HGH-CORDIC: A High-Radix Generalized Hyperbolic COordinate Rotation Digital Computer

Hui Chen[1], Lianghua Quan[2], Weiqiang Liu[1]
[1]*College of Integrated Circuits, Nanjing University of Aeronautics and Astronautics, China*
[2]*School of Electronic Science and Engineering, Nanjing University, China*

***Abstract*—In this paper, we propose a high-radix generalized hyperbolic coordinate rotation digital computer (HGH-CORDIC), which not only can compute the logarithmic and exponential functions with any fixed base, but also can reduce the number of iterations compared with the traditional CORDIC. First, we propose the general iteration formulas for HGH-CORDIC. Then we demonstrate its important convergence property and selection criteria, and illustrate them with the most commonly used examples. Through software simulation, we further prove the correctness of the theory. Finally, we analyze how to implement it efficiently in hardware. Compared with the state-of-the-art work, HGH-CORDIC can reduce the number of iterations by more than 50% with the same accuracy.**

***Index Terms*—High-radix, hyperbolic CORDIC, generalized, logarithmic, exponential, hardware implementation**

I. Introduction

Since Volder invented CORDIC (COordinate Rotation Digital Computer) [1], various research and applications about CORDIC have been pouring in. Over the past sixty years, many variants and hardware implementations of CORDIC have appeared to expand its application form. Later, it was first extended to other coordinate systems by Walther, who unified the CORDIC algorithm [2]. In this way, CORDIC has three coordinate systems: circular, linear, and hyperbolic. Coupled with the existence of rotation and vector modes, they can be used to compute many basic mathematical operations, such as trigonometric [3], [4], logarithmic, and exponential functions [5] besides multiplication and division.

The initial application was to build real-time navigation computers for aircraft and was primarily focused on implementing trigonometric functions [6]. Later, its applications are also extended to eigenvalue calculation [7], singular value decomposition (SVD) [8], robotics control [9], and artificial intelligence [10]. The reason why CORDIC is widely used is that it can continuously iterate to obtain a more accurate calculation result with only shift-and-add operations. However, such as the lookup table (LUT) method or the linear approximation method, also needs additional storage memory (or registers) or multipliers with large area cost, and they are unsuitable for obtaining high calculation precision.

The main disadvantage of CORDIC is that it requires more iterations to achieve higher accuracy, which results in long computation latency. To improve this shortcoming, many scholars have done research from different levels. On the one hand, they optimize the algorithm or hardware architecture of the traditional CORDIC (radix-2), and on the other hand, very-high radix CORDIC is proposed to reduce the number of iterations. The former's works include a semi-iterative 2D CORDIC algorithm [11] and rotation angle precoding method [12]. The latter's works include very-high radix CORDIC algorithm [13], [14]. In the actual hardware implementation, because of the highest complexity of the radix-8 CORDIC algorithm, its hardware area consumption is larger than radix-4 or radix-2 CORDIC. Although radix-8 CORDIC can further reduce the number of iterations under the same precision condition, it has lower cost performance than the other two CORDIC. Among the three CORDIC algorithms, radix-4 CORDIC is often the best choice. It can not only reduce the number of iterations than radix-2 CORDIC but also minimize the hardware cost through some approximation schemes. This is the reason why this paper will focus on radix-4 CORDIC.

Other aspects of CORDIC that have been studied include scalability. Because the convergence range of the CORDIC algorithm is limited, Hu Xiaobo *et al.* earlier proposed an extension method for CORDIC in various modes and coordinate systems [15]. This method also paves the way for further research to implement more complex mathematical functions, such as calculating Nth root [16]. Moreover, the traditional hyperbolic CORDIC can only be used to calculate exponential and logarithmic functions with base-e (i.e., e^x and lnx), which has certain application limitations. To solve this problem, Luo *et al.* proposed a generalized hyperbolic CORDIC (GH-CORDIC) algorithm, which can realize logarithmic and exponential functions of arbitrary base (i.e., b^x and $log_b x$, b is an integer) [17]. The proposed extension method also provides optimization means for implementing many complex functions, such as X^Y-like computations [18] and base-2 softmax function [19]. In particular, base-2 hyperbolic CORDIC is more promising in floating-point operations, such as floating-point logarithmic function [20] and Nth root [21]. Unfortunately, the previous extension theory and related good works are all based on the radix-2 CORDIC algorithm. The disadvantage of long calculation latency makes it unacceptable in applications with strict computing speed requirements. This is the motivation of this paper to put forward the theory of high-radix GH-CORDIC, and why the analysis especially

This work was supported by the National Nature Science Foundation of China under Grant No. 92364201. Corresponding authors: Hui Chen and Weiqiang Liu (e-mail: huichen@nuaa.edu.cn; liuweiqiang@nuaa.edu.cn).

focuses on the base-2 hyperbolic CORDIC.

To sum up, GH-CORDIC has a wide range of application values, and utilizing high-radix CORDIC theory is an important means to solve the long computation latency problem of traditional GH-CORDIC. Therefore, we propose the theory of high-radix GH-CORDIC (HGH-CORDIC) to fill the gap, and especially present the most commonly used hyperbolic (with base-2) CORDIC (with radix-4) in detail. Of course, the proposed theory can also support higher-radix GH-CORDIC, such as radix-8 and radix-16 GH-CORDIC.

The rest of this work is structured as follows. Section II introduces the theory of radix-2 GH-CORDIC and shows how to use it to calculate arbitrary fixed-base logarithms and exponentials. Section III develops the theory of HGH-CORDIC, which can quickly compute the base-b (b is a variable) logarithmic and exponential functions with lower iterations than radix-2 GH-CORDIC. In Section IV, some software simulation experiments are shown. Then, we make a hardware implementation analysis of HGH-CORDIC in Section V. Finally, we summarize the work in Section VI.

II. Theory of the Radix-2 Generalized Hyperbolic CORDIC and Its Application

In this section, we first introduce the theory of the radix-2 GH-CORDIC. Then, we show how to realize the logarithmic and exponential functions with an arbitrary base using it.

A. Working Principle of Radix-2 GH-CORDIC

Like the traditional hyperbolic CORDIC, radix-2 GH-CORIC has two modes: vector and rotation. The essence of both modes is coordinate rotation, as shown in Fig. 1.

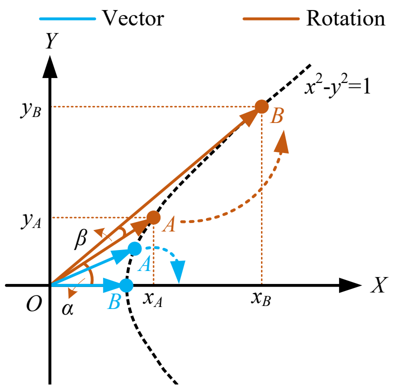


Fig. 1. Coordinate rotation principle of radix-2 GH-CORDIC.

The curve satisfies the equation $x^2 - y^2 = 1$. There are two color-coded points A and B. When A is rotated to B on the X-axis, it is considered a vector mode (GHV-CORDIC) (blue). When A rotates at an angle β to B, it is regarded as a rotation mode (GHR-CORDIC) (orange). According to the generalized hyperbolic definition, the coordinate of A is defined as

$$\begin{bmatrix} x_A \\ y_A \end{bmatrix} = \begin{bmatrix} \cosh_b \alpha \\ \sinh_b \alpha \end{bmatrix} = \begin{bmatrix} \frac{b^{\alpha}+b^{-\alpha}}{2} \\ \frac{b^{\alpha}-b^{-\alpha}}{2} \end{bmatrix}, \tag{1}$$

where $b \in \mathbb{N}^*$ but $b \neq 1$.

Taking the rotation mode as an example, when point A rotates to point B, the coordinates of point B will be

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} \cosh_b(\alpha+\beta) \\ \sinh_b(\alpha+\beta) \end{bmatrix}, \tag{2}$$

where β is the actual rotation angle. According to the mathematical identity transformation, (2) is unrolled to

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} x_A \cosh_b \beta + y_A \sinh_b \beta \\ y_A \cosh_b \beta + x_A \sinh_b \beta \end{bmatrix}. \tag{3}$$

If we extract $\cosh_b \beta$ from (3), it will be

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{1-\tanh_b^2 \beta}}(x_A + y_A \tanh_b \beta) \\ \frac{1}{\sqrt{1-\tanh_b^2 \beta}}(y_A + x_A \tanh_b \beta) \end{bmatrix}. \tag{4}$$

The basic working principle of GH-CORDIC is to decompose β into the combination of some small angle β_i with direction,

$$\beta_i = d_i \tanh_b^{-1}(2^{-i}),\ d_i \in \{-1, +1\},\ i = 1, 2, 3, \cdots. \tag{5}$$

Since $\tanh_b(\beta_i) = d_i 2^{-i}$, the complicated multiplication operation with y_A or x_A in (4) will become the simple shift operation in hardware. Equation (4) is now rewritten as

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \frac{1}{\sqrt{1-2^{-2i}}} \begin{bmatrix} 1 & d_i 2^{-i} \\ d_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \end{bmatrix}. \tag{6}$$

In the actual use of GH-CORDIC, the number of total iterations is usually determined according to the calculation accuracy requirements. That is, we do not use complex logic in the hardware to calculate the scaling factor $\Pi_{i=1}^{\infty} = \sqrt{1-2^{-2i}}$ (denoted as K_{gh}). Instead, K_{gh} will be computed as a constant ahead of time, and a constant multiplier may be used when the final result needs to be scaled by K_{gh}^{-1}.

Based on the aforementioned working principle, we can easily understand the GH-CORDIC iteration formulas [17]:

$$\begin{aligned} x_{i+1} &= x_i + d_i(2^{-i} y_i), \\ y_{i+1} &= y_i + d_i(2^{-i} x_i), \\ z_{i+1} &= z_i - d_i \tanh_b^{-1}(2^{-i}), \end{aligned} \tag{7}$$

where $\tanh_b^{-1}(2^{-i}) = \tanh^{-1}(2^{-i}) / \ln(b)$ and i starts from 1. The determined value of d_i in each iteration is related to the working mode of GH-CORDIC. In the GHR-CORDIC algorithm, d_i is determined by z_i. However, d_i is determined by x_i and y_i in the GHV-CORDIC algorithm. That is,

$$d_i = \begin{cases} \text{sign}(z_i), & \text{(for GHR-CORDIC)} \\ -\text{sign}(y_i). & \text{(for GHV-CORDIC)} \end{cases} \tag{8}$$

Through a certain number of iterations n, the convergent functions of GH-CORDIC are shown in Table I under the initial inputs (x_0, y_0, z_0).

TABLE I
CONVERGENT FUNCTIONS OF GH-CORDIC

Type	Outputs
GHR-CORDIC	$x_n = K_{gh}(x_0 \cdot \cosh_b(z_0) - y_0 \cdot \sinh_b(z_0))$
	$y_n = K_{gh}(y_0 \cdot \cosh_b(z_0) + x_0 \cdot \sinh_b(z_0))$
	$z_n \to 0$
GHV-CORDIC	$x_n = K_{gh}\sqrt{(x_0)^2 - (y_0)^2}$
	$y_n \to 0$
	$z_n = z_0 + \tanh_b^{-1}(y_0/x_0)$

B. Approach of Using GH-CORDIC

In order to calculate logarithmic and exponential results using GH-CORDIC, we need to place specific constraints on the initial inputs. From the convergence functions, it can be observed that the exponential function can be realized using GHR-CORDIC and the logarithmic function can be realized using GHV-CORDIC.

If we initialize the inputs of GHR-CORDIC as

$$x_0 = y_0 = 1/K_{gh},\ z_0 = Q, \tag{9}$$

when z_n approaches 0 through iterations, y_n approximates to

$$y_n = \cosh_b(Q) + \sinh_b(Q) = b^Q, \tag{10}$$

which happens to be an approximate exponential result.

Similarly, when the initialization of GHV-CORDIC inputs is given as

$$x_0 = Q + 1,\ y_0 = Q - 1,\ z_0 = 0, \tag{11}$$

when y_n approaches 0 through iterations, z_n converges to

$$z_n = \tanh_b(\frac{Q-1}{Q+1}) = \tanh(\frac{Q-1}{Q+1})/\ln b = \frac{1}{2}\log_b Q, \tag{12}$$

which is an approximate logarithmic result by shifting one bit to the left in hardware ($2 \cdot z_n$).

At this point, the derivation using GH-CORDIC to calculate b^Q and $log_b Q$ for any base-b is complete. Compared with the most primitive hyperbolic CORDIC [2], it no longer requires a constant multiplier ($\frac{1}{lnb} \cdot lnQ$) or divider ($\frac{lnQ}{lnb}$) to achieve arbitrary logarithms with base-b. However, it still works on the basic principle of radix-2 CORDIC, so GH-CORDIC achieves convergent results in a slow iterative process. So we propose a definition of GH-CORDIC with higher radix in the following.

III. DEFINITION OF HIGH-RADIX GH-CORDIC AND ITS VARIOUS PROPERTIES

Although the traditional hyperbolic CORDIC of high-radix was proposed in 1993 [22], it can only accelerate the calculation of natural logarithms and exponents (lnx and e^x), but can not realize arbitrary $log_b x$ and b^x functions. Therefore, we propose a method called HGH-CORDIC to solve it.

We first give the definitions of HGH-CORDIC. Then, we analyze their convergence range and make an example analysis. Next, we present the important selection criteria of HGH-CORDIC, especially for the demonstration of radix-4 GH-CORDIC (most cost-effective in hardware). Finally, we provide the general number of iterations of HGH-CORDIC.

A. The Proposed High-radix GH-CORDIC

When considering a higher radix r for GH-CORDIC, more results need to be calculated for each iteration to reduce the total number of iterations. Usually, the radix r is a power of two, and the rotation angle β is decomposed into a series of elementary angles, whose values are $\tanh_b(\beta_i) = d_i r^{-i}$. The coefficients d_i can take values from the minimally redundant integer set $[-r/2, r/2]$. In other words, the direction and rotation degree of each iteration is different. The degree may be large or small, or even the same. Especially for radix higher than 4, d_i is not integer powers of two and the hardware complexity of each iteration increases significantly [22].

The high-radix iteration formulas for the two modes of GH-CORDIC are different. We need to talk about them separately.

First, we propose the radix-r GH-CORDIC equations of rotation mode (HGHR-CORDIC) based on (7):

$$\begin{aligned} x_{i+1} &= x_i + d_i(r^{-i} y_i), \\ y_{i+1} &= y_i + d_i(r^{-i} x_i), \\ w_{i+1} &= r(w_i - r^i \tanh_b^{-1}(d_i r^{-i})). \end{aligned} \tag{13}$$

In order to have the non-zero digits always in the most significant positions to select d_i, here we use a scaled recurrence $w_i = r^i z_i$ to decompose the rotation angle instead of the more conventional recurrence $z_{i+1} = z_i - \tanh_b^{-1}(d_i r^{-i})$, which is a standard practice in other digit recurrences, such as division and square root [23]. The value of d_i is determined by a selection criterion, which must assure the convergence of the proposed HGHR-CORDIC algorithm. We need to bound the new variable w_i by upper limit ($U_i[q]$) and lower limit ($L_i[q]$)), which follows a similar method to the one proposed for the radix-r SRT division algorithm [24]. $U_i[q]$ and $L_i[q]$ are monotonic functions, i.e. $U_i[q] \le U_i[q+1]$ and $L_i[q] \le L_i[q+1]$. The bounded limits can be defined as follows:

$$\begin{aligned} U_i[q] &= r^i[\tanh_b^{-1}(qr^{-i}) + \frac{p}{r-1}\tanh_b^{-1}(r^{-i})], \\ L_i[q] &= r^i[\tanh_b^{-1}(qr^{-i}) - \frac{p}{r-1}\tanh_b^{-1}(r^{-i})], \end{aligned} \tag{14}$$

where $q \in \{\pm\frac{r}{2}, \pm(\frac{r}{2}-1), \cdots, 0\}$. In each iteration, there must be a selection function, $d_i = q$, for which w_i is limited in an interval $L_i[q] \le w_i \le U_i[q]$. To maximize the overlap between the intervals used for choosing different d_i values and minimize redundancy, p is often selected as $r/2$.

Through multiple iterations based on the desired precision, we can obtain the same convergent functions as radix-2 GH-CORDIC. However, one difference is that the scaling factor K_{gh} is no longer a predeterminable constant. For high-radix GH-CORDIC, it is reexpressed as K_{hgh} and should be

$$K_{hgh} = \prod_{i \ge 1} \sqrt{1 - d_i^2 r^{-2i}}. \tag{15}$$

Because the selected value of d_i for each iteration is not fixed, K_{gh} has to be calculated and compensated to preserve the norm of the vector. Later we will provide efficient solutions to get its results.

Second, for the radix-r GH-CORDIC equations of vector mode (HGHV-CORDIC), we similarly define a new scaled recurrence $w_i = r^i y_i$. After the conversion, the equations in (7) will be changed to

$$\begin{aligned} x_{i+1} &= x_i - d_i(r^{-2i}w_i), \\ w_{i+1} &= r(w_i - d_i x_i), \\ z_{i+1} &= z_i + \tanh_b^{-1}(d_i r^{-i}). \end{aligned} \tag{16}$$

To make sure the algorithm is still convergent, w_i also must be bounded between the lower limit function and upper limit function. Similarly, we define them as

$$U_i[a] = (a + \frac{p}{r-1})x_i, \;\; L_i[a] = (a - \frac{p}{r-1})x_i. \tag{17}$$

$d_i = a$ is selected by the criteria given in an interval $L_i[a] \leq w_i \leq U_i[a]$ to guarantee convergence of HGHV-CORDIC. Similarly, p is often selected as $r/2$.

Based on (16) and suitable selection criteria for d_i, we will get the same convergence result as the radix-2 GH-CORDIC. But most importantly, HGHV-CORDIC can quickly calculate the logarithmic results.

B. Convergence Range of HGH-CORDIC

According to the theorem of CORDIC [15], the convergence range of HGH-CORDIC depends on the maximum sum of the rotation angles β_{max}, which can be defined as

$$\begin{aligned} \beta_{max} &= \sum_{i=1}^{\infty}|\beta_i| = \sum_{i=1}^{\infty}|\tanh_b^{-1}(d_i r^{-i})| \\ &= \sum_{i=1}^{\infty}\frac{\tanh^{-1}(d_i r^{-i})}{|\ln b|} = \frac{1}{|\ln b|}\sum_{i=1}^{\infty}\tanh^{-1}(d_i r^{-i}). \end{aligned} \tag{18}$$

It is worth mentioning that the high-radix (such as radix-4) GH-CORDIC should not require any repeated iteration for convergence.

Therefore, the convergence range of HGHR-CORDIC can be given by

$$|z_0| \leq \beta_{max} \leq \frac{1}{|\ln b|}\sum_{i=1}^{\infty}\tanh^{-1}(\frac{r^{1-i}}{2}), \tag{19}$$

because the maximum value of d_i is $\frac{r}{2}$. Similarly, we can get the convergence range of HGHV-CORDIC as follows:

$$\begin{aligned} |\tanh_b^{-1}(\frac{y_0}{x_0})| &= |\frac{\tanh^{-1}(\frac{y_0}{x_0})}{\ln b}| \\ &\leq \beta_{max} = \frac{1}{|\ln b|}\sum_{i=1}^{\infty}\tanh^{-1}(d_i r^{-i}). \end{aligned} \tag{20}$$

So

$$|\tanh^{-1}(\frac{y_0}{x_0})| \leq \sum_{i=1}^{\infty}\tanh^{-1}(d_i r^{-i}) \leq \sum_{i=1}^{\infty}\tanh^{-1}(\frac{r^{1-i}}{2}). \tag{21}$$

When we use HGHV-CORDIC to compute $log_b Q$, the input Q must meet the following condition.

$$|\frac{Q-1}{Q+1}| \leq \tanh(\sum_{i=1}^{\infty}\tanh^{-1}(\frac{r^{1-i}}{2})). \tag{22}$$

For example, when the radix of HGH-CORDIC is 4 and the base of the generalized hyperbolic function is 2.

$$|\frac{Q-1}{Q+1}| \leq \tanh(\sum_{i=1}^{\infty}\tanh^{-1}(\frac{4^{1-i}}{2})) = 0.6148. \tag{23}$$

So the range of Q using HGHV-CORDIC can be derived as

$$Q \in [0.2386, \; 4.1924]. \tag{24}$$

If we use HGHR-CORDIC, the input range of z_0 (Q) will be

$$|z_0| = |Q| \leq \frac{1}{|\ln 2|}\sum_{i=1}^{\infty}\tanh^{-1}(\frac{4^{1-i}}{2}) = 1.0339. \tag{25}$$

C. Selection Criteria in HGH-CORDIC

From the previous subsection, it can be seen that the selection function of d_i is crucial, which determines whether HGH-CORDIC can converge. In addition, it also determines the cost of hardware implementation.

Let's start with HGHV-CORDIC. For the sake of derivation and exposition, we take $r = 4$ and $b = 2$ as an example (denoted as H4G2HV-CORDIC). From (17), we get a clear equation to guarantee the convergence of H4G2HV-CORDIC.

$$U_i[a] = (a + \frac{2}{3})x_i, \; L_j[a] = (a - \frac{2}{3})x_i. \tag{26}$$

Then, we can use (26) to derive the criteria intervals to select d_i. The value of the variable w_i should be bounded within this interval in each iteration to ensure convergence. For example, to choose $d_i = 2$, w_i must fall within the interval $[\frac{4}{3}x_i, \; \frac{8}{3}x_i]$. Analogously, to choose $d_i = 1$, w_i must fall within the interval $[\frac{1}{3}x_i, \; \frac{5}{3}x_i]$. They have an overlap interval, which is $[\frac{4}{3}x_i, \; \frac{5}{3}x_i]$. We can select any value from the overlap interval. In other words, many judgment criteria for w_j can be used. However, adopting arbitrary functions can easily lead to complex and impractical implementations. Therefore, we try to design hardware-friendly judgment criteria to produce a simple implementation architecture.

According to the overlap principle, the relationship among selection value, judgment criteria, and overlap interval is presented in Table II. To facilitate the hardware implementation, $\{\frac{3}{2}x_i, \; \frac{1}{2}x_i, \; -\frac{1}{2}x_i, \; -\frac{3}{2}x_i\}$ are selected as the judgment boundary from the overlap interval.

TABLE II
H4G2HV-CORDIC: RELATIONSHIP AMONG SELECTION VALUE, JUDGMENT CRITERIA AND OVERLAP INTERVAL

Selection Value	Judgment Criteria	Overlapping Interval
2	$w_i \geq \frac{3}{2}x_i$	$[\frac{4}{3}x_i, \; \frac{5}{3}x_i]$
1	$\frac{1}{2}x_i \leq w_i < \frac{3}{2}x_i$	$[\frac{1}{3}x_i, \; \frac{2}{3}x_i]$
0	$-\frac{1}{2}x_i \leq w_i < \frac{1}{2}x_i$	$[-\frac{2}{3}x_i, \; -\frac{1}{3}x_i]$
-1	$-\frac{3}{2}x_i \leq w_i < -\frac{1}{2}x_i$	$[-\frac{5}{3}x_i, \; -\frac{4}{3}x_i]$
-2	$w_i < -\frac{3}{2}x_i$	/

Similar to the high-radix CORDIC of the circular system, we can also seek an iteration i such that the judgment boundaries belong to the common overlap region. Although the object of our study is the vector mode of generalized hyperbolic CORDIC, its high-radix principle is the same as the vector mode of traditional circular CORDIC. So we're not going to do detailed reasoning. So we skip the details of its derivation here. According to the theoretical proof in [25], we can directly get the following inequality:

$$\begin{aligned} L_\infty[2] &\le D_i(2) \le U_i[1] \quad (i \ge 2), \\ L_\infty[1] &\le D_i(1) \le U_i[0] \quad (i \ge 1), \\ L_\infty[0] &\le D_i(-1) \le U_i[-1] \quad (i \ge 1), \\ L_\infty[-1] &\le D_i(-2) \le U_i[-2] \quad (i \ge 2), \end{aligned} \tag{27}$$

where D_i is the selected judgment boundary from the overlap interval. Finally, we can get the simplified judgment boundary for different iterations in (28), which means that we only need to calculate the judgment boundaries in the first iteration for $D_i[\pm 1]$ and first two iterations for $D_i[\pm 2]$. From this iteration on, the calculated values $[D_2(2),\, D_1(1),\, D_1(-1),\, D_2(-2)]$ are valid for the remaining iterations.

$$\begin{aligned} D_i(\pm 1) &= \pm\frac{1}{2}\cdot x_1 \quad (\text{if } i \ge 1), \\ D_i(\pm 2) &= \begin{cases} \pm\frac{3}{2}\cdot x_1 & (\text{if } i = 1), \\ \pm\frac{3}{2}\cdot x_2 & (\text{if } i \ge 2). \end{cases} \end{aligned} \tag{28}$$

Next, we discuss the selection criteria for H4G2HR-CORDIC ($r = 4$ and $b = 2$). Other cases are similar. To be able to obtain a selection function independent of the iteration index, the limits of the d_i selection intervals must be the same in each iteration. Then, we can identify a common overlap region for all iterations, which determines the limits of the selection intervals independent of i. It is possible for all micro-rotations with $i \ge 1$.

First, we discuss the case for $i \ge 1$ and find the overlap between the selection intervals corresponding to coefficients $\{0, \pm 1, \pm 2\}$. If the above situation is true, then we can get the following relationship:

$$d_i = q \text{ if } L[q] \le w_i \le U[q], \tag{29}$$

where $L[q] = max\{L_i[q]\}$ and $U[q] = min\{U_i[q]\}$. Before proving that (29) is true, we need to complete another theoretical proof to guarantee it. It should be emphasized that the theory is not the same as the high-radix CORDIC in the circular system [26].

THEOREM: Selecting d_i according to the criterion given in (14), if we define $P_i[q] = 4^i \tanh_2^{-1}(q4^{-i})$, we have that

$$\begin{aligned} &\begin{cases} U_1[q] \ge U_i[q] \ge (q + \frac{2}{3})/ln2 & \Leftarrow q \ge 0, \\ L_1[q] \ge L_i[q] \ge (q - \frac{2}{3})/ln2 & \Leftarrow q > 0, \end{cases} \\ &\begin{cases} U_\infty[q] \ge U_i[q] \ge log_2[(\frac{5}{3})^{\frac{4}{3}}(\frac{4+q}{4-q})^2] & \Leftarrow q < 0, \\ L_\infty[q] \ge L_i[q] \ge log_2[(\frac{4+q}{4-q})^2/(\frac{5}{3})^{\frac{4}{3}}] & \Leftarrow q \le 0. \end{cases} \end{aligned} \tag{30}$$

PROOF: Let's take $U_i[q]$ as an example, and $L_i[q]$ goes through the same proof process. We define 4^{-i} as a new variable k and the equation $U_i[q]$ as a new function $f(k)$ of k, where $0 < k \le \frac{1}{4}$. So

$$\begin{aligned} f(k) &= \frac{1}{k}[\tanh_2^{-1}(qk) + \frac{2}{3}\tanh_2^{-1}(k)] \\ &= \frac{1}{\ln 2 \cdot k}[\tanh^{-1}(qk) + \frac{2}{3}\tanh^{-1}(k)]. \end{aligned} \tag{31}$$

Taking the derivative of the function $f(k)$, we can obtain

$$\begin{aligned} f^{'}(k) &= \frac{1}{k\ln 2}[\frac{q}{1-(qk)^2} - \frac{1}{2k}\ln(\frac{1+qk}{1-qk})] + \\ &\frac{2}{3}\cdot\frac{1}{k\ln 2}[\frac{1}{1-k^2} - \frac{1}{2k}\ln(\frac{1+k}{1-k})] \\ &= \frac{1}{k^2\ln 2}[g(qk) + \frac{2}{3}g(k)], \end{aligned} \tag{32}$$

where $g(qk) = \frac{qk}{1-(qk)^2} - \frac{1}{2}\ln(\frac{1+qk}{1-qk})$. If we define qk as a new variable t,

$$\begin{aligned} t &\in (0, \frac{1}{2}] \quad \Leftarrow q > 0, \\ t &\in [-\frac{1}{2}, 0) \ \Leftarrow q < 0. \end{aligned} \tag{33}$$

If $t = 0$, $g(t) = 0$. The derivative of the function $g(t)$ is

$$g^{'}(t) = \frac{t^4 + t^2}{(1-t^2)^2} > 0. \tag{34}$$

Thus, $g(t)$ is a monotonically increasing function, and $g(t) > 0$ if $q > 0$. Conversely, $g(t) < 0$ if $q < 0$. When $q = 1$, we will know that $g(k) > 0$. Further, $f(k)$ ($U_i[q]$) is a monotonically increasing (decreasing) function when $q \ge 0$. Because $g(-t) = -g(t)$, when $q = -1$ or $q = -2$,

$$\frac{2}{3}g(k) \le \frac{2}{3}g(-qk) = -\frac{2}{3}g(qk), \tag{35}$$

So $f^{'}(k) \le \frac{1}{k^2\ln 2}\cdot\frac{1}{3}g(qk) < 0$, which shows $f(k)$ ($U_i[q]$) is a monotonically decreasing (increasing) function when $q < 0$. To sum up, we can get the minimum value of $U_i[q]$ for different q.

$$\begin{aligned} U_i[q] &\ge P_\infty[q] + \frac{2}{3}P_\infty[1] = (q + \frac{2}{3})/ln2,\ q \ge 0, \\ U_i[q] &\ge P_1[q] + \frac{2}{3}P_1[1] = log_2[(\frac{5}{3})^{\frac{4}{3}}(\frac{4+q}{4-q})^2],\ q < 0. \end{aligned} \tag{36}$$

After proving $L_i[q]$ in the same way, it will indicate that the theorem in (30) is correct for $i \ge 1$. Now we can use (29) to find the most appropriate judgment boundary value for selecting d_i. If $i \to \infty$,

$$\begin{aligned} L_1[q+1] &\le U_\infty[q] \quad \Leftarrow q \ge 0, \\ L_\infty[q+1] &\le U_1[q] \quad \Leftarrow q < 0. \end{aligned} \tag{37}$$

Then we will always have a common overlap region:

$$\begin{aligned} 2.1873 = L_1[2] &\le D_i(2) \le U_\infty[1] = 2.4045, \\ 0.4913 = L_1[1] &\le D_i(1) \le U_\infty[0] = 0.9618, \\ -0.9618 = L_\infty[0] &\le D_i(-1) \le U_1[-1] = -0.4913, \\ -2.4045 = L_\infty[-1] &\le D_i(-2) \le U_i[-2] = -2.1873, \end{aligned} \tag{38}$$

which is true for all $i \geq 1$. Therefore, for H4G2HR-CORDIC, the relationship among selection value, judgment criteria, and overlap interval are presented in Table III. To simplify the hardware implementation with fewer significant bits, $\{2.25, 0.5, -0.5, -2.25\}$ are selected as the judgment boundary from the overlap interval.

TABLE III
H4G2HR-CORDIC: Relationship Among Selection Value, Judgment Criteria and Overlap Interval

Selection Value	Judgement Criteria	Overlapping Interval
2	$w_i \geq 2.25$	$[2.1873, 2.4045]$
1	$0.5 \leq w_i < 2.25$	$[0.4913, 0.9618]$
0	$-0.5 \leq w_i < 0.5$	$[-0.9618, -0.4913]$
-1	$-2.25 \leq w_i < -0.5$	$[-2.4045, -2.1873]$
-2	$w_i < -2.25$	/

D. Number of Iterations of HGH-CORDIC

According to the previous scholars' derivation of the number of iterations of high-radix CORDIC [22], [27], to achieve n-bit precision, we can get the same general rule as follows:

$$N_{hgh} = \lceil \frac{n + log_2(r/2)}{log_2 r} \rceil, \tag{39}$$

where N_{hgh} represents the minimum number of iterations required for HGH-CORDIC. To facilitate the analysis, we still take the commonly used H4G2HV-CORDIC and H4G2HR-CORDIC as examples.

Let's start with H4G2HV-CORDIC again. After some iterations n, we know that the rotation angle β_n between the X-axis and the final vector (x_n, y_n) is:

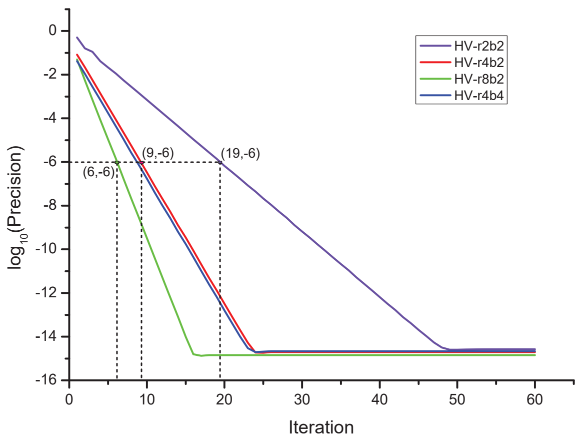
$$\begin{aligned} \beta_n &= \tanh_2^{-1}(y_n/x_n) = \tanh_2^{-1}(w_n 4^{-n}/x_n) \\ &\leq \tanh_2^{-1}[(2+\frac{2}{3})4^{-n}] = \tanh_2^{-1}(\frac{4}{3}\cdot 2^{-2n+1}). \end{aligned} \tag{40}$$

Obviously, the value of $\frac{4}{3}\cdot 2^{-2n+1}$ is between the interval $(2^{-2n+1}, 2^{-2n+2})$. Therefore, the number of iterations required by H4G2HV-CORDIC is half less than that of traditional GHV-CORDIC, under the condition of achieving the same accuracy. This is in line with the expected conclusion.

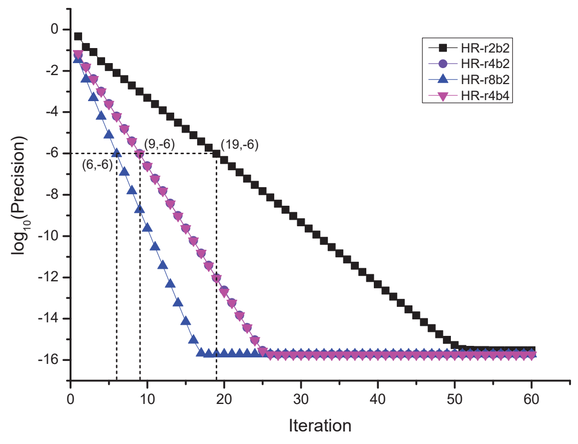
For H4G2HR-CORDIC, the derivation and proof process of iterations is similar to that of high-radix circular CORDIC in [13]. We omit the process in detail here. Also, we can conclude that H4G2HR-CORDIC halves the number of iterations with respect to the traditional GHR-CORDIC.

IV. Software Simulation of HGH-CORDIC

Based on the above theory, we can prove the correctness of HGH-CORDIC through software simulation, and can intuitively observe its advantages. To explain the differences between different radix r and whether there are differences between different base b, we choose four cases for simulation using MATLAB and analyze them by control variable method. The combinations of (r, b) corresponding to these cases are $(2, 2)$, $(4, 2)$, $(8, 2)$, and $(4, 4)$. Their simulation results are shown in Fig. 2. The horizontal coordinate is the number of radix-r CORDIC iterations, and the vertical coordinate is the calculation precision (expressed after taking log_{10} operation). The default numeric type in MATLAB is double.



(a)



(b)

Fig. 2. Computing $log_b x$ and b^x using HGH-CORDIC and traditional GH-CORDIC. (a) Vector Mode; (b) Rotation Mode.

For each case, we randomly selected 10,000 data from different convergence ranges for testing. The results show that the iterations of radix-4 is half of that of radix-2, and that of radix-8 is two-thirds of that of radix-4, which is basically consistent with the results of theoretical derivation. In addition, under the condition of radix-4, no matter how b changes, its convergence speed and calculation precision are basically same, which is also in line with expectations.

V. Analysis of Hardware Implementation

In this section, we will analyze the hardware implementation architecture of HGH-CORDIC, whose general architecture is denoted as $ARCH(r, b)$. Taking the specific case $ARCH(4, 2)$ as an example, we will analyze how to efficiently implement the key components, including inverse scaling factor K_{hgh}^{-1}, selection criteria d_i, and inverse hyperbolic tangent function $tanh_2^{-1}$. Other architectures $ARCH(r, b)$ can learn from this idea and analyze them according to the specific situations.

A. Analysis of the Inverse Scaling Factor

According to Table I, (9) and (11), when we use H4G2HV-CORDIC to implement the logarithmic function, it is still not necessary to calculate the scaling factor. However, when calculating the exponential function, the initial inputs of H4G2HR-CORDIC can not be like (11), x_0 and y_0 should only be 1. At this point, the scaling factor is no longer a predictable constant, and we need to multiply K_{hgh}^{-1} after the number of iterations is over. Obviously, calculating K_{hgh}^{-1} in (15) is a complicated process because d_i is uncertain (d_i^2 has three different values) after each iteration. If it is implemented directly in hardware, it will be very resource-consuming. One of the most common solutions [13] is to take Taylor series expansion in (41) and combine it with a smaller LUT.

$$K_{hgh}^{-1} = (1 - d_i^2 4^{-2i})^{-\frac{1}{2}} \approx 1 + \frac{1}{2} d_i^2 4^{-2i} + \frac{3}{8} d_i^4 4^{-4i} + \cdots \tag{41}$$

For $i > n/4$, K_{hgh}^{-1} can be approximated as 1 because the result of the second term $\frac{1}{2} d_i^2 4^{-2i}$ or more terms is often so small relative to the expected precision that it can be ignored. For $i \geq \lfloor n/8+1 \rfloor$, K_{hgh}^{-1} can be approximated by the first two terms of (41) in n-bit precision. For $i \geq \lfloor n/12+1 \rfloor$, K_{hgh}^{-1} can be approximated by the first three terms. For the remaining iterations, we can implement it with a small LUT. Usually, the scaling factor is determined by the first $n/4+1$ iterations for n-bit precision. Compared with all the use of the LUT method, through the hierarchical implementation, the size of LUT can be reduced from $3^{n/4+1}$ to $3^{\lfloor n/12 \rfloor +1}$.

For example, if $n = 16$, the required size of LUT is only $3^1 \times 16 = 48$ bits ($i = 1$). For the case $i \geq \lfloor n/12+1 \rfloor = 2$, we can compute the scaling factor in each iteration using the shift-and-add operation over the value obtained from LUT and d_i. To use less logic, we can even all use LUT for iterations less than $i = \lfloor n/8+1 \rfloor = 3$. Even so, the LUT only needs to store $3^2 - 1 = 8$ 16-bit data (0 is not stored). Then the table should only have three input bits and 17 output bits (including one integer bit). Therefore, although we do a lot of approximation operations, compared to the full-precision calculation implementation, the accuracy will not be lost. On the contrary, our hardware overhead can be greatly reduced.

B. Analysis of the Selection Criteria

For H4G2HR-CORDIC, as can be seen from Table III, we can select multiple combined boundary constant values within the overlap interval. But for the sake of simplified hardware implementation, we can choose a uniform set of hardware-friendly comparison points $\{\pm 0.5, \pm 2.25\}$. They require at most 1 sign bit, 2 integer bits, and 2 fractional bits. That is, the width of the comparator only needs to be up to 5 bits.

But for H4G2HV-CORDIC, the choice of d_i is related to the variable x_i. From [25], we can reduce the bit width of the comparator without making errors. Otherwise, we have to use an n-bit adder or subtractor to compare two points (L_i and U_i), which not only increases the hardware area but may also have a longer delay than the shifters.

According to the theoretical derivation in [25], the distance between $U_a(x_i)$ and $D_i(a+1)$ must be greater than or equal to $\frac{1}{6} x_i$ and it must be greater than 2^{-f}, where f refers to the truncated fractional bits. Assume that the input range of $log_2 x$ is [1, 2], that is, the initial value range of x_0 of H4G2HV-CORDIC is [2, 3]. Through analysis, the minimum value of x_i in all iterations is 2 and the maximum value is 3 ($max\{w_i\}$=5.43). We can deduce that $f > 1.585$ and we need at least 2 fractional bits. Therefore, we totally assimilate 6 bits for x_i and w_i, including 2 fractional bits, 3 integer bits, and 1 sign bit.

C. Analysis of the Inverse Hyperbolic Tangent Function

No matter which mode HGH-CORDIC works in, they must calculate $\tanh_b^{-1}(d_i r^{-i})$. If we want to calculate it directly, it is very complicated. The general method is to use LUT. As the number of iterations increases, the size of the LUT increases exponentially. To reduce the LUT size, we can find an approximate scheme with low precision loss by Taylor series expansion of $\tanh_b^{-1} x$, which is shown in (42).

$$\tanh_b^{-1} x = \frac{1}{lnb}(x + \frac{1}{3}x^3 + \frac{1}{5}x^5 + \cdots) \tag{42}$$

It is observed that when $i \geq \lceil \frac{n}{6} \rceil$, $\tanh_b^{-1}(d_i r^{-i})$ can be approximated to $(d_i r^{-i})/lnb$. In this case, w_{i+1} in (13) and z_{i+1} in (16) will become

$$w_{i+1} = r(w_i - \frac{1}{lnb} d_i),\ z_{i+1} = z_i + \frac{1}{lnb}(d_i r^{-i}), \tag{43}$$

where $\frac{1}{lnb}$ is a concrete constant. When $d_i \neq 0$, there are only two possible results for either $\frac{r|d_i|}{lnb}$ or $\frac{|d_i|}{lnb}$ for fixed r and b. For example, when $r = 4$ and $b = 2$, $\frac{4|d_i|}{ln2} = 5.7708$ or 11.5416, $\frac{|d_i|}{ln2} = 1.4427$ or 2.8854. The binary representation of 1.4427 is $(1.0111)_2$, the other three constants can be obtained by shifting 1, 2, or 3 bits to the left, respectively.

When i is less than $\lceil \frac{n}{6} \rceil$, we can calculate the complicated term $r^{i+1} \tanh_b^{-1}(d_i r^{-i})$ in (13) or the term $\tanh_b^{-1}(d_i r^{-i})$ in (16) in advance, and then select them by looking up the table. Similarly, when $r = 4$, $b = 2$, and $n = 16$, we only need to store $2 \times 2 = 4$ constants in LUT except that $d_i = 0$. Through the above approximate ideas, we can ensure that the hardware resources are minimized. It is worth learning that we can also use the zero-skipping technique [25] (used in high-radix circular CORDIC) to further reduce the number of iterations (about 20%) and resources.

D. Analysis of Performance

The proposed HGH-CORDIC is not only multi-functional but also can speed up the iteration of hyperbolic CORDIC. A comparative analysis of performance can be concluded in Table IV under the same n-bit precision, including supported functions, supported radix, and iterations.

In terms of hardware resources, high-radix CORDIC requires comparators and a constant multiplier (only when computing b^x) compared to radix-2 CORDIC. The higher the radix, the more comparators are needed for parallel computation,

TABLE IV
COMPARATIVE ANALYSIS OF PERFORMANCE

Item	[17]	[27]	Proposed
Function	$log_b x$, b^x	lnx, e^x	$log_b x$, b^x
Radix	2	$r \geq 8$	$r \geq 4$
Iteration	$n + m$	$\lceil \frac{n - log_2 R}{log_2 r} \rceil + 1$	$\lceil \frac{n + log_2 (r/2)}{log_2 r} \rceil$

m: The total repeated iterations, when $i = 4,\ 13,\ 30, \cdots$
R: R needs to be less than r.

otherwise the critical path will be too long. From the previous work [14], we can observe that when designing a pipelined hardware architecture, the area of radix-4 CORDIC will be smaller than that of radix-2 due to the decreasing number of cascades. Radix-8, on the other hand, has a small gain because the implementation complexity increases dramatically but the number of iterations decreases disproportionately.

In other aspects, the results are presented in Table IV. Compared with [27], we can support the logarithmic and exponential operations with arbitrary base. And we also support radix-4, which has the most cost-effective performance and hardware overhead. Compared with the traditional GH-CORDIC [17], we can reduce more iterations to speed up the calculation of arbitrary $log_b x$ and b^x. In general, the proposed HGH-CORDIC gathers almost all the advantages of traditional hyperbolic CORDIC, making it more perfect for hardware implementation and applications.

VI. CONCLUSION

Compared with applied research, the development speed of theoretical research is relatively lagging behind. To fill the gap in theoretical research, we propose a theory of high-radix generalized hyperbolic CORDIC for more applications. Because base-2 logarithmic and exponential functions are the most commonly used nonlinear symmetric pairing functions in floating-point conversions, and radix-4 CORDIC is the best cost-effective algorithm in hardware, we focus on the specific case (r=4, b=2) to demonstrate the theory of HGH-CORDIC. Then we elaborate on the feasibility and practicability of the theory through simulation and analysis. Finally, the advantages of HGH-CORDIC are summarized through comparative analysis, such as low latency and high precision, which are usually difficult to achieve at the same time.

In the future, we will conduct research on its hardware implementation based on this theory, especially to explore the low-latency and high-precision hardware architectures for its related applications, such as for X^Y-like functions.

REFERENCES

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, 1959.

[2] J. S. Walther, "A unified algorithm for elementary functions," in *Spring. Joint Computer Conf.*, 1971, pp. 379–385.

[3] F. De Dinechin and M. Istoan, "Hardware implementations of fixed-point atan2," in *IEEE 22nd Symposium on Computer Arithmetic (ARITH)*, 2015, pp. 34–41.

[4] M. Langhammer and B. Pasca, "Floating point tangent implementation for FPGAs," in *IEEE 24th Symposium on Computer Arithmetic (ARITH)*, 2017, pp. 64–65.

[5] J. Johnson, "Efficient, arbitrarily high precision hardware logarithmic arithmetic for linear algebra," in *IEEE 27th Symposium on Computer Arithmetic (ARITH)*, 2020, pp. 25–32.

[6] J. S. Walther, "The story of unified CORDIC," *Journal of VLSI Signal Processing*, vol. 25, no. 2, pp. 107–112, 2000.

[7] Z. F. Yang *et al.*, "Given-rotation-based generalized eigenvalue decomposition processor for MU-MIMO precoding," in *IEEE Int. Symposium Circuits Syst. (ISCAS)*, 2019, pp. 1–4.

[8] A. Shiri and G. K. Khosroshahi, "An FPGA implementation of singular value decomposition," in *Iranian Conf. Elect. Eng.*, 2019, pp. 416–422.

[9] D. Biswas, Z. Ye, E. B. Mazomenos, M. Jobges, and K. Maharatna, "CORDIC framework for quaternion-based joint angle computation to classify arm movements," in *IEEE Int. Symposium Circuits Syst. (ISCAS)*, 2018, pp. 1–5.

[10] M. Heidarpur, A. Ahmadi, M. Ahmadi, and M. Rahimi Azghadi, "CORDIC-SNN: On-FPGA STDP learning with Izhikevich neurons," *IEEE Trans. Circuits Syst. I*, vol. 66, no. 7, pp. 2651–2661, 2019.

[11] A. Chakraborty and A. Banerjee, "Low latency semi-iterative CORDIC algorithm using normalized angle recoding and its VLSI implementation," in *Int. Conf. Commun. Signal Process. (ICCSP)*, 2019, pp. 13–20.

[12] T. K. Rodrigues and E. E. Swartzlander, "Adaptive CORDIC: Using parallel angle recoding to accelerate rotations," *IEEE Trans. Comput.*, vol. 59, no. 4, pp. 522–531, 2010.

[13] E. Antelo, J. Villalba, J. Bruguera, and E. Zapata, "High performance rotation architectures based on the radix-4 CORDIC algorithm," *IEEE Trans. Comput.*, vol. 46, no. 8, pp. 855–870, 1997.

[14] J. Rudagi and S. Subbaraman, "Comparative analysis of radix-2, radix-4, radix-8 CORDIC processors," in *Int. Conf. Inventive Comput. Informat. (ICICI)*, 2017, pp. 378–382.

[15] X. Hu, R. Harber, and S. Bass, "Expanding the range of convergence of the CORDIC algorithm," *IEEE Trans. Comput.*, vol. 40, no. 1, pp. 13–21, 1991.

[16] Y. Luo *et al.*, "CORDIC-based architecture for computing Nth root and its implementation," *IEEE Trans. Circuits Syst. I*, vol. 65, no. 12, pp. 4183–4195, 2018.

[17] Y. Y. Luo *et al.*, "Generalized hyperbolic CORDIC and its logarithmic and exponential computation with arbitrary fixed base," *IEEE Trans. VLSI Syst.*, vol. 27, no. 9, pp. 2156–2169, 2019.

[18] S. Mopuri and A. Acharyya, "Low complexity generic VLSI architecture design methodology for N^{th} root and N^{th} power computations," *IEEE Trans. Circuits Syst. I*, vol. 66, no. 12, pp. 4673–4686, 2019.

[19] Y. Zhang *et al.*, "High-precision method and architecture for base-2 softmax function in DNN training," *IEEE Trans. Circuits Syst. I*, vol. 70, no. 8, pp. 3268–3279, 2023.

[20] H. Chen, K. Cheng, Z. Lu, Y. Fu, and L. Li, "Hyperbolic CORDIC-based architecture for computing logarithm and its implementation," *IEEE Trans. Circuits Syst. II*, vol. 67, no. 11, pp. 2652–2656, 2020.

[21] W. Hong *et al.*, "Low-cost high-precision architecture for arbitrary floating-point Nth root computation," in *IEEE Int. Symposium Circuits Syst. (ISCAS)*, 2023, pp. 1–5.

[22] J. Bruguera, E. Antelo, and E. Zapata, "Design of a pipelined radix 4 CORDIC processor," *Parallel Computing*, vol. 19, no. 7, pp. 729–744, 1993.

[23] T. Lang and P. Montuschi, "Very-high radix combined division and square root with prescaling and selection by rounding," in *Proceedings of the 12th Symposium on Computer Arithmetic (ARITH)*, 1995, pp. 124–131.

[24] M. Anane, H. Bessalah, M. Issad, N. Anane, and H. Salhi, "Higher radix and redundancy factor for floating point SRT division," *IEEE Trans. VLSI Syst.*, vol. 16, no. 6, pp. 774–779, 2008.

[25] J. Villalba, E. Zapata, E. Antelo, and J. Bruguera, "Radix-4 vectoring CORDIC algorithm and architectures," *J. VLSI Signal Process. Syst.*, vol. 19, pp. 127–147, 1998.

[26] A. Changela, M. Zaveri, and A. Lakhlani, "ASIC implementation of high performance radix-8 CORDIC algorithm," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2018, pp. 699–705.

[27] E. Antelo, T. Lang, and J. D. Bruguera, "Very-high radix CORDIC rotation based on selection by rounding," *J. VLSI Signal Process. Syst.*, vol. 25, pp. 141–153, 2000.