pre-processing is required before iteration. As shown in Fig. 5, we need to determine whether the input number is in the specified range. If not, shift the input number left or right by $2i$ bit (i for shifting times) until it is in the range. When the calculation is done, shift the output left or right by i bit to eliminate the impact of pre-process. As shown in Fig. 6, when input is out of the range, with pre-process we can still get a result of high accuracy.

As shown in Fig. 7, the system uses a pipelined architecture to implement the core calculation part. According to different input of calculation modes, the rotation direction is decided by a multiplexer, so that addition or subtraction would be executed in each iteration. As mentioned in III-A, value of arc tangent of the rotation angles does not needed to be all saved as a table. As shown in Table 2, only seven values of $\arctan(2^{-n})$ need to be pre-stored. When $n > 6$, value of $\arctan(2^{-n})$ can be easily approximated by 2^{-i}, which is achieved by shifting in hardware implementation.

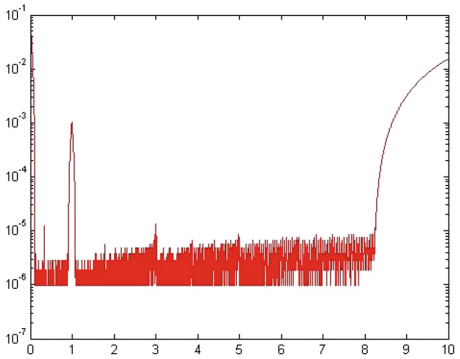

Fig. 4. The available range for the original square root algorithm

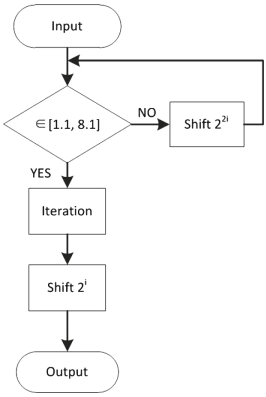

Fig. 5. Flow diagram of initialization for square root mode

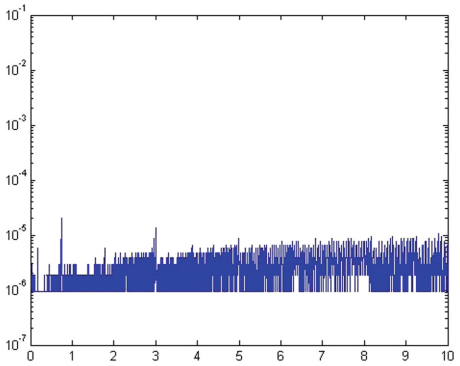

Fig. 6. The available range after the initial operation for square root

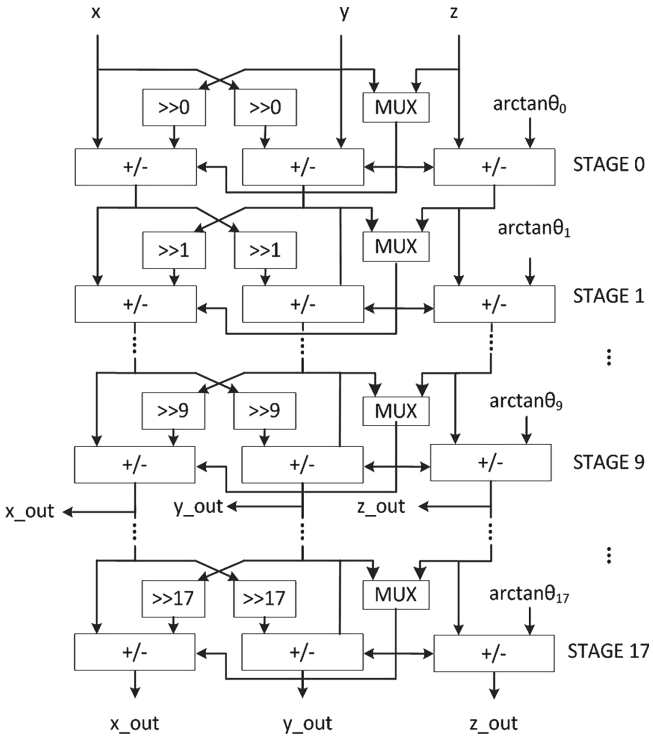

Fig. 7. Structure of the pipelined CORDIC

Table 2. Table of rotation angle

n	Tangent	Value	Hex value
0	arctan	0.78539816	C90FE
1	arctan	0.46364761	76B1A
2	arctan	0.24497866	3EB6F
3	arctan	0.12435499	1FD5C
4	arctan	0.06241881	0FFAB
5	arctan	0.03123983	07FF5
6	arctan	0.01562373	03FFF

5 Results and Dissussion

The CORDIC engine is implemented by Verilog HDL at behavior-level. The simulation is performed in VCS, and its results are shown in Table 3. It can be seen that the CORDIC engine have a high accuracy of about 10^{-6}. For sin/cos and arctan mode, the result is ready after 20 clock cycles as shown in Figs. 8 and 9, respectively. For square root mode, the result is ready after 12 clock cycles when

Table 3. Simulation results

Mode	Input	Exact result	Hex result	Decimal result	Error
Sin	$\frac{1}{4}\pi$	0.707106	0B5054	0.707111	4.6×10^{-6}
	$\frac{3}{8}\pi$	0.923880	0EC836	0.923879	0.1×10^{-6}
Cos	$\frac{1}{4}\pi$	0.707106	0B504a	0.707102	4.2×10^{-6}
	$\frac{3}{8}\pi$	0.382683	061F75	0.382680	3.5×10^{-6}
Arctan	1	0.785398	0C9105	0.785405	7.1×10^{-6}
	2	1.107148	11B6E5	1.107152	4.0×10^{-6}
Square root	6	2.449489	27311F	2.449492	2.7×10^{-6}
	4	2	200001	2.000001	9.5×10^{-7}

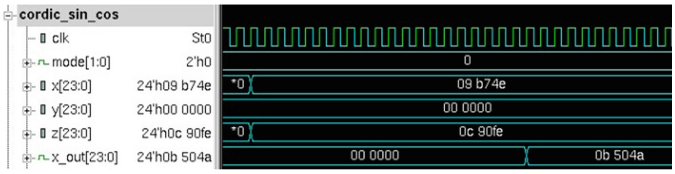

Fig. 8. Simulation of mode sin/cos

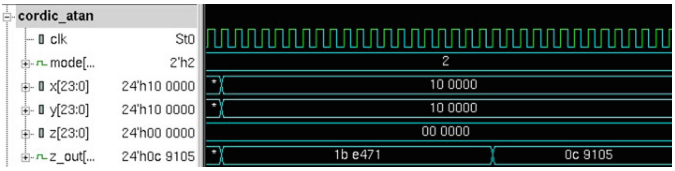

Fig. 9. Simulation of mode arctan

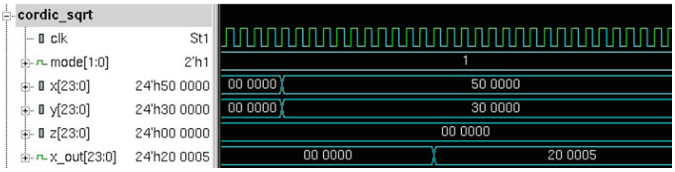

Fig. 10. Simulation of mode square root

input is in the specified range as shown in Fig. 10. Besides the clock cycles for each iteration, there are three more cycles for data input, initialization and choosing output data, respectively.

The design uses 65 nm low power process. Synthesis is carried out by Design Compiler. Placement and route(PR) use IC Compiler. The timing analysis is performed in Primetime. Synthesis and PR is done with the low threshold voltage devices and regular voltage devices under slow corner (1.08v/120°C). The use of

low threshold voltage devices can save power consumption. The analysis under slow corner leaves a margin for actual environment. The Table 4 shows the results of synthesis and PR, namely critical path delay and cell area. The size of this CORDIC engine is 600 μm $\times$ 350 μm. Design uses 1441 sequential cell to achieve all the three calculation modes which is of high source efficiency. The Fig. 11 indicates the clock tree structure of the design. It has a 7-level clock tree to balance the delay.

Table 4. Synthesis and pr results

	Synthesis	Place & route
Critical path delay	1.03 ns	1.24 ns
Combinational cell	37653	41398
Sequential cell	1441	1441
Cell area	117991.7 μm^2	155176.8 μm^2

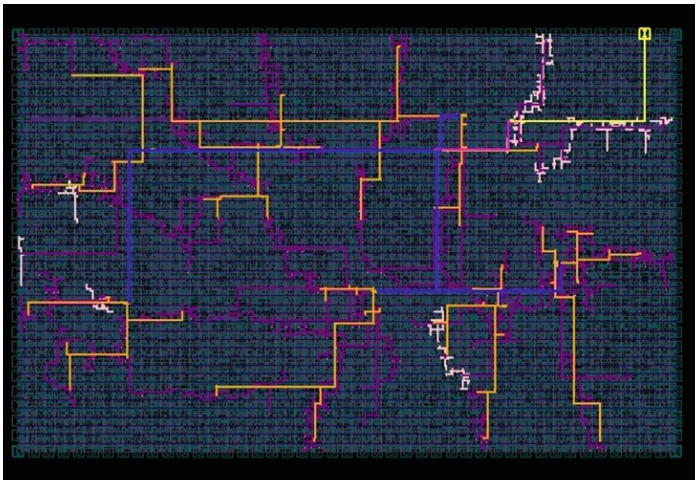

Fig. 11. The clock tree structure of backend place and route

6 Conclusions

In this paper, the design and implementation of an 24-bit efficient multi-mode CORDIC engine are proposed. This CORDIC engine can achieve high-accuracy with appropriate fixed-point design. It is of high-efficiency with fully pipelined

architecture. Instead of using various CORDIC units for different modes, it combines the calculation for sine/cosine, arc tangent and square root together. This multi-mode design makes the proposed CORDIC engine very area-efficient. It is suitable for VLSI implementation of high-precision ICA algorithm, as well as other applications in areas such as high-accurate biomedical signal processing, communication system and mathematic co-processor.

References

1. Cavallaro, J.R., Keleher, M.P., Price, R.H., Thomas, G.S.: VLSI implementation of a CORDIC SVD processor. In: Proceedings of Eighth University/Government/Industry Microelectronics Symposium, pp. 256–260 (1989)
2. Gisuthan, B., Srikanthan, T.: Pipelining flat CORDIC based trigonometric function generators. Proc. SPIE - Int. Soc. Opt. Eng. **33**, 77–89 (2002)
3. Maharatna, K., Banerjee, S., Grass, E., Krstic, M., Troya, A.: Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture. IEEE Trans. Circuits Syst. Video Technol. **5**, 1463–1474 (2005)
4. Ranjith, J., Muniraj, N.: FPGA implementation of optimized independent component analysis processor for biomedical application. In: International Conference on Computer Communication and Informatics, pp. 1–5 (2013)
5. Renardy, A.P., Ahmadi, N., Fadila, A.A., Shidqi, N.: FPGA implementation of CORDIC algorithms for sine and cosine generator. In: International Conference on Electrical Engineering and Informatics (2015)
6. Van, L.D., Wu, D.Y., Chen, C.S.: Energy-efficient FastICA implementation for biomedical signal separation. IEEE Trans. Neural Netw. **22**(11), 1809–1822 (2011)
7. Volder, J.E.: The CORDIC trigonometric computing technique. Electron. Comput. Ire Trans. **EC-8**, 330–334 (1959)
8. Walther, J.S.: A unified algorithm for elementary functions. In: Spring Joint Computer Conference, 18–20 May 1971, pp. 379–385 (1971)