Article

A Unified Reconfigurable CORDIC Processor for Floating-Point Arithmetic

Linlin Fang [1], Bingyi Li [1], Yizhuang Xie [1,*] and He Chen [1]

[1] Beijing Key Laboratory of Embedded Real-time Information Processing Technology, Beijing Institute of Technology, Beijing 100081, China; fanglinlin_1@163.com (L.F.); lby_bit@163.com (B.L.); xyz551_bit@bit.edu.cn (Y.X.); chenhe@bit.edu.cn (H.C.)

* Correspondence: xyz551_bit@bit.edu.cn; Tel.: +86-156-5279-7282

Abstract: This paper presents a unified reconfigurable coordinate rotation digital computer (CORDIC) processor for floating-point arithmetic. It can be configured to operate in multi-mode to achieve a variety of operations and replaces multiple single-mode CORDIC processors. A reconfigurable pipeline-parallel mixed architecture is proposed to adapt different operations, which maximizes the sharing of common hardware circuit and achieves the area-delay-efficiency. Compared with previous unified floating-point CORDIC processors, the consumption of hardware resources is greatly reduced. As a proof of concept, we apply it to 16384×16384 points target Synthetic Aperture Radar (SAR) imaging system, which is implemented on Xilinx XC7VX690T FPGA platform. The maximum relative error of each phase function between hardware and software computation and the corresponding SAR imaging result can meet the accuracy index requirements.

Keywords: reconfigurable architecture; CORDIC; Field Programmable Gate Array(FPGA); SAR imaging

1. Introduction

The CORDIC algorithm involves a simple shift-and-add iterative procedure to perform several computing tasks. It can execute the rotation of a two-dimensional (2-D) vector in linear, circular, and hyperbolic coordinates systems [1]. Due to the simplicity of its hardware implementation, CORDIC has a wide range of applications in signal processing and image processing, such as QR decomposition [2], singular value decomposition [3], 3-D graphics [4] and robotics [5]. The hardware implementation of these applications requires more than one CORDIC processor operating in different modes and different trajectories.

Several unified CORDIC processors have been designed and implemented in previous research. However, previous researches mainly focus on performing fixed-point number arithmetic system based on circular and hyperbolic coordinate. None of the previous designs include linear coordinate system. Literature [6] adopts the traditional pipeline iteration, and just adds the selection signal for mode selection. A reconfigurable CORDIC processor based on scaling-free CORDIC algorithm is demonstrated in [7], which simplifies the iterative computation at the expense of low accuracy. A floating-point CORDIC co-processor is proposed in [8], but the hardware architecture of pre- and post- processing module is too complicated and consumes large resources.

In this paper, we propose a unified reconfigurable floating-point CORDIC processor, which can be operated in three coordinate systems using either rotation-mode or vectoring-mode to complete a variety of operations. The range of convergence (ROC) of algorithm is extended by domain conversion, and the software per-simulation method is used to optimal data width and iterative numbers. According to the dynamically configurable features of FPGA, we optimize the iterative module by designing a pipeline-parallel mixed architecture based on binary-to-bipolar recoding(BBR) [9] techniques, which maximizes the sharing of common hardware circuit in different configurations. The proposed processor achieves high precision, less time latency as well as hardware-complexity of implementation.

To further validate the design, we establish the FPGA-based prototype and apply it to the calculation of three phase functions in chirp-scaling (CS) SAR imaging algorithm [10]. Compared with the simulation results of software, the relative error is less than 10^{-3}. After imaging processing, according to the quality assessment method as [11] described, the result is suitable for both vision and index requirements.

The rest of the article is organized as follows. In section 2, we review CORDIC algorithm. In Section 3, we deduce and simulate the ROC of the algorithm, and analysis the relationship of iterative numbers and data width. In section 4, we propose a unified reconfigurable CORDIC processor. Section 5 discusses the synthesis results of FPGA implementation and precision. Section 6 summarizes the main contributions of this work and concludes the paper.

2. Review of unified CORDIC algorithm

In 1971, Walther reformulated the CORDIC algorithm into a generalized and unified form which is suitable to perform rotations in circular, linear and hyperbolic coordinate systems [1]. It has laid a solid theoretical foundation for designing a unified CORDIC processor architecture. The generalized CORDIC is formulated as follows:

$$\begin{cases} X_{i+1} = X_i - m\delta_i(\mathrm{Y}_i 2^{-i}) \\ Y_{i+1} = Y_i + \delta_i(\mathrm{X}_i 2^{-i}) \\ Z_{i+1} = Z_i - \delta_i\theta_i \end{cases} , i = 0,1,2,\ldots,N-1, \tag{1}$$

For m=1,0 or -1, and $\theta_i = \tan^{-1}\left(2^{-i}\right)$, 2^{-i} or $\tanh^{-1}\left(2^{-i}\right)$, the algorithm given by (1) works in circular, linear or hyperbolic coordinate (CC, LC or HC) system, respectively. $\delta_i \in \{-1,1\}$ is the direction of micro-rotation. In the rotation mode (RM), coordinate Z is driven to zero, $\delta_i = sign\left(Z_i\right)$,and in the vectoring mode (VM), Y is driven to zero, $\delta_i = -sign\left(Y_i\right)$.The value of scaling factor is

$$K_m = \prod_{i=0}^{N-1}\left(1+m2^{-2i}\right)^{-1/2}, \tag{2}$$

The proposed reconfigurable CORDIC processor in this paper is aimed at the calculation of phase functions in chirp-scaling algorithm. Table I shows these elementary functions and operations which can be implemented by CORDIC. The pre- and post-processing step are necessary to perform the operation.

Table 1. Computations using CORDIC algorithm in different configurations

Operation	Configuration	Initialization	Output
Cosine/Sine	CC-RM	$X_0 = K_1 \quad Y_0 = 0 \quad Z_0 = \theta$	$X_n = \cos\theta \quad Y_n = \sin\theta$
Arctangent	CC-VM	$Z_0 = 0$	$Z_n = \tan^{-1}\left(Y_0/X_0\right)$
Multiplication	LC-RM	$Y_0 = 0$	$Y_n = X_0 \square Z_0$
Division	LC-VM	$Z_0 = 0$	$Z_n = Y_0/X_0$
Square-root	HC-VR	$X_0 = a+1 \quad Y_0 = a-1/$ $X_0 = a+1/4 \quad Y_0 = a-1/4$	$X_n = \left(1/K_{-1}\right)\sqrt{a}$

3. Software Pre-analysis of ROC and Iterative Numbers

3.1. Analysis of ROC

The ROC is an important aspect in the design of the CORDIC processor and determines the scope of the algorithm. We analyze the ROC based on theoretical deduction and verify it based on MATLAB simulation. The ROC of five types of floating-point operation is shown in Figure 1.

- Multiplication

According to the iterative formula of *Z-path*, after N iterations, the formula can be derived as follows:

$$Z_{N+1} = Z_1 - \left(\delta_1 2^{-1} + \delta_2 2^{-2} + \delta_3 2^{-3} + \cdots + \delta_N 2^{-N}\right), \tag{3}$$

Since $2^{-1} + 2^{-2} + 2^{-3} + \cdots + 2^{-N} = 1 - 2^{-N} < 1$, initial value need to satisfy $|Z_1| < 1$ to ensure that Z_{N+1} converges to 0 after N iterations.

- Division

According to iterative formula of *Y-path*, after N iterations, the formula can be derived as follows:

$$Y_{N+1} = Y_1 + X_1\left(\delta_1 2^{-1} + \delta_2 2^{-2} + \delta_3 2^{-3} + \cdots + \delta_N 2^{-N}\right), \tag{4}$$

Hence, initial value need to satisfy $|Y_1/X_1| < 1$ to ensure that Y_{N+1} converges to 0 after N iterations.

- Trigonometric functions

According to [12], the ROC of sine and cosine functions is $|Z| \le 99.827^{\circ}$,as well as $\left|\tan^{-1}(Y/X)\right| \le 99.827^{\circ}$ for arctangent function, they can't cover the entire cycle. Therefore, it is necessary to expand the range with some mathematical relationships.

- Square-root

In this paper, we discuss two kinds of transformation solutions for the input coordinates of the initial vector:

1st solution: $X_0 = X + 1, \mathrm{Y}_0 = X - 1$

2nd solution: $X_0 = X + 1/4, \mathrm{Y}_0 = X - 1/4$

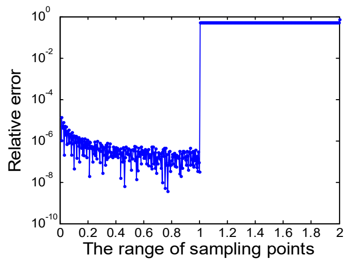
Where, X represents the operand.

Based on software simulation, the relative error graphs of five types operation are shown in Figure 1. The relative error $\mathrm{E}_{relative}$ can be represented as:

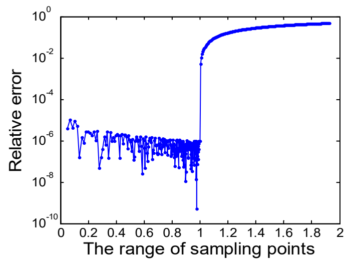
$$\mathrm{E}_{relative} = \frac{R_{results} - R_{arithmetic}}{R_{arithmetic}}, \tag{5}$$

Where, $R_{results}$ is software(MATLAB) simulation results of CORDIC algorithm, $R_{arithmetic}$ is the reference results provided by the MATLAB built-in arithmetic function in single float point precision.

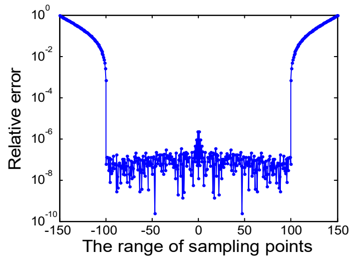
For square-root, the ROC of 1st and 2nd solution is (0.12,9.36) and (0.03,2.34) respectively. For fixed-point arithmetic, 1st solution is only operated in the integer part of the data, therefore, it will produce less carry and make the operation simpler. Besides, compared to 2nd solution, the ROC is more advisable. Finally, in hardware implementation stage, the convergence domain can be expanded by changing the decimal position of fixed-point numbers.



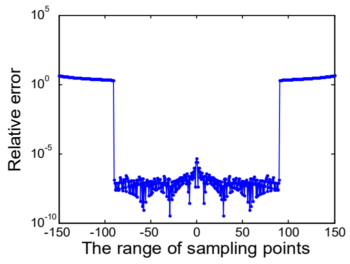
(**a**) Multiplication



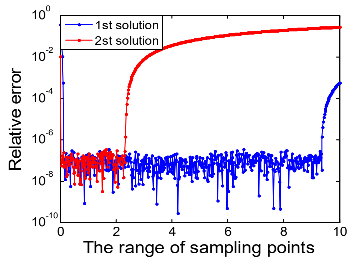
(**b**) Division



(**c**) Sine/Cosine



(**d**) Arctangent



(**e**) square-root

Figure 1. The ROC of different floating-point operation

3.2. Analysis of optimum iterative numbers and data width

There is a need for trade-off between hardware-cost, latency and numerical accuracy subject to the application. The accuracy of the CORDIC algorithm is related to the data bit width *b* and iterative numbers N. The total quantization error consists of two parts: approximate error and rounding error.

According to [13], the approximate error E_A and rounding error E_R can be described as:

$$E_A = 2^{-N+1}\left|v(0)\right|, \tag{6}$$

$$E_R = 2^{-b-0.5}\left[\frac{1+\sum_{j=1}^{N-1}\prod_{i=j}^{N-1}k(i)}{\prod_{i=0}^{N-1}k(i)}+1\right], \tag{7}$$

Where, $k(i)=\prod_{i=0}^{N-1}\sqrt{1+m2^{-2i}}$. Thus, the total quantification error E_{total} is:

$$E_{total} = E_A + E_R, \tag{8}$$

Based on the above theoretical derivation, we introduce the simulation-based method to confirm optimal number of iterations and data width, as shown in Figure 2. In order to satisfy $E_{total} \le 10^{-6}$, the number of iteration is set to 24, the data width *b* is set to 23, the word-length of the fixed-point number in rotation unit is set to 25 bits, which includes *1-bit* sign, *1-bit* integer and *23-bit* decimal.

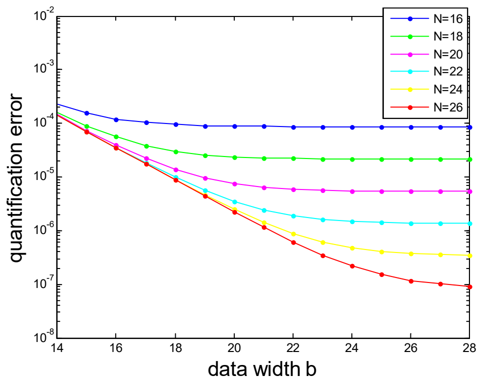


Figure 2. Relationship between iterative number N and data width b

4. Proposed reconfigurable CORDIC processor

The proposed reconfigurable floating-point CORDIC processor design is shown in Figure 3. By adding selectors, common circuits in different modes can be maximized for reuse. We set *2-bit* signal T1&T0 valued as 00,01 or 10 to represent circular, linear or hyperbolic coordinate system respectively, and *1-bit* signal P equals to 0 or 1 to represent rotation or vector mode respectively.

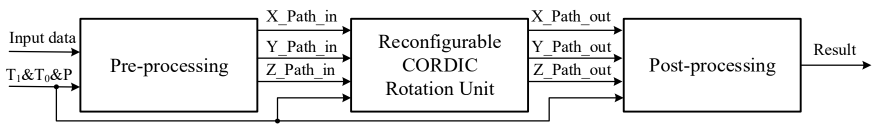


Figure 3. Proposed reconfigurable CORDIC processor

4.1. Pre-processing module

This module mainly completes the conversion from floating-point to fixed-point numbers and expands the ROC. In this paper, we adopt the IEEE-754 standard single precision floating-point data format [14]. The input data can be represented as:

$$X=(-1)^{S_X}\cdot 2^{E_X-Bias}\cdot(1.\mathrm{M}_X)\qquad Y=(-1)^{S_Y}\cdot 2^{E_Y-Bias}\cdot(1.\mathrm{M}_Y)\qquad Z=(-1)^{S_Z}\cdot 2^{E_Z-Bias}\cdot(1.\mathrm{M}_Z)$$

Where, $Bias=127$, the data consists of *1-bit* for sign(*S*), *8-bit* for exponent(*E*), and rest of *23-bit* for mantissa(*M*) or fractional part. Figure 4 shows the hardware structure of pre-processing module. Depending on the operation selected by user, appropriate data path is picked up by signal *T1&T0&P*. The outputs of the pre-processing module, i.e. *X0*, *Y0*, and *Z0*, in different data paths are shown in Table 2.

Table 2. The selection of different data-path

T_1&T_0&P	Operation	X_0	Y_0	Z_0	Exp
000	Sine/Cosine	M_X	M_Y	N_Z	/
001	Arctangent	N_X	N_Y	M_Z	/
010	Multiplication	M_X	M_Y	M_Z	E_X+E_Z-127
011	Division	M_X	$M_Y/2$	M_Z	E_Y-E_X+127
101	Square-root	M_X+1	M_Y-1	M_Z	$E_X/2$ or $E_Y/2$

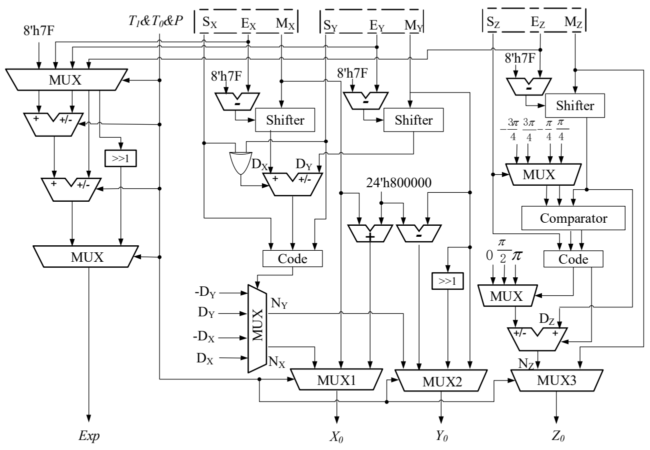


Figure 4. Hardware structure of pre-processing module

- Trigonometric functions operation

The mantissa is shifted according to the result of *E* minus *127*, and it is converted into fixed-point form of *1-bit* sign, *1-bit* integer and *23-bit* decimal part. In this module, the fixed-point numbers are expressed as D_X, D_Y and D_Z. Then, we use the method of mathematical transformation to map the data of entire circumference to the range of which can be covered by CORDIC algorithm. We divide the entire circumference into five intervals and encode it. Through the mapping relations in Table 3, the input phase or vector in interval B, C, D or E can be transformed into interval A. N_X, N_Y, N_Z are the output values after operation.

Table 3. Mapping relations of different quadrants

Domain	Range of the target angle	$S_2\&S_1\&S_0$	Sine/Cosine(Z)	Arctangent(X,Y)
A	$[-\pi/4,\pi/4]$	000	$Z=\theta$	$X'=X,Y'=Y$
B	$(\pi/4,3\pi/4]$	001	$Z=\theta-\pi/2$	$X'=-Y,Y'=X$
C	$(3\pi/4,\pi]$	010	$Z=\theta-\pi$	$X'=-X,Y'=-Y$
D	$(-\pi,-3\pi/4]$	011	$Z=\theta+\pi$	$X'=-X,Y'=-Y$
E	$(-3\pi/4,-\pi/4)$	100	$Z=\theta+\pi/2$	$X'=Y,Y'=-X$

- Multiplication and division operation

The mantissa is expressed as fixed-point number which includes *1-bit* sign, *1-bit* integer and *23-bit* decimal part as the input of CORDIC rotation unit. For multiplication, $E_Y=E_X+E_Z-127$. For division, $E_Z=E_Y-E_X+127$, in order to ensure $|Y/X|<1$, we add a right shifter in *Y* data path.

- Square-root operation

Determining the parity of exponent firstly, the exponent is divided by 2 directly, according to the exponent, the mantissa part is expressed as the corresponding fixed-point number format. The initial value of *X*- and *Y-path* is M_X plus 1, M_Y minus 1 respectively, where, M_X equal to M_Y.

4.2. *Design of reconfigurable CORDIC rotation module*

4.2.1. Rotation unit A

Rotation unit A is used to implement pipeline iterative structure, the single layer iteration structure is shown in Figure 5, it requires two *1-bit* shifters, three multiplexers and three *25-bit* add/sub units. Rotation direction *selx*, *sely* and *selz* are defined by MUX3. Table 4 shows the selection scheme of different multiplexers.

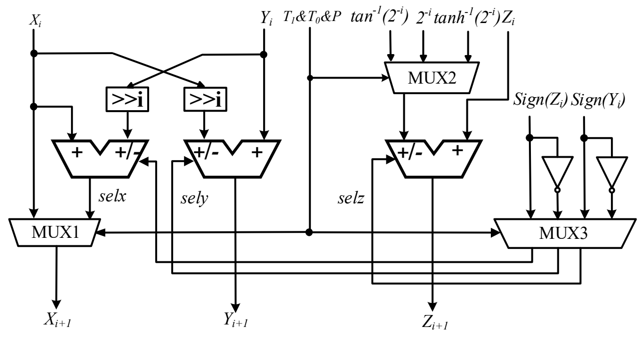


Figure 5. Architecture of reconfigurable rotation unit A (single-layer)

Table 4. The selection of data path

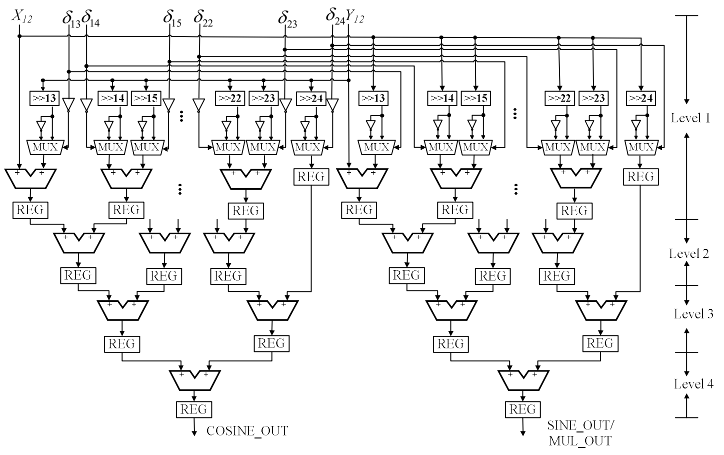
T₁&T₀&P	**MUX1**	**MUX2**		**MUX3**	
	X-Path	θ_i	*selx*	*sely*	*selz*
000	$x_{i+1}=x_i+selx\left(y_i2^{-i}\right)$	$\theta_i=\tan^{-1}\left(2^{-i}\right)$	$\sim sign(Z_i)$	$sign(Z_i)$	$\sim sign(Z_i)$
001	$x_{i+1}=x_i+selx\left(y_i2^{-i}\right)$	$\theta_i=\tan^{-1}\left(2^{-i}\right)$	$sign(Y_i)$	$\sim sign(Y_i)$	$sign(Y_i)$
010	$x_{i+1}=x_i$	$\theta_i=2^{-i}$	/	$sign(Z_i)$	$\sim sign(Z_i)$
011	$x_{i+1}=x_i$	$\theta_i=2^{-i}$	/	$\sim sign(Y_i)$	$sign(Y_i)$
101	$x_{i+1}=x_i+selx\left(y_i2^{-i}\right)$	$\theta_i=\tanh^{-1}\left(2^{-i}\right)$	$\sim sign(Y_i)$	$\sim sign(Y_i)$	$sign(Y_i)$

4.2.2. Rotation unit B

Rotation unit B is used to implement parallel structure. Based on binary-to-bipolar recoding (BBR) technique, in rotation mode, rotation directions δ_i can be predicted by the binary value of the initial input angle in parallel. Parallel processing method unfold the micro-rotation directly. Thus, the rotation of *X*- and *Y-path* can be executed concurrently. The iterations from the (N/2+1)th to the (N+1)th can be simplified as follows:

$$\begin{cases} X_{N+1}=X_{N/2+1}-Y_{N/2+1}\sum_{i=N/2+1}^{N}\delta_i 2^{-i} \\ Y_{N+1}=Y_{N/2+1}+X_{N/2+1}\sum_{i=N/2+1}^{N}\delta_i 2^{-i} \end{cases}, \tag{9}$$

On the basis of formula (9), we merge 13-24 micro-rotation stages. Because too much logic operation will lead to the decline of the overall clock frequency, we adopt a tree structure accumulation circuit in the design. Compared to traditional CORDIC structure, it saves 8 clock cycles (from 12 to 4) in the condition of low routing stress of hardware. The architecture of reconfigurable rotation unit B is shown in Figure 6.



X_{12} δ_{13} δ_{14} δ_{15} δ_{22} δ_{23} δ_{24} Y_{12}
>>13 >>14 >>15 >>22 >>23 >>24
MUX
REG
Level 1
Level 2
Level 3
Level 4
COSINE_OUT
SINE_OUT/ MUL_OUT

Figure 6. Architecture of reconfigurable rotation unit B

4.3. Post-processing module

This module implements the quadrant recovery of trigonometric functions and fixed-point to float-point conversion. We set *1-bit* signal $D = T_1 \wedge T_0$, $D = 0$ represents trigonometric operation and $D = 1$ represents other three operations. The recovery of quadrant is accomplished according to code signal *S₂&S₁&S₀* acquired in pre-processing module. As shown in Table 5, proper data are picked up in three paths. Then, according to leading '1' location of fixed-point data except sign bit, the exponent part is normalized and mantissa part is determined, finally, they are spliced together as the outputs, the fixed-point number is converted into floating-point number. The final output results are represented as *X_result*, *Y_result*, *Z_result*, respectively. The hardware structure is shown in Figure 7.

Table 5. Data selection of different path

$D\&S_2\&S_1\&S_0$	*X-Path*	*Y-Path*	*Z-Path*
0000	X	Y	Z
0001	-Y	X	$Z+\pi/2$
0010	-X	-Y	$Z-\pi$
0011	-X	-Y	$Z+\pi$
0100	Y	-X	$Z-\pi/2$
1XXX	X	Y	Z

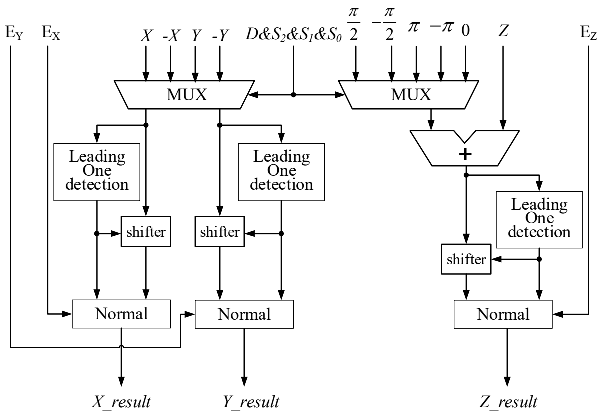


Figure 7. Hardware structure of post-processing module

4.4. *Proposed unified reconfigurable CORDIC architecture*

In rotation module, according to the dynamically configurable features of FPGA, we propose a pipeline-parallel mixed architecture, 1st to 12th iterative units are same in different modes, where all of them adopt rotation unit A to implement in pipeline. The differences are mainly reflected in the 13th to 24th units, as shown in Table 6. Module II is used to complete the compensation of scaling factor K in square root operation. The unified reconfigurable architecture is shown in Figure 8.

Table 6. Module Selection

Operation	Module Selection of 13-24 layers	Result output
Cosine/Sine	Module III	*X_result/Y_result*
Arctangent	Module I	*Z_result*
Square Root	Module II	*X_result*
Multiplication	Module III	*Y_result*
Division	Module I	*Z_result*

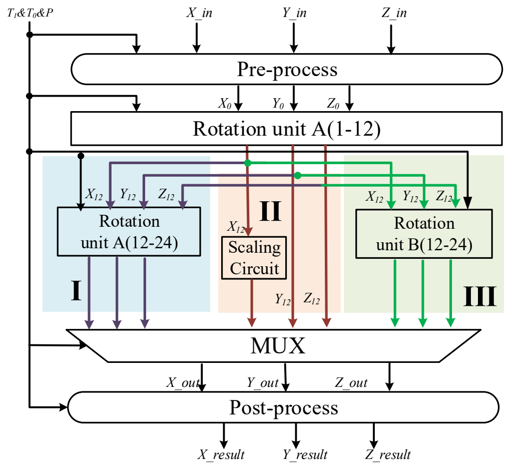


Figure 8. The unified reconfigurable CORDIC processor architecture

5. FPGA implementation results and comparison

5.1. *Synthesis results*

The reconfigurable CORDIC processor is coded in VHDL and synthesized using the Xilinx ISE 14.7 development tool. In order to achieve a fair comparison with other references, we choose Virtex5 XC5VFX130T FPGA as platform for implementation. The input data of processor uses single precision floating-point numbers, the X, Y and Z path in CORDIC rotation unit use 25-bit fixed-point numbers. Table 7 shows the FPGA resource occupation and a comparison with several previous works. We use the average relative error(ARE) in different modes for accuracy analysis, the relative error is expressed by formula (5), where, $R_{results}$ is the hardware simulation results of our designed processor. We can see, in the condition of same ARE, the LUT and register consumption are less than that of the related design described in [15], [17] and [8]. Compared with references [16], the proposed design can achieve higher frequency.

In order to compare with the unified fixed-point processor, we synthesis the rotation unit module separately, as shown in Table 8, when the data format is 16-bit fixed point and the iterative number is 17, the total consumption of LUTs and registers is similar with [6] and [7].However, [6] and [7] can only operate in circular and hyperbolic coordinate system, in addition to these two systems, our design also integrates linear coordinate system, which increase the stress of place and route, the max working frequency is lower than literature [6], but it is higher than [7] due to a simple hardware architecture design. When the data format is 25-bit fixed point, the iterative number is 24, the resource consumption is slightly larger, however, it can achieve higher precision. Therefore, the proposed processor can make a good compromise in resources, accuracy and speed.

Table 7. Comparison of resources with previous floating-point CORDIC processor

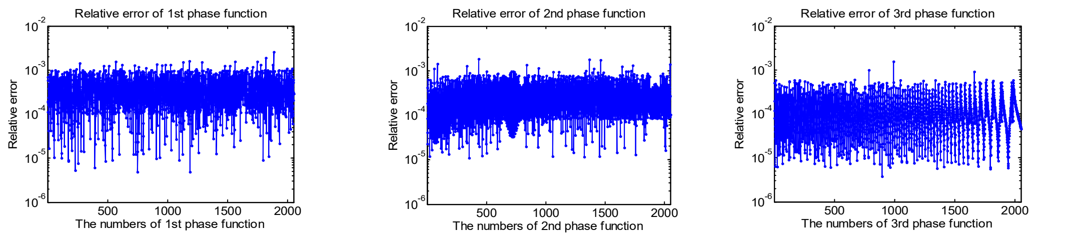
Method	Year	LUTs	Registers	Frequency (MHz)	ARE
[15]	2015	4361	410	125.00	
[16]	2011	2151	1535	133.80	
[17]	2010	3538	/	85.9	10^{-6}
[8]	2008	5412	5130	217.24	
proposed	2018	3528	3286	238.65	

Table 8. Resources comparison of rotation unit module

Data Format (fixed-point)	Method	Year	LUTs	Registers	Frequency (MHz)	ARE
16-bits	[6]	2014	1460	898	323.76	10^{-4}
	[7]	2016	2161	369	142.14	10^{-3}
16-bits	proposed	2018	1458	932	245.35	10^{-4}
25-bits	proposed	2018	2169	1687	243.78	10^{-6}

5.2. Precision analysis

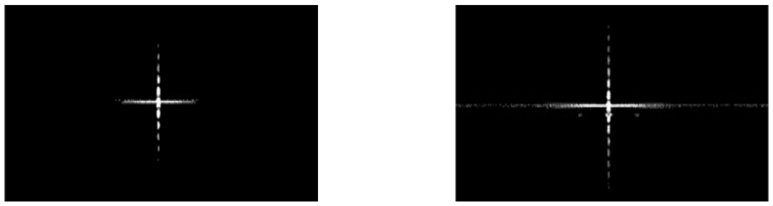
To verify the precision of our design, we apply the proposed processor to SAR imaging system, test scenario is 16384×16384 points target scene, the SAR imaging system is implemented in Virtex7 XC7VX690T FPGA. Due to the large data granularity, we adopt region-constant phase multipliers [18], the phase functions in the chirp scaling imaging algorithm are extracted by 8:1, the number of each phase function is 2048. We compare the three phase functions' hardware simulation results $R_{results}$ with the corresponding MATLAB simulation results $R_{arithmetic}$, on the basis of formula (5), calculating the maximum relative error for each factor line, as shown in Figure 9. We can see the value is between 10^{-3} and 10^{-5}, which is acceptable in high-resolution imaging. Generally, the precision of phase function mainly influences the imaging quality. Thus, we take advantage of the image quality assessment methodology described in [11] with integrity SAR imaging procedure to experiment the proposed phase function result.



(**a**) 1st phase function errors (**b**) 2nd phase function errors (**c**) 3rd phase function errors

Figure 9. The maximum relative errors of three phase functions

The imaging results of MATLAB and FPGA are shown in Figure 10 (a), (b), respectively, Table 9 shows the result of the point target imaging quality assessment. The peak side-lobe ratio (PSLR), integrated side-lobe ratio (ISLR) and resolution(RES) are commonly adopted to evaluate the imaging quality [11]. We can see that the imaging quality is good, approximate 2-dB degrade of PSLR and ISLR meet the requirements of the engineering indicators.



(a) MATLAB imaging **(b)** FPGA imaging

Figure 10. Point target imaging result

Table 9. Point target evaluation result

	Range			Azimuth		
	PSLR(dB)	**ISLR(dB)**	**Res.(m)**	**PSLR(dB)**	**ISLR(dB)**	**Res.(m)**
MATLAB	-12.61	-10.05	1.90	-12.52	-11.41	2.10
FPGA	-10.12	-9.13	2.40	-10.81	-9.83	2.73

6. Conclusions

In this paper, we design a unified reconfigurable floating-point CORDIC processor, which can be operated in different modes and calculate varieties of floating-point arithmetic. It replaces multiple single-mode processors and reduces the numbers of processors needed in the operation. Besides, the ROC is extended and multiple operations are integrated. A reconfigurable pipeline-parallel mixed architecture is proposed in rotation module to adapt different floating-point operations. Compared to traditional CORDIC processor, it greatly saves hardware resources and has high accuracy, with an acceptable maximum working frequency. Besides, the relative error of each phase function line between hardware and software computation is acceptable, and the corresponding SAR imaging result is good.

Funding: This work was supported by the National Natural Science Foundation of China under Grant 91438203, the Chang Jiang Scholars Program under Grant T2012122 and the Hundred Leading Talent Project of Beijing Science and Technology under Grant Z141101001514005.

Author Contributions: Linlin Fang and Bingyi Li conceived and developed the ideas behind the research. Linlin Fang performed the theoretical deviations, experiments, and simulations. Linlin Fang wrote the paper, Yizhuang Xie provided guidance and key suggestions. He Chen supervised the research, and revised the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Walther J S. A unified algorithm for elementary functions. Spring Joint Computer Conference. ACM, May 18-20, 1971,Vol.38, pp.379-385 (DOI: 10.1145/1478786.1478840).
2. Lightbody G, Woods R, Walke R. Design of a parameterizable silicon intellectual property core for QR-based RLS filtering. *Very Large Scale Integration Systems IEEE Transactions on*, **2003**, 11(4), pp.659-678 (DOI: 10.1109/TVLSI.2003.816142).
3. Cavallaro J R, Luk F T. CORDIC arithmetic for an SVD processor. *Journal of Parallel & Distributed Computing*, **1988**, 5(3), pp.113-120(DOI: 10.1109/ARITH.1987.6158686).
4. Lang T, Antelo E. High-Throughput CORDIC-Based Geometry Operations for 3D Computer Graphics. *IEEE Transactions on Computers*, **2005**, 54(3), pp.347-361(DOI: 10.1109/TC.2005.53).
5. Kameyama M, Amada T, Higuchi T. Highly parallel collision detection processor for intelligent robots. *Solid-State Circuits, IEEE Journal of*, **1992**, 27(4), pp.500-506(DOI: 10.1109/VLSIC.1991.760063).

6. Aggarwal S, Meher P K. Reconfigurable CORDIC architectures for multi-mode and multi-trajectory operations. IEEE International Symposium on Circuits and Systems. IEEE, 2014, pp.2490-2494(DOI: 10.1109/ISCAS.2014.6865678).
7. Aggarwal S, Meher P K, Khare K. Concept, Design, and Implementation of Reconfigurable CORDIC. *IEEE Transactions on Very Large Scale Integration Systems*, **2016**, 24(4), pp.1588-1592(DOI: 10.1109/TVLSI.2015.2445855).
8. Zhou J, Dong Y, Dou Y, et al. Dynamic Configurable Floating-Point FFT Pipelines and Hybrid-Mode CORDIC on FPGA. 2008(DOI: 10.1109/ICESS.2008.95).
9. Juang T B, Hsiao S F, Tsai M Y. Para-CORDIC: parallel CORDIC rotation algorithm. *Circuits & Systems I Regular Papers IEEE Transactions on*, **2004**, 51(8), pp.1515-1524(DOI: 10.1109/TCSI.2004.832734).
10. Runge H, Bamler R. A Novel High Precision SAR Focussing Algorithm Based On Chirp Scaling. Geoscience and Remote Sensing Symposium, IGARSS '92. International. IEEE, 1992, pp.372-375(DOI: 10.1109/IGARSS.1992.576715).
11. Zhang H, Li Y, Su Y. SAR image quality assessment using coherent correlation function. 2012(DOI: 10.1109/CISP.2012.6469650).
12. Meher P K, Valls J, Juang T B, et al. 50 Years of CORDIC: Algorithms, Architectures, and Applications. *IEEE Transactions on Circuits & Systems I Regular Papers*, **2009**, 56(9), pp.1893-1907(DOI: 10.1109/TCSI.2009.2025803).
13. Hu Y H. The quantization effects of the CORDIC algorithm. *Signal Processing IEEE Transactions on*, **1992**, 40(4), pp.834-844(DOI: 10.1109/78.127956).
14. IEEE. “IEEE Standard for Floating-Point Arithmetic,” IEEE Std 754-2008, 2008, pp.1-82(DOI: 10.1109/IEEESTD.2008.4610935).
15. Mack J, Bellestri S, Llamocca D. Floating point CORDIC-based architecture for powering computation. International Conference on Reconfigurable Computing and Fpgas. IEEE, 2015 (DOI: 10.1109/ReConFig.2015.7393311).
16. Surapong P, Glesner M. Pipelined Floating-Point Architecture for a Phase and Magnitude Detector Based on CORDIC. International Conference on Field Programmable Logic and Applications. IEEE Computer Society, 2011, pp.382-384(DOI: 10.1109/FPL.2011.74).
17. Munoz D M, Sanchez D F, Llanos C H, et al. FPGA based floating-point library for CORDIC algorithms. Programmable Logic Conference. IEEE, 2010, pp.55-60(DOI: 10.1109/SPL.2010.5483002).
18. Wen Y, He C, Xie Y Z, et al. A New Chirp Scaling Algorithm Using Region-Constant Phase Multipliers. Transactions of Beijing Institute of Technology, 2014(DOI: 10.15918/J.TBIT1001-0645.2014.03.017).