sensors

MDPI

Article

A Low-Latency RDP-CORDIC Algorithm for Real-Time Signal Processing of Edge Computing Devices in Smart Grid Cyber-Physical Systems

Mingwei Qin [1,2], Tong Liu [1,2,*], Baolin Hou [1,2], Yongxiang Gao [1,2], Yuancheng Yao [1,2] and Haifeng Sun [3]

[1] School of Information Engineering, Southwest University of Science and Technology, Mianyang 621000, China
[2] Robot Technology Used for Special Environment Key Laboratory of Sichuan Province, Mianyang 621000, China
[3] School of Computer Science and Technology, Southwest University of Science and Technology, Mianyang 621000, China
* Correspondence: liutong@mails.swust.edu.cn; Tel.: +86-177-9313-8400

Abstract: Smart grids are being expanded in scale with the increasing complexity of the equipment. Edge computing is gradually replacing conventional cloud computing due to its low latency, low power consumption, and high reliability. The CORDIC algorithm has the characteristics of high-speed real-time processing and is very suitable for hardware accelerators in edge computing devices. The iterative calculation method of the CORDIC algorithm yet leads to problems such as complex structure and high consumption of hardware resource. In this paper, we propose an RDP-CORDIC algorithm which pre-computes all micro-rotation directions and transforms the conventional single-stage iterative structure into a three-stage and multi-stage combined iterative structure, thereby enabling it to solve the problems of the conventional CORDIC algorithm with many iterations and high consumption. An accuracy compensation algorithm for the direction prediction constant is also proposed to solve the problem of high ROM consumption in the high precision implementation of the RDP-CORDIC algorithm. The experimental results showed that the RDP-CORDIC algorithm had faster computation speed and lower resource consumption with higher guaranteed accuracy than other CORDIC algorithms. Therefore, the RDP-CORDIC algorithm proposed in this paper may effectively increase computation performance while reducing the power and resource consumption of edge computing devices in smart grid systems.

Keywords: smart grid; edge computing; signal processing; CORDIC; scaling factor

Citation: Qin, M.; Liu, T.; Hou, B.; Gao, Y.; Yao, Y.; Sun, H. A Low-Latency RDP-CORDIC Algorithm for Real-Time Signal Processing of Edge Computing Devices in Smart Grid Cyber-Physical Systems. *Sensors* **2022**, *22*, 7489. https://doi.org/10.3390/s22197489

Academic Editors: Peng Zeng, Ning Zhang, Lei Liu, Chunhe Song and Juan M. Corchado

Received: 2 August 2022
Accepted: 28 September 2022
Published: 2 October 2022

1. Introduction

Over the past few years, cloud computing infrastructure has been the dominant solution used to handle heavy computational tasks related to smart grid applications [1]. With the rise of smart grid cyber–physical systems and the growing number of Internet of Things (IoT) devices, cloud computing can no longer satisfy all the computing needs of smart grid applications. Edge computing can move data processing tasks from remote cloud computing centers to devices at the edge of the network. Edge computing technology alleviates network congestion, latency and packet loss in smart grid architectures under cloud computing [2,3]. The architecture of edge-enabled smart grid cyber-physical systems is shown in Figure 1, which consists of an access layer, an edge layer, a network layer, a platform layer, and an application layer. Here, the ability to process intelligent edge data is provided by the edge computing devices at the Edge Layer.

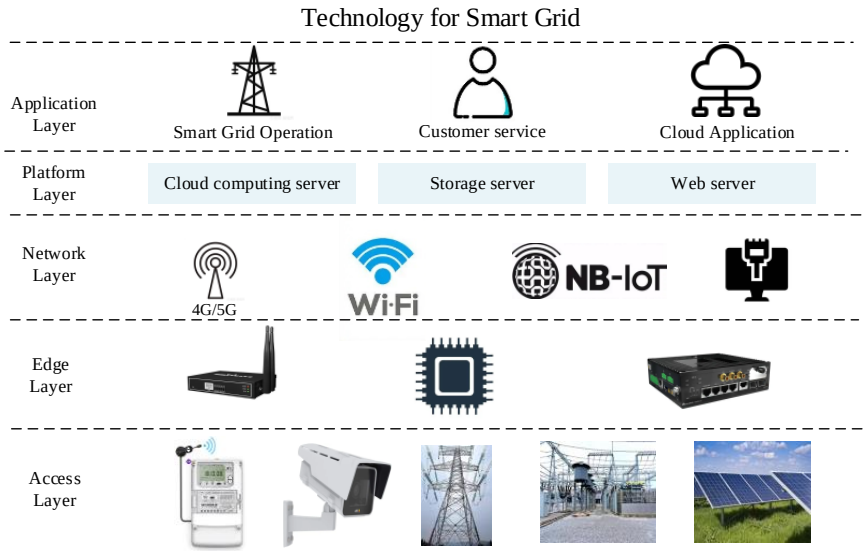

Figure 1. Basic Structure of Edge-enabled Smart Grids.

Many scholars have proposed applications of edge computing, which are especially suitable for smart grids. A cloud edge collaborative intelligent method for object detection was proposed in the literature [4], and it is applied to insulator string recognition defect detection in the power IIoT. A fault detection method for pumping units based on edge intelligence, which effectively improves the fault detection accuracy while maintaining low computational requirements, was proposed in [5]. The important features of edge computing applied to smart grid mainly include: support for real-time [6–10] and low power [11–15] consumption. Data processing is the most time-consuming and energy-intensive part of edge computing. Furthermore, since edge computing devices cannot guarantee high-capacity storage, processing these large volumes of data is an important issue to be addressed [16].

Most of the current edge computing devices use an edge computing framework based on heterogeneous computing, as shown in Figure 2. In this framework, the computational power provided by edge devices mainly depends on hardware accelerators, which include Digital Signal Processing (DSP), Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA). In the heterogeneous framework based on CPU+FPGA [17], FPGA has the characteristics of reconfiguration and energy efficiency. The literature [18] appropriately places DSP operators on edge devices, so that the edge layer can reduce the energy consumption of each event by as much as 4%. The edge computing device based on CPU+ASIC structure [19] has a significantly better acceleration ratio than the existing GPU+CPU method, and has the advantages of small size and low power consumption. A signal processing algorithm named CORDIC is often used in hardware to handle complex data computation problems in real-time. It can implement many complex functions and mathematical problems with simple addition, subtraction, and shift operations. Table 1 lists some applications of the CORDIC algorithm, including trigonometric functions [20], hyperbolic functions [21], FFT [22] and singular value decomposition [23]. Nonetheless, the computational speed of the conventional CORDIC algorithm is limited by the number of iterations, i.e., the more iterations of the CORDIC algorithm, the higher the computational accuracy and the longer the time delay. Therefore, reducing the number of iterations of the CORDIC algorithm, while ensuring the computational accuracy of the algorithm, can reduce the computational latency and hardware resource consumption.

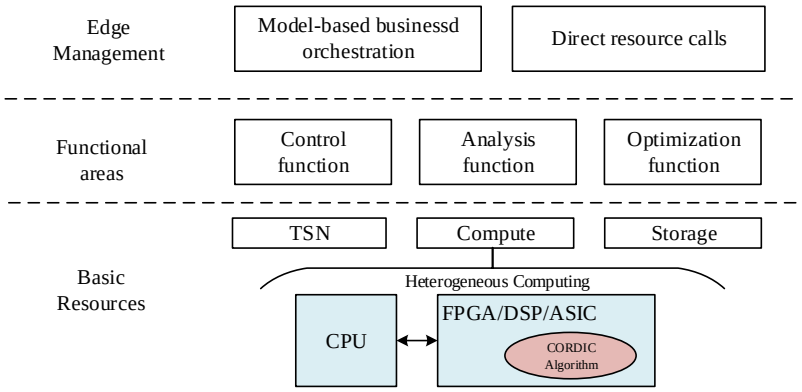

Figure 2. Edge computing framework based on heterogeneous computing.

Table 1. Application of CORDIC Algorithm.

Application Category	Functions Implemented
basic arithmetic	multiplication
	division
trigonometric function	sin x
	cos x
	tan x
inverse trigonometric function	arcsin x
	$\arcsin^{-1} x$
	$\arctan^{-1} x$
hyperbolic function	cosh x
	sinh x
	tanh x
	$\tanh^{-1} x$
other common functions	$\sqrt{x}$
	In(x)
	e^x
other applications	Fast Fourier transform
	Matrix eigenvalue estimation
	Singular value decomposition
	Digital frequency synthesis

In summary, this work aimed to discover a high-performance CORDIC algorithm. As a result, we proposed a RDP-CORDIC algorithm and implemented the hardware design of the algorithm. The RDP-CORDIC algorithm, characterized by fewer iterations, less hardware resource consumption and faster processing speed, can effectively improve the data processing speed and reduce the latency and power consumption of edge computing devices in smart grid cyber-physical systems. The main contributions of this work are as follows:

- We proposed a rotation direction prediction method of the CORDIC algorithm, which completed the calculation of all the micro-rotation directions by inputting the angle and direction prediction constants, providing the basis for the subsequent merge iteration;
- A constant compensation algorithm for direction prediction was proposed to achieve higher accuracy of direction prediction, being able to solve the problem of large memory consumption under the condition of high accuracy;
- The single-stage iterative structure of the CORDIC algorithm was replaced by a three-stage and multi-stage iterative structure. Based on this structure, the CORDIC algorithm design with high accuracy, low latency, and low power consumption was achieved.

2. Related Work

The CORDIC algorithm was proposed by Volder in 1959 and was later generalized by Walther. Subsequently, some other methods were proposed that aimed to enhance the precision and reduce iterations and resource consumption. Among them, Radix-4 CORDIC algorithms [24] worked on zero hopping technology to reduce the number of iterations for rotation to 50%. Later, a hybrid radix 2–4 CORDIC algorithm with high-performance compensation technique waspresented [25] with reduced number of iterations by 1/4, including scale factor calculation and compensation. Nevertheless, the computation and correction of variable scale factor was a focused issue for higher radix CORDIC algorithms [26–29] and advanced hybrid CORDIC algorithms [30]. The scale-free CORDIC algorithm [31,32] approximated the sine and cosine functions by the Taylor series, thereby eliminating the need for the scalar factors, except for a limited convergence range and poor accuracy. A new hybrid CORDIC algorithm was proposed [33] to be able to further reduce the latency of CORDIC by reducing the number of iterations equal to $(3N/8) + 1$. A technique reported in low latency CORDIC algorithm [34] utilized the binaryto-bipolar recoding (BBR) method to reduce the overall iterations to $(N + 1)/3$, with no scale factor compensation. Similar to [23], the CORDIC algorithms [35–37] cut down time and memory at the expense of accuracy. The CORDIC II algorithm proposed in [38] had excellent performance in terms of resource consumption and latency, but its low accuracy held it back. Table 2 lists some important features of the above related CORDIC algorithms, including the rotation radix, prediction of rotation direction and whether the scaling factor is fixed.

Table 2. Features of related CORDIC algorithms.

CORDIC Algorithm	Radix	Rotation Direction Prediction	Fixed Scaling Factor
R-2 CORDIC [26]	R-2	×	√
R-4 CORDIC [24]	R-4	×	×
R-8 CORDIC [28]	R-8	×	×
scaling-free CORDIC [31]	MIX-R	×	√
Mixed-R-scaling-free CORDIC [29]	MIX-R	×	√
BBR-CORDIC [34]	R-2	√	×
CORDIC II [38]	R-2	×	×
RDP-CORDIC [proposed]	R-2	√	√

3. Conventional CORDIC Algorithm

The CORDIC algorithm contains two modes (rotation mode, vector mode) and three coordinate systems (circular coordinates, linear coordinates, hyperbolic coordinates). Different functions can be derived under different modes and different coordinate systems. The CORDIC algorithm rotation mode of the circle coordinate system was taken as an example to construct the simplest vector rotation model, as shown in Figure 3.

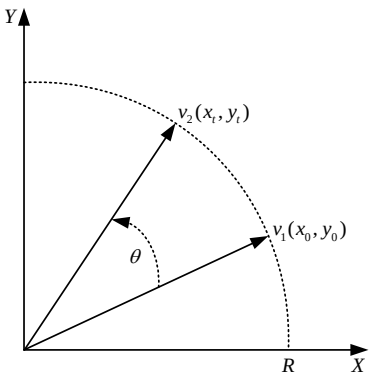

Figure 3. Rotation model for CORDIC algorithm.

Suppose vector v_1 is rotated by θ to obtain vector v_2, and the coordinates of v_1 and v_2 are (x_0, y_0), (x_t, y_t) respectively, then, the equation of change of vector coordinates can be expressed by Equation (1).

$$\begin{cases} x_t = x_0 \cos\theta - y_0 \sin\theta = \cos\theta(x_0 - y_0 \tan\theta) \\ y_t = x_0 \sin\theta + y_0 \cos\theta = \cos\theta(y_0 + x_0 \tan\theta) \end{cases} \tag{1}$$

By dividing the single rotation angle of Equation (1) into multiple directed rotations $\theta_i = \tan^{-1}(2^{-i})$, each rotation can be expressed by the iterative Equation (2).

$$\begin{cases} x_{i+1} = \cos\theta_i(x_i - d_i \cdot y_i \cdot 2^{-i}) \\ y_{i+1} = \cos\theta_i(y_i + d_i \cdot x_i \cdot 2^{-i}) \\ z_{i+1} = z_i - d_i \cdot \theta_i \end{cases} \quad \begin{matrix} if(z_i > 0) \quad d_i = 1 \\ else \quad\quad d_i = -1 \end{matrix} \tag{2}$$

where z_{i+1} indicates the remaining angle and d_i is the direction of rotation. (x_{i+1}, y_{i+1}) indicates the coordinates of (x_i, y_i) after the next rotation. In the rotation mode, the remaining angle z_{i+1} value was used as a direction reference, and after n iterations, the z_{i+1} value tended to zero and the vector v_i almost tended to the vector, thus realizing the successive approximation calculation.

Since $\cos\theta_i$ in Equation (2) involved multiplication in the iterative calculation process, it can be proposed not to participate in the iterative operation. Let $K = \prod_{i=0}^{n} \cos\theta i = 1/[(1 + 2^{-2i})^{0.5}]$, $1/K$ is the scaling factor mentioned above, then the iteration equations of the radix-2 CORDIC algorithm in rotation mode at the $(i + 1)$th step are as follows:

$$\begin{cases} x_{i+1} = x_i - d_i \cdot y_i \cdot 2^{-i} \\ y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i} \\ z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}) \end{cases} \quad \begin{matrix} if(z_i > 0) \quad d_i = 1 \\ else \quad\quad d_i = -1 \end{matrix} \tag{3}$$

To facilitate the subsequent verification of the performance of the CORDIC algorithm, the principles for computing the sine and cosine functions are described below. Combining Equation (1) with Equation (3), a formula for the CORDIC algorithm that calculates the sine and cosine functions can be introduced. If given $z_0 = \theta$, the coordinates of Equation (3) are (x_n, y_n) after n iterations of calculation.

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} \approx \frac{1}{K} \cdot \begin{pmatrix} x_0 \cos\theta - y_0 \sin\theta \\ x_0 \sin\theta + y_0 \cos\theta \end{pmatrix} \tag{4}$$

From Equation (4), it can be found that taking $x_0 = K$ and $y_0 = 0$, after n iterations of calculation, x_n and y_n will be equal to the values of $\cos\theta$ and $\sin\theta$, respectively. Therefore, the calculation of sine and cosine functions based on the CORDIC algorithm was implemented.

4. RDP-CORDIC Algorithm

The micro-rotation direction of the conventional CORDIC algorithm is determined by the remaining angle after the last iteration, which leads to the problem of high latency. Although the high latency problem may be solved by way of a parallel pipeline structure, it increases the hardware resource overhead, and the most effective way to solve the high latency is to reduce the number of iterations. In this paper, a rotation direction prediction CORDIC (RDP-CORDIC) algorithm was proposed to reduce the number of iterations by calculating all micro-rotation directions in advance, so that the conventional single-stage iterative structure could be changed into a multi-stage iterative structure. The current direction prediction algorithms mainly include the Booth encoding method and the binary-to-bipolar recoding (BBR) method. The Booth encoding method is responsible for predicting the direction of rotation after $[N - \log_2{}^3]/3$ iterations, thus reducing the number of iterations by about $1/2$. The BBR is impressed by decomposing the input angle θ into a combination of a larger angle and several 2^{-i} radians so that the direction of rotation is determined by the binary bit value of θ each rotation. Note that, the BBR method

requires a ROM to store all the computation results after $N/3-1$ iterations, and the ROM consumption increases as precision gets higher, e.g., 16-bit precision requires a ROM of $26 \times 16 \times 2$ (bit) size.

4.1. Rotation Direction Prediction

Considering that the BBR method allows the binary bit value of angle to represent the direction of micro-rotation, i.e., $\theta = \sum_{i=0}^{\infty} d_i 2^{-i} = (d_\theta)_2$, this method fixes the rotation angle as 2^{-i}, resulting in large consumption of ROM resources. Therefore, the micro-rotation angle chosen for the RDP-CORDIC algorithm was $\tan^{-1}(2^{-i})$, and a new rotation direction prediction method needs to be sought.

The input angle $\theta \in [0, \pi/4]$ can be expressed as:

$$\theta = \sum_{i=0}^{n=\infty} \sigma_i \tan^{-1}\left(2^{-i}\right) \tag{5}$$

where, $\sigma_i \in \{-1, 1\}$, $i \geq 1$. Let $\sigma_i = 2d_i - 1$, $d_i \in \{0, 1\}$, at which point s = 2, then $\theta = \sum_{i=1}^{n=\infty}(2d_i - 1)\tan^{-1}\left(2^{-i}\right)$, and Equation (6) could be derived.

$$\begin{aligned} \theta &= \sum_{i=1}^{\infty}(2d_i - 1)\tan^{-1}(2^{-i}) \\ &= \sum_{i=1}^{\infty}(2d_i - 1)(2^{-i} - \tfrac{2^{-3i}}{3} + \tfrac{2^{-5i}}{5} - \tfrac{2^{-7i}}{7} + \cdots) \\ &= \sum_{i=1}^{\infty}(2d_i - 1)(2^{-i}) - \sum_{i=1}^{\infty}(2d_i - 1)(\tfrac{2^{-3i}}{3} - \tfrac{2^{-5i}}{5} + \tfrac{2^{-7i}}{7} - \cdots) \\ &= 2d_\theta - 1 + \sum_{i=1}^{\infty}(2^{-i} - \tan^{-1}(2^{-i})) - 2\sum_{i=1}^{\infty} d_i(2^{-i} - \tan^{-1}(2^{-i})) \end{aligned} \tag{6}$$

Let $\varepsilon = \sum_{i=1}^{\infty}\left(2^{-i} - \tan^{-1}\left(2^{-i}\right)\right)$, $\lambda = \sum_{i=1}^{\infty} d_i(2^{-i} - \tan^{-1}(2^{-i}))$, it came from Taylor Formula that when $i \geq [(N - \log_2 3)/3]$, $2^{-i} \cong \tan^{-1}\left(2^{-i}\right)$, λ could be reduced to Equation (7), where [*] means to take an integer greater than or equal to *.

$$\lambda = \sum_{i=1}^{[(N-\log_2 3)/3]} d_i(2^{-i} - \tan^{-1}(2^{-i})) \tag{7}$$

ε is a constant of about 0.0421115429, the final rotation direction prediction formula is introduced as in Equation (8).

$$\begin{aligned} d_\theta &= 0.5\theta + 0.5 - 0.5\varepsilon + \lambda \\ &= 0.5\theta + \lambda + 0.478944228537446 \end{aligned} \tag{8}$$

From Equation (8), the direction of rotation could be calculated by entering the angle and λ. The binary bit value of the final calculation pointed the direction of rotation. Equation (6) gave the calculation of the value of d_1, d_2, d_3, d_4 and d_5 in various combinations for 16-bit precision. In order to determine the rules for the value of λ, the cumulative value of the rotation angle corresponding to λ was viewed as the angle reference, which was denoted as θ_{cp}. It should be noted that in the calculation, the values for d_6~d_{16} were 0. When calculating θ_{cp}, not only the sum of the angles of d_1~d_5 rotation may be covered, but also the micro-selected rotation angles of d_6~d_{16} should be accumulated. Equation (9) is the calculation of the reference angle value $\theta_{\mathrm{cp}(m)}$.

$$\theta_{\mathrm{cp}(m)} = \sum_{i=1}^{m}(2d_i - 1)\tan^{-1}(2^{-i}) - \sum_{\mathrm{i}=m+1}^{16}\tan^{-1}(2^{-i}) \tag{9}$$

Looking at the interval range of the input angle size, the redundant data is removed and the final direction prediction constants are shown in Table 3. The final direction

calculation result is affected by the accuracy of the direction prediction constants λ and θ. In order to satisfy the accuracy of d_θ, it is necessary to make the accuracy of λ higher than that of d_θ. For the 16-bit precision of d_θ, λ needs 17-bit size precision. Since the integer bits of both λ and θ are 0, each prediction constant in ROM only needs 16 bits in size. Therefore, to implement the RDP-CORDIC algorithm with N bit precision, the size of the prediction constants λ and θ in ROM is also the same as N bit.

Table 3. Prediction constant table for 16-bit precision direction.

$\{d_1,d_2,d_3,d_4,d_5\}$	λ	θ_{cp5}
01111	0.03635239	−0.0305780
10000	0.03636256	0.03190169
10001	0.03643358	0.09425964
10010	0.03644375	0.15673931
10011	0.03699740	0.21813201
10100	0.03700756	0.28061168
10101	0.03707859	0.34296963
10110	0.03708875	0.40544930
10111	0.04137373	0.45937935
11000	0.04138389	0.52185901
11001	0.04145492	0.58421697
11010	0.04146508	0.64669663
11011	0.04201873	0.70808934
11100	0.04202890	0.77056900

The minimum angular reference value λ_{cp5} for different values of λ is given in Table 3. The process of rotation direction prediction can be summarized as follows:

1. Compare the input angle with θ_{cp} in the direction prediction constant, and select the value of λ corresponding to a value close to and less than or equal to θ_{cp};
2. The binary value d_θ representing the micro-rotation direction was calculated based on λ. Finally, the prediction of the micro-rotation direction in the non-iterative case was performed.

4.2. ROM Resource Optimization

A 14 × 16 × 2 bit ROM resource was required to make the above-mentioned 16-bit precision direction prediction. According to the theory of rotation direction prediction algorithm proposed in Section 4.1, the N bit width accuracy required a ROM of $2^{[(N-\log_2 3)/3]} \times N$ bit size, and the ROM consumption increased sharply with the increase of accuracy. The reason for the sharp increase of ROM consumption was that the high accuracy direction prediction asked for more λ values to be selected, leading to an increase in the table of direction prediction constants. It may be useful to analyze the λ expansion and let $m = [(N - \log_2 3)/3)]$, then λ_m and λ_{m+1} are as in Equations (10) and (11), respectively.

$$\lambda_m = \sum_{i=1}^{m} d_i(2^{-i} - \tan^{-1}(2^{-i})) \tag{10}$$

$$\begin{aligned} \lambda_{m+1} &= \sum_{i=1}^{m+1} d_i(2^{-i} - \tan^{-1}(2^{-i})) \\ &= \lambda_m + d_{m+1}(2^{-m-1} - \tan^{-1}(2^{-m-1})) \end{aligned} \tag{11}$$

Let $\mu_i = 2^{-i} - \tan^{-1}(2^{-i})$, and the first 10 iterations of μ_i are given in Table 4.

Table 4. Value of μ_i constants for the first 10 iterations.

i	μ_i
1	0.0363523910
2	0.0050213369
3	0.0006450055
4	$8.1190004043 \times 10^{-5}$
5	$1.0166569732 \times 10^{-5}$
6	$1.2713795232 \times 10^{-6}$
7	$1.5893989889 \times 10^{-7}$
8	$1.9868033028 \times 10^{-8}$
9	$2.4835211812 \times 10^{-9}$
10	$3.1044068054 \times 10^{-10}$

Combined with Taylor's formula, when $m \geq [(N + \log_2(3/20) - 3)/5]$, Equation (11) can be reduced to Equation (12)

$$\lambda_{m+1} = \lambda_m + d_{m+1} \cdot 2^{-3} \cdot \mu_m \tag{12}$$

Equation (12) is the relationship between λ_{m+1} and λ_m, and similarly, the value of λ_{m+i} can be calculated from λ_m. Thus, an accuracy compensation algorithm for λ is proposed, where λ is composed of a fixed λ_s and an accuracy compensation λ_c.

$$\lambda = \lambda_s + \lambda_c \tag{13}$$

In Equation (13), $s = [(N + \log_2(3/20) - 3)/5]$, the accuracy compensation λ_c is calculated as Equation (14).

$$\lambda_c = \sum_{i=s+1}^{m} d_i \cdot \mu_{s+1} \cdot 2^{-3(i-s-1)} \tag{14}$$

Since the accuracy of the rotation direction that can be derived is equal to $s \times 3 + \log_2 3 > m$, d_i in Equation (14) can be calculated based on λ_s. To sum up, the ROM consumption of the direction prediction constant was reduced from $2^{[N-\log_2 3]/3} * N$ bit to $2^{[N+\log_2(3/20)-3]/5} * N$ bit by using the direction prediction constant accuracy compensation method, which achieved high accuracy and reduced the ROM consumption.

4.3. Iterative Merging

After getting the direction of rotation, the conventional single-stage iterative calculation Equation (3) now can be changed to a three-stage combined iteration, as shown in Equation (15).

$$\begin{cases} x_{i+3} = x_i(1 - \frac{d_i \cdot d_{i+1}}{2^{(2i+1)}} - \frac{d_i \cdot d_{i+2}}{2^{(2i+2)}} - \frac{d_{i+1} \cdot d_{i+2}}{2^{(2i+3)}}) - y_i \cdot \sum_{j=i}^{i+2} (2^{-j} \cdot d_j) + y_i \cdot \frac{d_i \cdot d_{i+1} \cdot d_{i+2}}{2^{(3i+3)}} \\ y_{i+3} = y_i(1 - \frac{d_i \cdot d_{i+1}}{2^{(2i+1)}} - \frac{d_i \cdot d_{i+2}}{2^{(2i+2)}} - \frac{d_{i+1} \cdot d_{i+2}}{2^{(2i+3)}}) + x_i \cdot \sum_{j=i}^{i+2} (2^{-j} \cdot d_j) - x_i \cdot \frac{d_i \cdot d_{i+1} \cdot d_{i+2}}{2^{(3i+3)}} \end{cases} \tag{15}$$

When the number of iterations $i > [(N - 3)/3]$, $x_i\ 2^{-(3i+3)}$ and $y_i\ 2^{-(3i+3)}$ dropped to machine zero, Equation (15) removed $y_i \cdot \frac{d_i \cdot d_{i+1} \cdot d_{i+2}}{2^{(3i+3)}}$ and $x_i \cdot \frac{d_i \cdot d_{i+1} \cdot d_{i+2}}{2^{(3i+3)}}$ two terms, the three-level combined iteration formula became Equation (16).

$$\begin{cases} x_{i+3} = x_i(1 - \frac{d_i \cdot d_{i+1}}{2^{(2i+1)}} - \frac{d_i \cdot d_{i+2}}{2^{(2i+2)}} - \frac{d_{i+1} \cdot d_{i+2}}{2^{(2i+3)}}) - y_i \cdot \sum_{j=i}^{i+2} (2^{-j} \cdot d_j) \\ y_{i+3} = y_i(1 - \frac{d_i \cdot d_{i+1}}{2^{(2i+1)}} - \frac{d_i \cdot d_{i+2}}{2^{(2i+2)}} - \frac{d_{i+1} \cdot d_{i+2}}{2^{(2i+3)}}) + x_i \cdot \sum_{j=i}^{i+2} (2^{-j} \cdot d_j) \end{cases} \tag{16}$$

When $i > [(N-1)/2]$, $x_i\,2^{-(2i+1)}$ and $y_i\,2^{-(2i+1)}$ are machine zeros, and the subsequent iterative process can be represented by the multilevel merge iteration of Equation (17).

$$\begin{cases} x_{i+n} = x_i - y_i \cdot \sum_{j=i}^{i+n} 2^{-j} \cdot d_j \\ y_{i+n} = y_i + x_j \cdot \sum_{j=i}^{i+n} 2^{-j} \cdot d_j \end{cases} \tag{17}$$

In summary, the flow of rotation iteration is as follows:

1. For the number of iterations $i \leq [(N-3)/3]$, the three-stage merge iteration Formula (15) was used;
2. When $[(N-3)/3] < i \leq [(N-1)/2]$, the three-stage merge iteration simplified Formula (16) was used;
3. Finally when $i > [(N-1)/2]$, Formula (16) for multi-stage merge iteration calculation was used.

5. Hardware Design of RDP-CORDIC Algorithm

The main hardware structures for implementing the CORDIC algorithm are loop iterative structures and pipelined iterative structures. The loop iterative structure is simple in design and consumes less hardware, but the computation speed slows down as the accuracy increases. The pipelined iterative structure is more complex and consumes more hardware, but the computation speed is much higher. For edge computing devices used in smart grid cyber-physical systems, the faster pipelined iterative structure is more suitable.

5.1. RDP-CORDIC Algorithm Structure Design

The pipeline structures of the classical optimized CORDIC algorithm and the RDP-CORDIC algorithm are shown in (a) and (b) of Figure 4, respectively. The direction of each rotation of the classical optimized CORDIC algorithm depends on the sign of the remaining angle θ_n, and the result needs to be iteratively calculated in a $[(N-1)/2]$ stage pipeline method. The structure of RDP-CORDIC algorithm consists of two parts: the direction prediction part on the left side and the rotation iteration on the right side. The direction prediction module calculates all micro-rotation directions in advance, while the rotation iteration part transforms the single-stage iterative structure into a three-stage and multi-stage iterative structure since all micro-rotation directions are known. The new structure cut off part of hardware overhead and latency compared to the conventional structure.

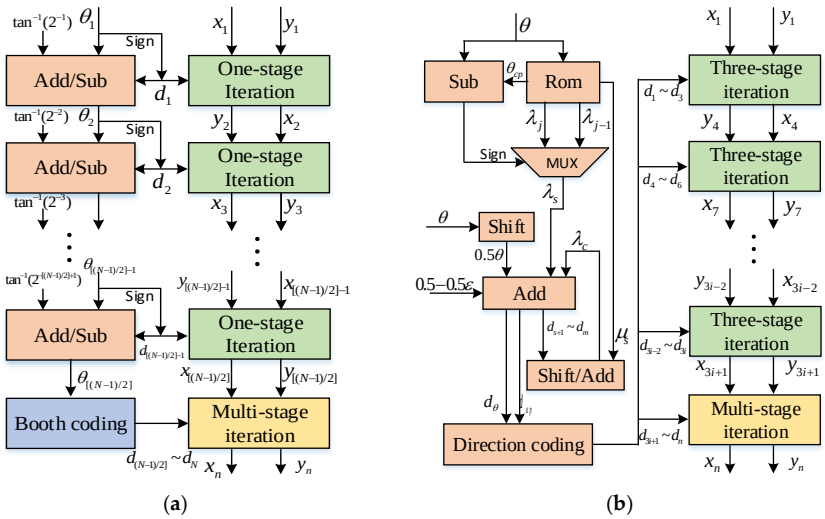

Figure 4. Structure of the classical optimized CORDIC algorithm and the RDP_CORDIC algorithm: (**a**) Classical optimized CORDIC algorithm pipeline structure; (**b**) Structure of RDP-CORDIC algorithm.

As shown below, the workflow of the RDP-CORDIC algorithm consists of three steps (Algorithm 1).

Algorithm 1 RDP-CORDIC workflow

1. Directional rough prediction
   (1) Pre-store the direction prediction constants θ_{cp}, λ_s, and μ_i in a ROM of size $2[(N - (\log2(3/20) - 3)/5]$ bits;
   (2) Use the MSB of the input angle θ as the lookup address of the ROM for reading out θ_{cp};
   (3) Send the sign bit of the value of the input angle θ minus θ_{cp} to the selection input port of the multiplexer, and the multiplexer outputs the corresponding value of λ_s;
   (4) Add up λ_s, θ, and $0.5 - 0.5\varepsilon$ to get the rough rotation direction prediction value d_{ap}.
2. Accurate direction prediction
   (1) Shift and sum up the rough rotation direction prediction d_{s+1}~d_m with μs according to Equation (14) to calculate the compensation value λ_c.
   (2) Calculate the exact direction prediction value d_θ by re-summing λ, θ and $0.5 - 0.5\varepsilon$.
3. Iteration calculation
   (1) In the iterative calculation part uses multiple three-level merge iteration modules and one multi-level merge iteration module;
   (2) Set the input values of the iterative calculation module as $x_1 = K$ and $y_1 = 0$;
   (3) The rotation directions d_1~d_{3s-1} are determined by the rough direction value dap, and the rotation directions d_{3s}~d_n are determined by the accurate direction value d_θ.

5.2. Calculation of Sine and Cosine Function Based on RDP-CORDIC Algorithm

To verify the performance of the RDP-CORDIC algorithm, the RDP-CORDIC algorithm was arranged to calculate the sine and cosine functions. Since the selected initial rotation angle value was $\tan^{-1}(2^{-i})$, where i was an integer greater than 0, the calculated angle range was limited to $[0, \pi/4]$. The symmetry of trigonometric function were combined with the trigonometric change to expand the input angle range, and the final change relationship is shown in Table 5.

Table 5. Angle interval conversion.

Input Angle Range θ_e	**Angle after Conversion** θ	$\cos\theta_e$	**sin** θ_e
$[0, \pi/4)$	θ_e	$\cos\theta$	$\sin\theta$
$[\pi/4, \pi/2)$	$\pi/2 - \theta_e$	$\sin\theta$	$\cos\theta$
$[\pi/2, 3\pi/4)$	$\theta_e - \pi/2$	$-\sin\theta$	$\cos\theta$
$[3\pi/4, \pi)$	$\pi - \theta_e$	$-\cos\theta$	$\sin\theta$
$[\pi, 5\pi/4)$	$\theta_e - \pi$	$-\cos\theta$	$-\sin\theta$
$[5\pi/4, 3\pi/2)$	$3\pi/2 - \theta_e$	$-\sin\theta$	$-\cos\theta$
$[3\pi/2, 7\pi/4)$	$\theta_e - 3\pi/2$	$\sin\theta$	$-\cos\theta$
$[7\pi/4, 2\pi]$	$2\pi - \theta_e$	$\cos\theta$	$-\sin\theta$

Based on the RDP-CORDIC algorithm, the implementation architecture of sine and cosine function calculation is shown in Figure 5, including an angle interval folding module, a direction prediction module, multiple three-stage iteration modules, a multi-stage merge iteration, and a triangular constant change module. The working principle is that the angle interval folding module transforms the input angle of any size into the interval [0~2π] and then sends the 3 bit angle range code to the angle transformation module.The rotation direction prediction module calculates all micro-rotation directions in advance based on the input angle, and then passes the direction values to the back-end iterative calculation module; after three-stage of running iterations and multi-stage of combined iterations, the calculated sine and cosine function values are output. Finally, the sine and cosine signals obtained by simulation with Vivado's Simulation software are shown in Figure 6.

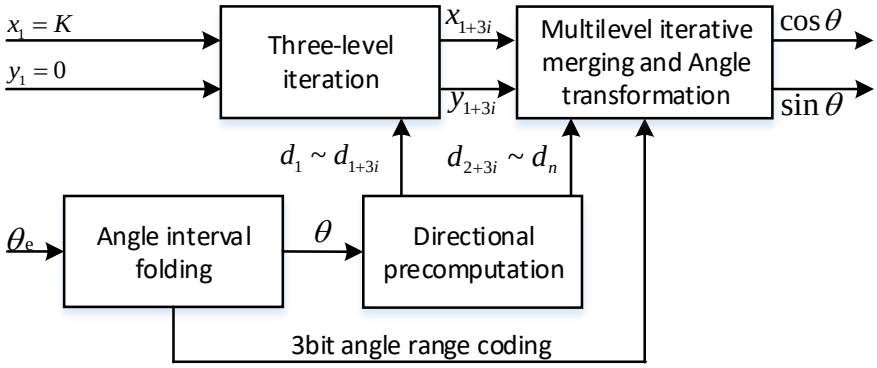

Figure 5. Implementation of RDP-CORDIC algorithm with sine and cosine functions.

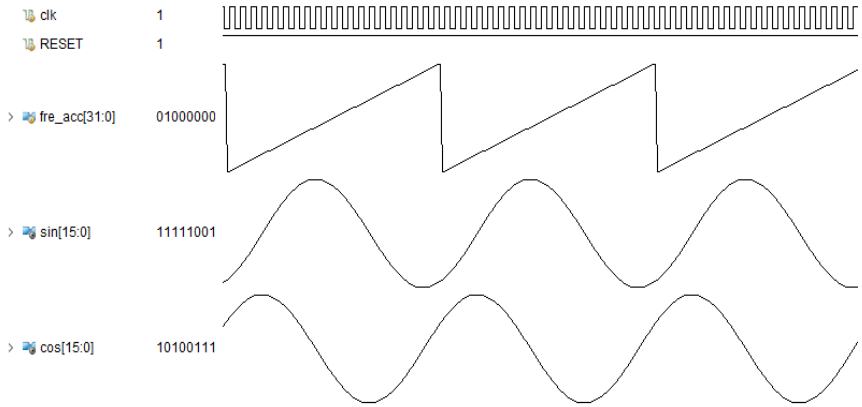

Figure 6. Simulation waveforms of calculated sine and cosine functions.

5.3. More Applications of the RDP-CORDIC Algorithm

As described in the introduction section, the RDP-CORDIC algorithm can also be used in more areas of smart grids. In a smart grid system, the frequency of the power system is an important indicator of power quality and needs to be detected in real time. If there is a problem in a section of the smart grid, the source of the fault can be cut off in time to protect the grid. Based on CORDIC algorithm to implement FFT, it can efficiently measure the higher harmonic and interference noise of power signal. The core of FFT implementation using CORDIC algorithm is to use CORDIC algorithm to implement complex multiplication operations in FFT. The complex multiplication operation in FFT is as in Equation (18).

$$X_k = X_0 * W_N^{nk} \tag{18}$$

where $X_0 = x_0 + j \times y_0$, $X_k = x_k + j \times y_k$, bringing $W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$ into Equation (18), the imaginary and real parts of X_k after being simplified are as in Equation (19).

$$\begin{cases} x_k = x_0 \cos(-\frac{2nk\pi}{N}) - y_0 \sin(-\frac{2nk\pi}{N}) \\ y_k = y_0 \cos(-\frac{2nk\pi}{N}) + x_0 \sin(-\frac{2nk\pi}{N}) \end{cases} \tag{19}$$

The multiplication of the complex sequence and the rotation factor can be seen as the vector X_0 rotated by $\theta = -2nk\pi/N$, Then, using the CORDIC algorithm idea, we can transform Equation (19) into Equation (1). So the complex multiplication of FFT can then be implemented by the RDP-CORDIC algorithm.

In addition, the CORDIC algorithm can also realize singular value decomposition (SVD) for image denoising, data compression, etc. With the increase of matrix dimension, the computation volume of SVD grows exponentially, which has a great impact on the

computation real-time of edge computing devices. Taking the 2 × 2 matrix G as an example, its bilateral Jacobi SVD algorithm was calculated as Equations (20) and (21).

$$\begin{bmatrix} \cos\theta_L & -\sin\theta_L \\ \sin\theta_L & \cos\theta_L \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \cos\theta_R & -\sin\theta_R \\ \sin\theta_R & \cos\theta_R \end{bmatrix} = \begin{bmatrix} \delta_1 & 0 \\ 0 & \delta_2 \end{bmatrix} \tag{20}$$

$$\begin{cases} \theta_R + \theta_L = \arctan(\frac{c+b}{d-a}) \\ \theta_R - \theta_L = \arctan(\frac{c-b}{d+a}) \end{cases} \tag{21}$$

In Equation (20), a, b, c, d are the four elements of the second-order matrix G. θ_R and θ_L are the left and right rotation angles, calculated by Equation (21). The values of δ_1 and δ_2 are the singular values of the matrix G. In the above operation, both the arc tangent funcion and the sine/cosine function can be implemented by the CORDIC algorithm. The structure of the 2 × 2 SVD module based on the RDP-CORDIC algorithm is shown in Figure 7.

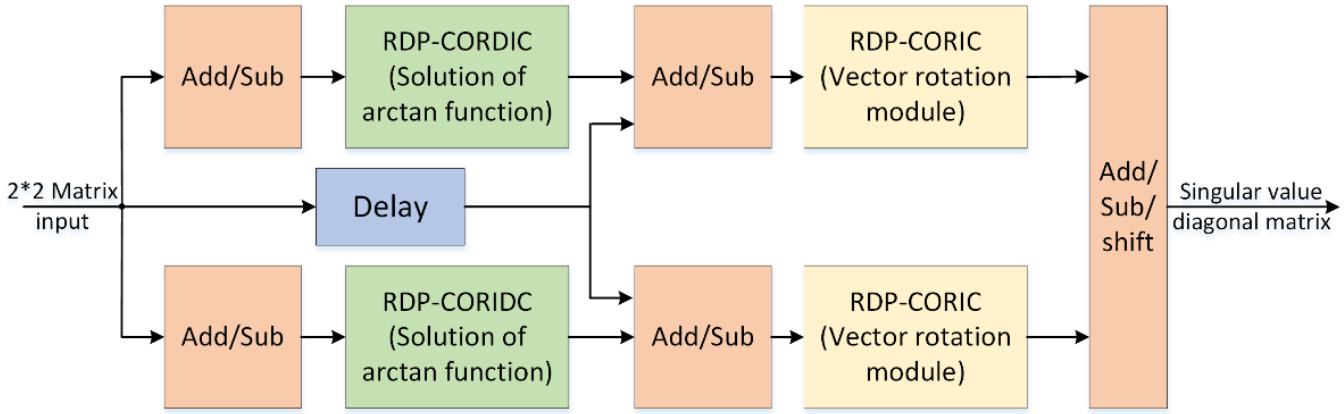

Figure 7. Structure of 2 × 2 SVD module based on RDP-CORDIC algorithm.

6. Performance Testing and Analysis

The hardware design of the 16-bit fixed-point decimal RDP-CORDIC algorithm was implemented on the Xilinx Kintex7 325T series FPGA hardware platform using Verilog HDL. In the first place, the effect of ROM resource optimization of the predictive direction CORDIC algorithm proposed in Section 4.2 was tested, and the size of RAM resources consumed before and after the optimization was compared. Subsequently, the proposed RDP-CORDIC algorithm was tested and compared with other related CORDIC algorithms in terms of latency, resource consumption, and power consumption. Based on the hardware structure of the RDP-CORDIC algorithm, the maximum absolute value errors of the sine and cosine functions, logarithmic function, square root function, hyperbolic sine, and hyperbolic cosine function were also tested at 16-bit accuracy. Finally, we analyze the time and maximum absolute value errors of the sine and cosine functions computed using the RDP-CORDIC algorithm.

6.1. ROM Optimization Results of the RDP-CORDIC Algorithm

Figure 8 shows the comparison of ROM resource consumption before and after ROM optimization for this algorithm, from which it can be seen that the ROM consumption before the optimization is much higher than that after optimization. For 16-bit precision, the unoptimized algorithm requires a ROM size of 448 bits., while the optimized algorithm requires only 160 bits. for 32-bit precision, the unoptimized algorithm requires a ROM size of 56,320 bits, while the optimized algorithm consumes only 1888 bits. as the data bit width increases, the difference in ROM consumption before and after optimization increases significantly.

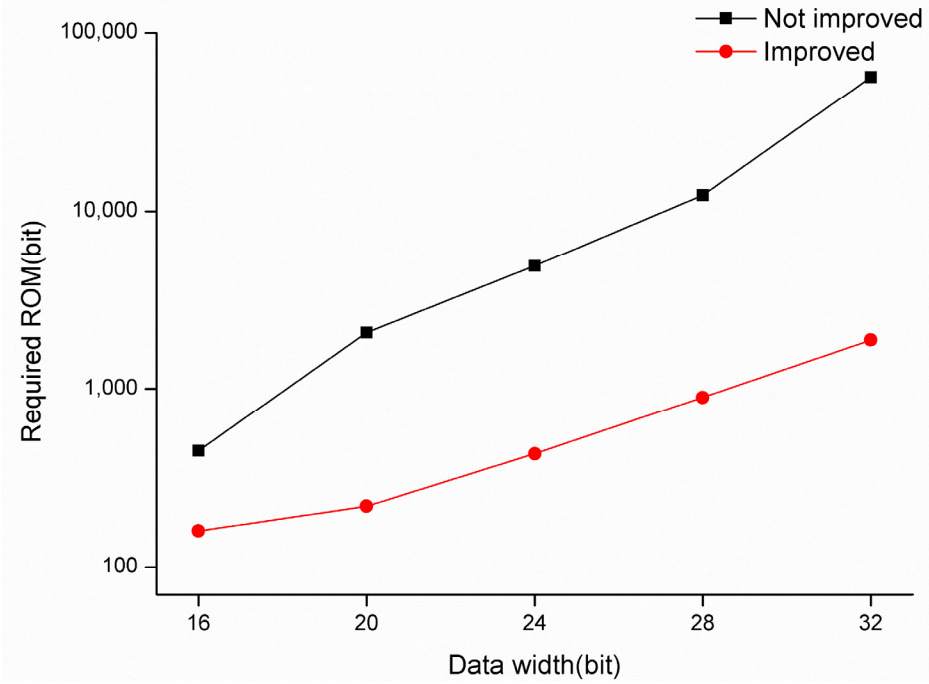

Figure 8. ROM resources required at different bit widths before and after optimization.

6.2. Performance Comparison of CORDIC Algorithms

The test results of the RDP-CORDIC algorithm and other related CORDIC algorithms in terms of latency and resource consumption are shown in Table 6. Apparently, in terms of latency, the R-4 CORDIC, R-8 CORDIC and Mixed-R CORDIC reduced latency but increased ROM consumption and had high hardware complexity. The RDP-CORDIC algorithm had a 70% lower latency compared to the conventional CORDIC algorithm, being parallel to the BBR-CORDIC algorithm. In terms of resource consumption, the RDP-CORDIC algorithm was similar to the CORDIC II algorithm, but the CORDIC II algorithm displayed a larger latency. Although the ROM consumption of the new algorithm was slightly higher than that of the conventional R-2 CORDIC algorithm, the RDP-CORDIC algorithm was clearly more advantageous, exchanging the ultra-small ROM capacity for 70% latency and 40% other resource consumption. In terms of power consumption, the proposed RDP-CORDIC algorithm facilitated an ultra-low power consumption of 28 mW. A comprehensive comparison showed that the RDP-CORDIC algorithm illustrated some advantages over other CORDIC algorithms in terms of latency, resource consumption and power consumption.

Table 6. Performance comparison results of different CORDIC algorithms.

CORDIC Algorithm	Number of Iterations	LUTs + FF	ROM	Power (mW)
R-2 CORDIC [26]	17	2362	0	71
R-4 CORDIC [24]	9	1886	24 × 16	66
R-8 CORDIC [28]	7	1566	36 × 16	65
Mixed-R-scaling-freeCORDIC [29]	8	1771	12 × 16	53
BBR-CORDIC [34]	5	1643	102 × 16	82
CORDIC II [38]	7	1433	24 × 16	32
RDP-CORDIC [proposed]	5	1438	12 × 16	28

6.3. Test of Calculation Error and Calculation Time of Variousfunctions

Figure 9 shows the absolute error curves for the calculation of sine and cosine functions, logarithmic function, sqrt function and hyperbolic sine and hyperbolic cosine functions based on the RDP-CORDIC algorithm at 16-bit data width. The maximum magnitude error

of the sine and cosine functions is clearly less than 3.04×10^{-5}. The reason for the different errors for each input angle is that the error is 0 only when the accumulated value of the angle of directional rotation is equal to the input angle. However, the direction of rotation is not certain for different input angles, which results in the difference between the totalized rotation value and the input angle value. For other functions, the input test data is limited to a different range due to the characteristic limitations of the CORDIC algorithm. As in Figure 9c–f, the input angles are limited to [0.2, 9.5], [0.03, 2], [−1.12, 1.12] and [−1.12, 1.12], respectively. The test results show that the RDP-CORDIC algorithm performs well on a variety of functions, with maximum absolute errors less than 7.7×10^{-4}. Because the computation time of each function is the same through the CORDIC algorithm, the sine and cosine functions are used for the test computation time. The time of the single computation of the sine and cosinefunctions for different CORDIC algorithms are compared in Table 7, and it can be found that the RDP-CORDIC algorithm takes only 60 ns at a system clock of 100 MHz.

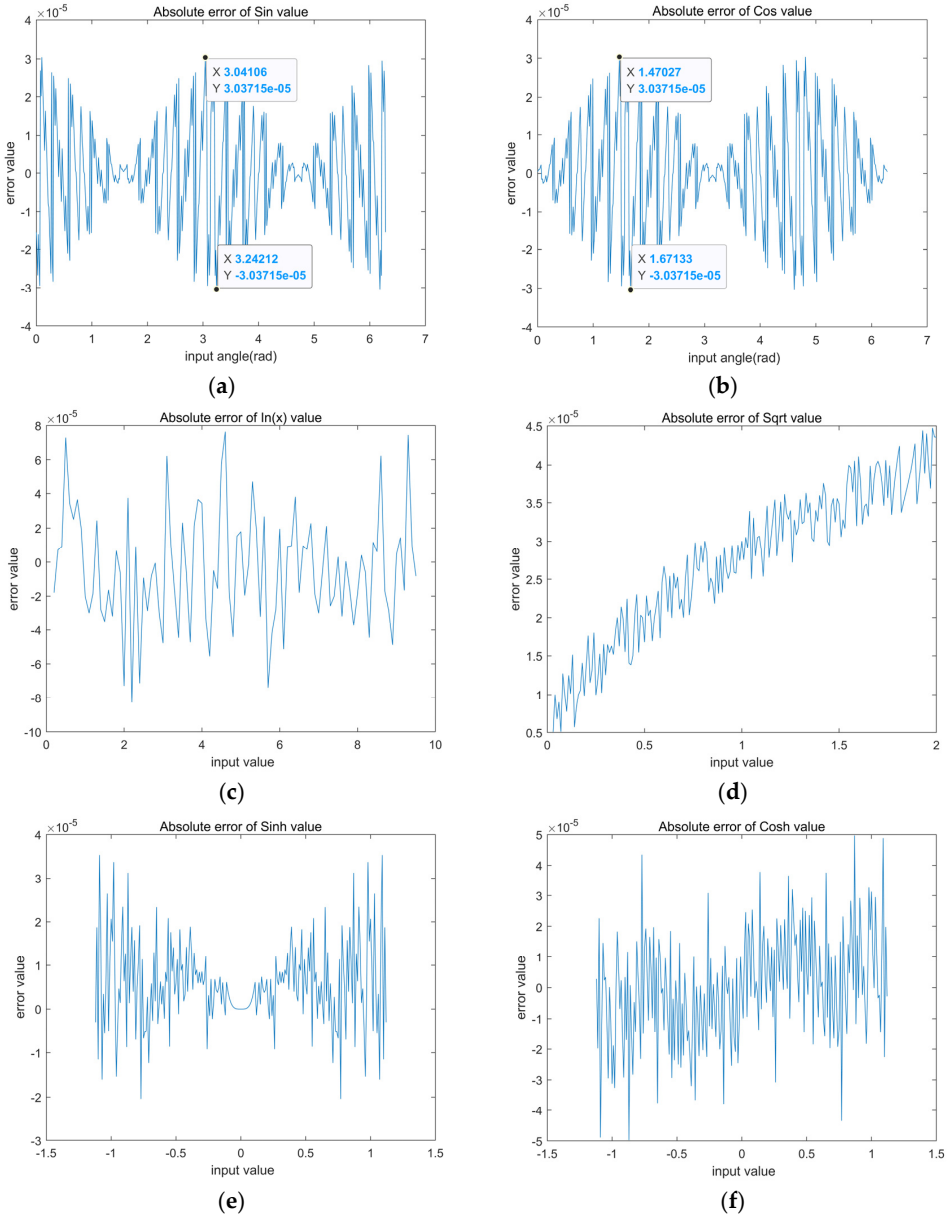

Figure 9. The various functions calculation error base on RDP-CORDIC algorithm: (**a**) Absolute error of sine signal value; (**b**) Absolute error of cosine signal value; (**c**) Absolute error of ln(x) value; (**d**) Absolute error of sqrt value; (**e**) Absolute error of sinh value; (**f**) Absolute error of cosh value.

Table 7. Time consumed for the first calculation of the sine and cosine function.

CORDIC Algorithm	Time (ns)
R-2 CORDIC [26]	190
R-4 CORDIC [24]	100
R-8 CORDIC [28]	80
BBR-CORDIC [34]	60
CORDIC II [38]	90
RDP-CORDIC [proposed]	60

Finally, the maximum absolute error of sine and cosine functions realized with each algorithm under different bit widths was tested, and the test results are shown in Figure 10. The results show that the RDP-CORDIC algorithm maintains the optimal performance in various bit width cases, and the error of the RDP-CORDIC algorithm is much lower than that of the CORDIC II algorithm, with similar resource consumption to that of the RDP-CORDIC algorithm. The reason why the RDP-CORDIC algorithm has higher accuracy compared with other algorithms is that the RDP-CORDIC algorithm calculates the rotation direction by formula once before iteration, while other CORDIC algorithms need to calculate the rotation direction each time, and multiple calculations may cause accuracy degradation.

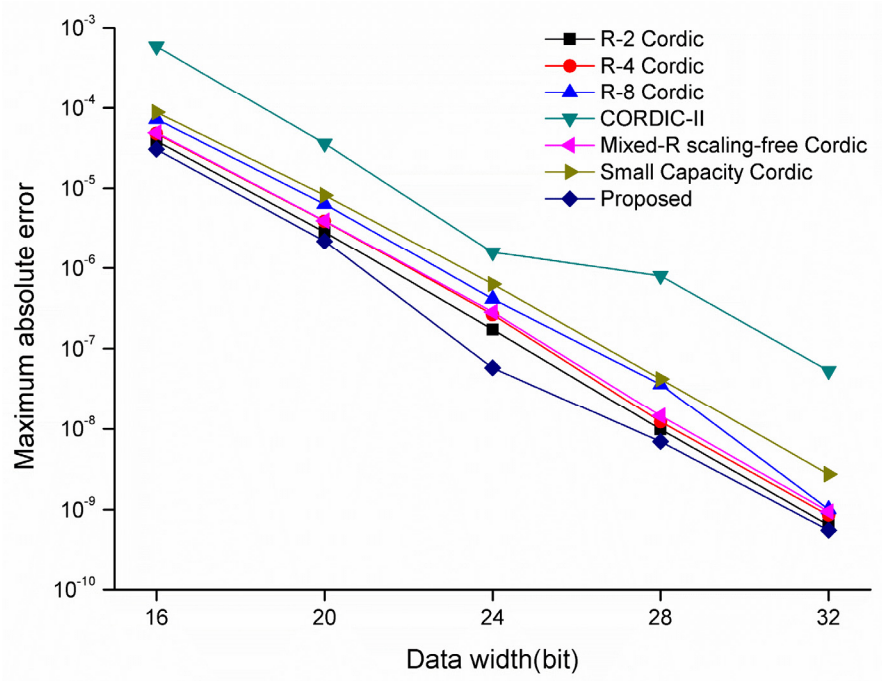

Figure 10. Maximum absolute error of sine and cosine signals calculated by various CORDIC algorithms at different bit widths.

7. Conclusions

Edge computing devices used in smart grid cyber-physical systems require real-time high-speed data processing capabilities with low power requirements. The CORDIC algorithm is widely used as a high-speed real-time numerical computation algorithm in hardware accelerator for edge computing devices. Limited by the excessive number of iterations and resource consumption, conventional CORDIC algorithms are too large to perform well in edge computing of smart grids. In this paper, a RDP-CORDIC algorithm was proposed in attempt to predict all micro-rotation directions in the non-iterative case, and then transform the conventional single-stage iteration structure into three-stage combined iteration and multi-stage combined iteration structure. An accuracy compensation algorithm for the direction prediction constants of the RDP-CORDIC algorithm was also

proposed, which reduces the ROM consumption to 0.33% of the original at 16-bit accuracy. Finally, the hardware design of RDP-CORDIC algorithm was implemented on Xilinx Kintex7 325T series FPGA platform, along with the calculation of sine function, cosine functions, logarithmic function and other functions, accordingly. The test results showed that the RDP-CORDIC algorithm was plainly superior to other CORDIC algorithms in terms of latency, resource consumption and power consumption. The time and the maximum absolute error of the sine and cosine functions computed by the RDP-CORDIC algorithm also had some advantages over other CORDIC algorithms. It was experimentally confirmed that the proposed RDP-CORDIC algorithm was able to reduce resource consumption and increase the computational speed while maintaining the computational accuracy compared to other improved CORDIC algorithms. In the edge computing for signal processing of smart grid cyber-physical systems, the RDP-CORDIC algorithm exhibited its potential to effectively improve the speed of real-time data processing and reduce the power consumption of edge computing. The application of the RDP-CORDIC algorithm will be the focus of our future work: to improve the speed of smart grid topology identification and line loss rate calculation. Further efforts will be made with a particular focus on the signal processing capability of edge computing devices in the smart grid cyber-physical system.

Author Contributions: Conceptualization, M.Q. and T.L.; methodology, T.L.; hardware, T.L.; validation, B.H. and Y.Y.; formal analysis, T.L.; investigation, M.Q.; resources, M.Q.; data curation, B.H.; writing—original draft preparation, T.L.; writing—review and editing, T.L. and Y.Y.; visualization, H.S. and Y.G.; supervision, M.Q. and Y.G.; project administration, M.Q.; funding acquisition, M.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (NSFC), (Grant No. 62261051). This research was also funded by the Sichuan Provincial Science and Technology Department "Research on Key Technologies of Adaptive Communication in Complex Environment", project number 2019YJ0309.

Acknowledgments: We thank H.W., Q.Z., F.K., Z.Z., Y.L., Z.C., W.Z. and C.G. for the completion of this article. The correspondence author TL especially thanks RR for her support to him. We thank the project "Research on Key Technologies of Adaptive Communication in Complex Environment" (Project No. 2019YJ0309) of the Sichuan Provincial Science and Technology Department for its support of this work.

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

References

1. Gilbert, G.M.; Naiman, S.; Kimaro, H.; Bagile, B. A Critical Review of Edge and Fog Computing for Smart Grid Applications. In *IFIP Advances in Information and Communication Technology*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 763–775. [CrossRef]
2. Wang, Z.; Jiang, D.; Wang, F.; Lv, Z.; Nowak, R. A Polymorphic Heterogeneous Security Architecture for Edge-Enabled Smart Grids. *Sustain. Cities Soc.* **2021**, *67*, 102661. [CrossRef]
3. Qin, M.; Gao, Y.; Hou, B.; Wang, H.; Zhou, W.; Yao, Y. Research on Efficient Channel Decoding Algorithm for Memory Channel and Short Packet Transmission in Smart Grid. *Front. Energy Res.* **2022**, *10*, 1014. [CrossRef]
4. Song, C.; Xu, W.; Han, G.; Zeng, P.; Wang, Z.; Yu, S. A Cloud Edge Collaborative Intelligence Method of Insulator String Defect Detection for Power IIoT. *IEEE Internet Things J.* **2021**, *8*, 7510–7520. [CrossRef]
5. Song, C.; Liu, S.; Han, G.; Zeng, P.; Yu, H.; Zheng, Q. Edge Intelligence Based Condition Monitoring of Beam Pumping Units under Heavy Noise in the Industrial Internet of Things for Industry 4.0. *IEEE Internet Things J.* **2022**, 1. [CrossRef]
6. Zhang, Y.; Liang, K.; Zhang, S.; He, Y. Applications of edge computing in PIoT. In Proceedings of the 2017 IEEE Conference on Energy Internet and Energy System Integration (EI2), Beijing, China, 26–28 November 2017; pp. 1–4.
7. Hussain, M.; Alam, M.S.; Beg, M.M. Fog assisted cloud models for smart grid architectures-comparison study and optimal deployment. *arXiv* **2018**, arXiv:1805.09254.
8. Pan, X.; Jiang, A.; Wang, H. Edge-Cloud Computing Application, Architecture, and Challenges in Ubiquitous Power Internet of Things Demand Response. *J. Renew. Sustain. Energy* **2020**, *12*, 062702. [CrossRef]
9. Albayati, A.; Abdullah, N.F.; Abu-Samah, A.; Mutlag, A.H.; Nordin, R. A Serverless Advanced Metering Infrastructure Based on Fog-Edge Computing for a Smart Grid: A Comparison Study for Energy Sector in Iraq. *Energies* **2020**, *13*, 5460. [CrossRef]
10. Kumar, S.; Agarwal, S.; Krishnamoorthy, A.; Vijayarajan, V.; & Kannadasan, R. Improving the response time in smart grid using fog computing. In Proceedings of the 2nd International Conference on Data Engineering and Communication Technology, Pune, India, 15–16 December 2017; Springer: Singapore, 2019; pp. 563–571.

11. Lei, W.; Jiang, Y.; Wen, H.; Xu, A.; Ming, Z.; Hou, W.; Yin, Y. New Features of Automatic Meter Reading System: Based on Edge Computing. In Proceedings of 2019 International Conference on Energy, Power, Environment and Computer Application(ICEPECA 2019)., Wuhan, China, 20–21 January 2019; pp. 364–368.
12. Yu, D.; Ma, Z.; Wang, R. Efficient Smart Grid Load Balancing via Fog and Cloud Computing. *Math. Probl. Eng.* **2022**, *2022*, 3151249. [CrossRef]
13. Yang, K.; Jiang, L.; Low, S.; Liu, S. Privacy-Preserving Energy Scheduling for Smart Grid with Renewables. *IEEE Access* **2020**, *8*, 132320–132329. [CrossRef]
14. Diamantoulakis, P.D.; Bouzinis, P.S.; Sarigannidis, P.G.; Ding, Z.; Karagiannidis, G.K. Optimal Design and Orchestration of Mobile Edge Computing with Energy Awareness. *IEEE Trans. Sustain. Comput.* **2022**, *7*, 456–470. [CrossRef]
15. Song, C.; Han, G.; Zeng, P. Cloud Computing Based Demand Response Management Using Deep Reinforcement Learning. *IEEE Trans. Cloud Comput.* **2021**, *10*, 72–81. [CrossRef]
16. Zhu, Z.; Zhang, J.; Zhao, J.; Cao, J.; Zhao, D.; Jia, G.; Meng, Q. A Hardware and Software Task-Scheduling Framework Based on CPU+FPGA Heterogeneous Architecture in Edge Computing. *IEEE Access* **2019**, *7*, 148975–148988. [CrossRef]
17. Amarasinghe, G.; de Assunção, M.D.; Harwood, A.; Karunasekera, S. A Data Stream Processing Optimisation Framework for Edge Computing Applications. In Proceedings of the 2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC), Singapore, 29–31 May 2018; pp. 91–98. [CrossRef]
18. Yunzhou, Z.; Mo, Z.; Haoqi, L.; Gang, Z. Innovative Architecture of Single Chip Edge Device Based on Virtualization Technology. *Pervasive Mob. Comput.* **2019**, *52*, 100–112. [CrossRef]
19. Kumar, P.A. FPGA Implementation of the Trigonometric Functions Using the CORDIC Algorithm. In Proceedings of the 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 15–16 March 2019; pp. 894–900. [CrossRef]
20. Fu, W.; Xia, J.; Lin, X.; Liu, M.; Wang, M. Low-Latency Hardware Implementation of High-Precision Hyperbolic Functions Sinhx and Coshx Based on Improved CORDIC Algorithm. *Electronics* **2021**, *10*, 2533. [CrossRef]
21. Mahdavi, H.; Timarchi, S. Area–Time–Power Efficient FFT Architectures Based on Binary-Signed-Digit CORDIC. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2019**, *66*, 3874–3881. [CrossRef]
22. Younes, H.; Ibrahim, A.; Rizk, M.; Valle, M. Efficient FPGA Implementation of Approximate Singular Value Decomposition based on Shallow Neural Networks. In Proceedings of the 2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS), Washington DC, USA, 6–9 June 2021; pp. 1–4. [CrossRef]
23. Nguyen, H.-T.; Nguyen, X.-T.; Pham, C.-K. A Low-Latency Parallel Pipeline CORDIC. *IEICE Trans. Electron.* **2017**, *E100.C*, 391–398. [CrossRef]
24. Villalba, J.; Zapata, E.L.; Antelo, E.; Bruguera, J.D. Radix-4 vectoring cordic algorithm and architectures. *J. VLSI Signal Process. Syst. Signal Image Video Technol.* **1998**, *19*, 127–147. [CrossRef]
25. Antelo, E.; Bruguera, J.D.; Zapata, E.L. Unified Mixed Radix 2-4 Redundant CORDIC Processor. *IEEE Trans. Comput.* **1996**, *45*, 1068–1073. [CrossRef]
26. Bruguera, J.D.; Antelo, E.; Zapata, E.L. Design of a Pipelined Radix 4 CORDIC Processor. *Parallel Comput.* **1993**, *19*, 729–744. [CrossRef]
27. Antelo, E.; Villalba, J.; Bruguera, J.D.; Zapata, E.L. High Performance Rotation Architectures Based on the Radix-4 CORDIC Algorithm. *IEEE Trans. Comput.* **1997**, *46*, 855–870. [CrossRef]
28. Parmar, Y.; Sridharan, K. Precomputation-Based Radix-4 CORDIC for Approximate Rotations and Hough Transform. *IET Circuits Devices Syst.* **2018**, *12*, 413–423. [CrossRef]
29. Tang, W.; Xu, F. A Noniterative Radix-8 CORDIC Algorithm with Low Latency and High Efficiency. *Electronics* **2020**, *9*, 1521. [CrossRef]
30. Changela, A.; Zaveri, M.; Verma, D. Mixed-Radix, Virtually Scaling-Free CORDIC Algorithm Based Rotator for DSP Applications. *Integration* **2021**, *78*, 70–83. [CrossRef]
31. Jaime, F.J.; Sanchez, M.A.; Hormigo, J.; Villalba, J.; Zapata, E.L. Enhanced Scaling-Free CORDIC. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2010**, *57*, 1654–1662. [CrossRef]
32. Moroz, L.; Taras, M.; Herasym, M. Improved scaling-free CORDIC algorithm. In Proceedings of the 2013 11th East-West Design and Test Symposium (EWDTS), Rostov-on-Don, Russia, 27 September 2013; IEEE Computer Society: Washington, DC, USA, 2013.
33. Shukla, R.; Ray, K.C. Low Latency Hybrid CORDIC Algorithm. *IEEE Trans. Comput.* **2014**, *63*, 3066–3078. [CrossRef]
34. Yao, Y.; Feng, Z. BBR-Based Iteration-Free CORDIC Algorithm. *J. Circuits Syst. Comput.* **2018**, *27*, 1850076. [CrossRef]
35. Zhang, Y.Y.; Liu, J.R.; Wang, Z.Y.; Mo, J.J.; Yu, F.X. Implementation of direct digital frequency synthesizer based on three-step rotation coordinate rotation digital computer algorithm. *J. Zhejiang Univ. Eng. Sci.* **2019**, *53*, 2034–2040.
36. Khurshid, B.; Khan, J.J. An Efficient Fixed-Point Multiplier Based on CORDIC Algorithm. *J. Circuits Syst. Comput.* **2020**, *30*, 2150080. [CrossRef]
37. Kumar, A.; Kumar, A.; Singh Tomar, G. Hardware Chip Performance of CORDIC Based OFDM Transceiver for Wireless Communication. *Comput. Syst. Sci. Eng.* **2022**, *40*, 645–659. [CrossRef]
38. Garrido, M.; Kallstrom, P.; Kumm, M.; Gustafsson, O. CORDIC II: A New Improved CORDIC Algorithm. *IEEE Trans. Circuits Syst. II Express Briefs* **2016**, *63*, 186–190. [CrossRef]