Generalized Hyperbolic CORDIC and Its Logarithmic and Exponential Computation With Arbitrary Fixed Base

Yuanyong Luo, Yuxuan Wang, Yajun Ha, *Senior Member, IEEE*, Zhongfeng Wang, *Fellow, IEEE*, Siyuan Chen, and Hongbing Pan

***Abstract*— This paper proposes a generalized hyperbolic COordinate Rotation Digital Computer (GH CORDIC) to directly compute logarithms and exponentials with an arbitrary fixed base. In a hardware implementation, it is more efficient than the state of the art which requires both a hyperbolic CORDIC and a constant multiplier. More specifically, we develop the theory of GH CORDIC by adding a new parameter called base to the conventional hyperbolic CORDIC. This new parameter can be used to specify the base with respect to the computation of logarithms and exponentials. As a result, the constant multiplier is no longer needed to convert base *e* (Euler's number) to other values because the base of GH CORDIC is adjustable. The proposed methodology is first validated using MATLAB with extensive vector matching. Then, example circuits with 16-bit fixed-point data are implemented under the TSMC 40-nm CMOS technology. Hardware experiment shows that at the highest frequency of the state of the art, the proposed methodology saves 27.98% area, 50.69% power consumption, and 6.67% latency when calculating logarithms; it saves 13.09% area, 40.05% power consumption, and 6.67% latency when computing exponentials. Both calculations do not compromise accuracy. Moreover, it can increase 13% maximum frequency and reduce up to 17.65% latency accordingly compared to the state of the art.**

***Index Terms*— Architecture, exponential, generalized hyperbolic COordinate Rotation Digital Computer (GH CORDIC), hyperbolic CORDIC, logarithm.**

I. Introduction

THE computation of logarithms and exponentials is widely used in VLSI circuits design. Generally, logarithms and exponentials with base 2 are called binary logarithms and exponentials; logarithms and exponentials with base *e* are called natural logarithms and exponentials; logarithms and exponentials with base 10 are called decimal logarithms and exponentials. In general, there are two kinds of approaches to evaluate logarithms and exponentials: approximation method and iterative method. Although loads of well-related research achievements have been proposed on these methods, there is still plenty of room for improvement. First, current approaches do not support easy porting to other fixed bases while they are needed. Second, current approaches still have room to further reduce the hardware overheads. In this paper, we will propose a promising solution to abovementioned concerns.

The following literature addresses the evaluation of logarithms and exponentials using the approximation method. [1]–[4] evaluate binary logarithms and exponentials via simple piecewise linear approximation. When the output approaches zero, this method encounters notably large relative error. In order to overcome this shortage, Nam *et al*. [5] perform finer subdivisions around the output of zero since the error increases as the output value gets closer to zero. Subsequently, they have designed a processor of the logarithmic number system for 3-D graphics. The main shortcoming of a simple linear approximation method is the high relative error with limited lookup tables. Paul *et al.* [6] use a second-order polynomial approximation method to reduce the relative error. The main contribution of [6] is approximating the multiplication required for second-order approximation by a method that is fast and results in a small error. The aforementioned efforts all adopt the uniform segmentation scheme to approximate binary logarithms and exponentials, consequently incurring too many segments (lookup tables). A nonuniform segmentation and piecewise linear approximation method, which yields smaller segments, is therefore proposed in [7]. Based on the idea of nonuniform segmentation scheme, Liu *et al*. [8] and Zhu *et al.* [9] make the worst case error the same in all segments. Ha and Lee [10] further optimize the hardware implementation proposed in [8] in terms of hardware efficiency. In addition to binary logarithms and exponentials, a piecewise polynomial approximation for natural logarithms and exponentials is presented in [11]. To approximate natural logarithms and exponentials, Langhammer and Pasca [12] adopt a third-order piecewise Taylor expansion approximation method which yields a faster processing speed than the method provided in [11]. All the approximation methods above require lookup tables. Lookup tables of larger size contribute to higher computing precision.

In addition to the approximation approach, the iterative method is also widely exploited. A fast and fixed-point iterative method for computing binary logarithms is presented

Manuscript received December 2, 2018; revised February 26, 2019, April 14, 2019, and May 3, 2019; accepted May 24, 2019. Date of publication June 18, 2019; date of current version August 23, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61376075 and in part by the Key Research and Development Program of Jiangsu Province under Grant BE2015153. *(Corresponding authors: Yuanyong Luo; Hongbing Pan.)*

Y. Luo, Y. Wang, Z. Wang, and H. Pan are with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China (e-mail: luoyuanyong@yeah.net; 542590982@qq.com; zfwang@nju.edu.cn; phb@nju.edu.cn).

Y. Ha is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail: hayj@shanghaitech.edu.cn).

S. Chen is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: nj.chen.siyuan@gmail.com).

Digital Object Identifier 10.1109/TVLSI.2019.2919557

in [13]. This method requires shift, add, and multiplication operations. As an alternative, the digit-recurrence method with selection by rounding is capable of producing one accurate bit in each iteration. Pineiro *et al.* [14] exploits this digit-recurrence method with high-radix arithmetic units to reduce the number of iterations when extracting binary logarithms and exponentials. Today, many commercial applications demand decimal floating-point arithmetic units. Therefore, Chen *et al.* [15], [16] propose the architectures of decimal logarithms and exponentials using the digit-recurrence method mentioned above. In addition, hyperbolic COordinate Rotation Digital Computer (CORDIC)-based iterative method for natural logarithms and exponentials deserves mentioning [17], [18]. This method computes natural logarithms as well as exponentials only by simple shift-add operations and is widely deployed in many fields, including statistics [19]–[21], support vector machine in machine learning [22]–[24], fire neuron [25], turbo decoding [26], arbitrary waveform generator [27], and so forth.

In summary, the approximation method is applicable for low-latency applications with the penalty of low-precision computation, whereas the iterative method is suitable for high-precision computation with the penalty of high latency. From the literature survey presented above, we also observe that all methods consider a fixed base when extracting logarithms and exponentials. If altering the base to other fixed value, the careful design process is much needed, especially for approximation method (binary and natural bases) and digit-recurrence method (binary and decimal bases), which is very annoying to IC designers. The hyperbolic CORDIC-based approach has been widely deployed in many applications as it distinguishes from others for its definite and simple iterative formulas with high computing precision. However, it is only capable of directly calculating natural logarithms and exponentials. When computing logarithms and exponentials with other bases rather than natural base using hyperbolic CORDIC, an extra constant multiplier is always needed to convert base of Euler's number to other values [28]–[31]. Therefore, this paper aims at making the conventional hyperbolic CORDIC stronger than before by giving it the capability of calculating logarithms and exponentials with the arbitrary fixed base without using an extra constant multiplier. The effort is valuable because other bases besides natural base are also widely needed, such as a binary base for 3-D graphics [5], the decimal base for commercial applications [15], [16], and even other arbitrary fixed base for computing compound interest and so forth.

Circular CORDIC was invented by Volder [32], [33] in 1959 to evaluate trigonometric functions. To make the generation of the rotation direction fully parallel, an approximate Circular CORDIC was proposed recently [40]. Based on Circular CORDIC, John Walter developed the hyperbolic CORDIC for the calculation of natural logarithms, natural exponentials, and square roots [34], [35]. To achieve the goal, this paper extends the computation capacities of the conventional hyperbolic CORDIC by adding a new parameter called base, and consequently develops the generalized hyperbolic CORDIC (GH CORDIC), which is capable of computing logarithms and exponentials with an arbitrary fixed base by specifying the new parameter without using an extra constant multiplier. It will be seen in this paper that conventional hyperbolic CORDIC is a special case of GH CORDIC. After the derivation of GH CORDIC, we found that the work of this paper can also guide the deeper research of [36] to fully scale the computation range of circular CORDIC. This will be elaborated in the last paragraph of Section III-B.

GH CORDIC-based methodology for computing logarithms and exponentials with an arbitrary fixed base is validated on the platform of MATLAB with extensive vector matching. Software experiment reveals that the accuracy of the proposed method is consistent with the hyperbolic CORDIC-based approach to computing natural logarithms and exponentials. Subsequently, example circuits with 16-bit input data are modeled using Verilog HDL and synthesized using Design Compiler under the TSMC 40-nm CMOS technology. Hardware experiment shows that at the highest frequency of the state of the art, the proposed methodology saves 27.98% area, 50.69% power consumption, and 6.67% latency when calculating logarithms; it saves 13.09% area, 40.05% power consumption, and 6.67% latency when computing exponentials. Both calculations do not compromise accuracy. In addition, it can increase 13% maximum frequency and reduce up to 17.65% latency accordingly when compared to the state of the art.

The paper is organized as follows. Section II introduces the conventional hyperbolic CORDIC-based method for calculating logarithms and exponentials with a arbitrary fixed base, where an additional constant multiplier is needed. Section III develops the theory of GH CORDIC. GH CORDIC-based methodology for computing logarithms and exponentials with an arbitrary fixed base is outlined in Section IV. In addition, software testing is performed in this section. Then, we give the hardware experimental results and compare our methodology with the prior arts in Section V. Finally, we summarize our work in Section VI.

II. HYPERBOLIC CORDIC-BASED APPROACH

In this section, the hyperbolic CORDIC is described. Then, we elaborate on the state-of-the-art method using hyperbolic CORDIC to extract logarithms and exponentials with an arbitrary fixed base.

A. Introduction to hyperbolic CORDIC

hyperbolic CORDIC has two different modes. The first is of rotation mode (HR CORDIC). The second is of vector mode (HV CORDIC). The iterative formulas for hyperbolic CORDIC of these two different modes are given below.

The iterative formulas of HR CORDIC are as follows:

$$x^{i+1} = x^i + \operatorname{sign}(z^i)(2^{-i}y^i) \tag{1a}$$

$$y^{i+1} = y^i + \operatorname{sign}(z^i)(2^{-i}x^i) \tag{1b}$$

$$z^{i+1} = z^i - \operatorname{sign}(z^i)\tanh^{-1}(2^{-i}) \tag{1c}$$

where i starts with 1 and is an integer.

TABLE I
CONVERGENT FUNCTIONS FOR HYPERBOLIC CORDIC

CORDIC	**OUTPUTS**
HV	$x_n = K_h\sqrt{x_0^2 - y_0^2}$
	$y_n = 0$
	$z_n = z_0 + tanh^{-1}(y_0/\ x_0)$
HR	$x_n = K_h(x_0\, cosh\, z_0 + y_0\, sinh\, z_0)$
	$y_n = K_h(y_0\, cosh\, z_0 + x_0\, sinh\, z_0)$
	$z_n = 0$

The iterative formulas of HV CORDIC are as follows:

$$x^{i+1} = x^i - \text{sign}(y^i)(2^{-i}y^i) \tag{2a}$$
$$y^{i+1} = y^i - \text{sign}(y^i)(2^{-i}x^i) \tag{2b}$$
$$z^{i+1} = z^i + \text{sign}(y^i)\tanh^{-1}(2^{-i}) \tag{2c}$$

where i starts with 1 and is an integer.

From (1) and (2), it can be seen that the major difference between HR and HV is the iterated condition, i.e., the sign function. HR CORDIC drives z to 0, while HV CORDIC drives y to 0. For the hyperbolic CORDIC, it needs to be noted that when the iterative sequence number$i$ equals $4, 13, 40\ldots$, the corresponding stage of iteration should be executed twice, otherwise hyperbolic CORDIC will not converge [17]. After several iterations, the outputs of hyperbolic CORDIC will converge to some special functions as shown in Table I [37]. The scale-factor K_h of hyperbolic CORDIC can be computed by

$$K_h = \prod\nolimits_{i=1}^{n}\left(\sqrt{1-2^{-2i}}\right). \tag{3}$$

Note that in (3), items with respect to repeated iterations should be multiplied twice. Table I shows that hyperbolic CORDIC is capable of evaluating inverse hyperbolic tangent, hyperbolic sine, and hyperbolic cosine, which establishes the foundation of computing natural logarithms and exponentials.

B. Method Using hyperbolic CORDIC

Here, we detail the state-of-the-art method using hyperbolic CORDIC to calculate logarithms and exponentials with an arbitrary fixed base. It has been outlined in [17] that hyperbolic CORDIC is capable of computing natural logarithms and exponentials. Based on the above results, [28]–[31] add a constant multiplier to convert the natural base to other values.

More specifically, in order to extract natural logarithms, we need to initialize the inputs of HV CORDIC as follows:

$$x_0 = R+1 \quad y_0 = R{-}1 \quad z_0 = 0. \tag{4}$$

Then, according to Table I, the output of z_n can be formulated as

$$z_n = \tanh^{-1}\left(\frac{R-1}{R+1}\right) = \frac{1}{2}\ln(R). \tag{5}$$

Finally, z_n is left shifted by 1 bit to obtain the actual value of natural logarithms.

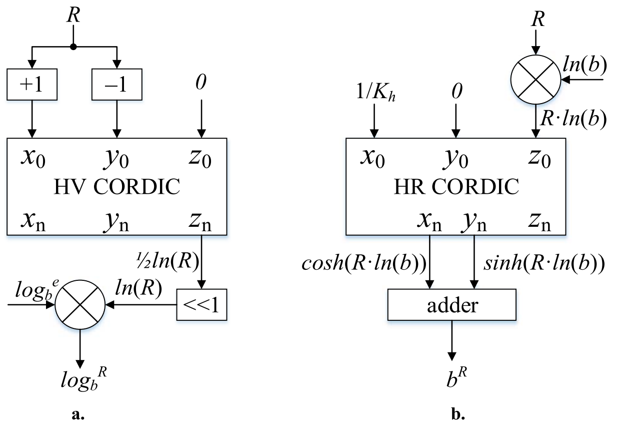


Fig. 1. Computing flow of the state of the art [28]–[31]. (a) Computing flow of $\log_b^R$. (b) Computing flow of b^R.

As for the computation of natural exponentials, initialization of the inputs of HR CORDIC is given as follows:

$$x_0 = \frac{1}{K_h} \quad y_0 = 0; \quad z_0 = R. \tag{6}$$

According to Table I, the outputs of this operation can be expressed as

$$x_n = \cosh(R) \qquad y_n = \sinh(R). \tag{7}$$

So that natural exponentials are computed by an add operation based on an identity

$$\exp(R) = \cosh(R) + \sinh(R). \tag{8}$$

Up until now, the hyperbolic CORDIC-based approach to extracting natural logarithms and exponentials has been clarified. The state-of-the-art method using hyperbolic CORDIC to address the logarithms and exponentials with an arbitrary fixed base is further based on the following two identities [28]–[31]:

$$\log_b^R = \ln(R) \times \log_b^e \tag{9}$$
$$b^R = \exp(R \times \ln(b)). \tag{10}$$

In identities (9) and (10), $\log_b^e$ and $\ln(b)$ are two constants. It is evident that a constant multiplier is required to convert the natural base to arbitrary fixed base b. The entire computing flow is shown in Fig. 1.

III. GENERALIZED HYPERBOLIC CORDIC

In this section, based on the conventional hyperbolic CORDIC, we develop the theory of GH CORDIC. Compared to the conventional hyperbolic CORDIC, GH CORDIC is characterized by a new parameter called base, which can be used to directly convert the natural base of logarithms and exponentials derived by conventional hyperbolic CORDIC to other certain values without using an additional constant multiplier. Theoretically, if the new parameter is specified as Euler's number, the conventional hyperbolic CORDIC is a special case of GH CORDIC. In the following parts of this section, we first define the generalized hyperbolic functions and then develop the theory of GH CORDIC. Characteristic of convergence and convergence range of the proposed new CORDIC are also analyzed in this section.

A. Definitions of Generalized hyperbolic Functions

In order to understand why hyperbolic CORDIC is capable of computing natural logarithms and exponentials, it is essential to trace back to its mathematical essence, i.e., hyperbolic functions. hyperbolic functions include hyperbolic cosine, hyperbolic sine, and hyperbolic tangent. In addition, hyperbolic cosine and hyperbolic sine should meet the relationship of the hyperbolic curve. The following expressions summarize the definitions of hyperbolic functions:

$$\cosh(x) = \frac{e^x + e^{-x}}{2} \tag{11a}$$

$$\sinh(x) = \frac{e^x - e^{-x}}{2} \tag{11b}$$

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{11c}$$

It is easy to verify that hyperbolic cosine (11a) and hyperbolic sine (11b) meet the relationship of hyperbolic curve formulated by

$$\cosh^2(x) - \sinh^2(x) = 1. \tag{12}$$

Based on the definitions of hyperbolic cosine and hyperbolic sine expressed by (11a) and (11b), (8) always holds true. Therefore, the hyperbolic CORDIC can calculate natural exponentials with an additional add operation.

From iterative formulas of (1) and (2), it is clear that hyperbolic CORDIC tightly relates to inverse hyperbolic tangent. According to the definition of hyperbolic tangent expressed by (11c), we can easily derive the expression of an inverse hyperbolic tangent as follows:

$$\tanh^{-1}(x) = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right), \quad \mathrm{x} \in (-1, 1). \tag{13}$$

Combining (13) with (5), we can see that hyperbolic CORDIC is capable of computing natural logarithms because inverse hyperbolic tangent can be defined using natural logarithm as shown in (13).

Based on the analysis above, it is not hard for us to realize that due to the use of Euler's number e in definitions of hyperbolic functions, hyperbolic CORDIC becomes capable of calculating natural logarithms and exponentials. Therefore, in order to achieve the goal of computing logarithms and exponentials with an arbitrary fixed base without using an extra constant multiplier, we need to redefine the hyperbolic functions by replacing the corresponding Euler's number with a parameter.

To avoid confusion, we term our redefinitions of the generalized hyperbolic functions. First, we replace Euler's number e in (11a) and (11b) with a parameter called baseband, thus develop the definitions of generalized hyperbolic cosine and sine denoted as gcosh(·) and gsinh(·)

$$\mathrm{gcosh}(x) = \frac{b^x + b^{-x}}{2} \tag{14a}$$

$$\mathrm{gsinh}(x) = \frac{b^x - b^{-x}}{2}. \tag{14b}$$

TABLE II
FEATURES OF HYPERBOLIC AND GENERALIZED HYPERBOLIC FUNCTIONS

Function / Item	**Hyperbolic**	**Generalized Hyperbolic**
sine	$\frac{e^x - e^{-x}}{2}$	$\frac{b^x - b^{-x}}{2}$
cosine	$\frac{e^x + e^{-x}}{2}$	$\frac{b^x + b^{-x}}{2}$
hyperbolic curve	$cosh^2(x) - sinh^2(x) = 1$	$gcosh^2(x) - gsinh^2(x) = 1$
tangent	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\frac{b^x - b^{-x}}{b^x + b^{-x}}$
inverse tangent	$\frac{1}{2} ln(\frac{1+x}{1-x})$	$\frac{1}{2} log_b^{\frac{1+x}{1-x}} = \frac{tanh^{-1}(x)}{ln(b)}$
equality 1	$tanh^{-1}\left(\frac{R-1}{R+1}\right) = \frac{1}{2} ln(R)$	$gtanh^{-1}\left(\frac{R-1}{R+1}\right) = \frac{1}{2} log_b^R$
equality 2	$exp(R) = cosh(R) + sinh(R)$	$b^R = gcosh(R) + gsinh(R)$

According to (11), (14a), and (14b), generalized hyperbolic tangent denoted as gtanh(·) can be given by

$$\mathrm{gtanh}(x) = \frac{\mathrm{gsinh}(x)}{\mathrm{gcosh}(x)} = \frac{b^x - b^{-x}}{b^x + b^{-x}}. \tag{14c}$$

From (14a) and (14b), it is evident that the following equation always holds true:

$$\mathrm{gcosh}^2(x) - \mathrm{gsinh}^2(x) = 1. \tag{15}$$

Therefore, just like (12), identity (15) has proven that generalized hyperbolic cosine and sine also meet the relationship of the hyperbolic curve.

Up until now, we have defined the generalized hyperbolic functions expressed by (14). Since the conventional hyperbolic CORDIC relates to inverse hyperbolic tangent, it is essential to derive the expression of inverse generalized hyperbolic tangent too. According to (13) and (14c), we have

$$\mathrm{gtanh}^{-1}(x) = \frac{1}{2}\log_b^{\frac{1+x}{1-x}} = \frac{\ln\left(\frac{1+x}{1-x}\right)}{2\ln(b)} = \frac{\tanh^{-1}(x)}{\ln(b)}. \tag{16}$$

Comparing generalized hyperbolic tangent (16) to hyperbolic tangent (13), we can see they are very similar.

It is time to return to the goal of defining generalized hyperbolic functions. Conventional hyperbolic CORDIC is capable of computing natural logarithms and exponentials mainly due to two identities (5) and (8). Similarly, it is easy to verify that we also have the following two identities for generalized hyperbolic functions:

$$g\mathrm{tanh}^{-1}\left(\frac{R-1}{R+1}\right) = \frac{1}{2}\log_b^R \tag{17}$$

$$b^R = \mathrm{gcosh}(R) + \mathrm{gsinh}(R). \tag{18}$$

After observing (17) and (18), we can infer that if hyperbolic functions are replaced by generalized hyperbolic functions, an improved hyperbolic CORDIC will be able to compute logarithms and exponentials with an arbitrary fixed base b. Table II outlines the difference between hyperbolic functions and generalized hyperbolic functions.

B. Iterative Formulas of Generalized hyperbolic CORDIC

In Section III-A, we defined the generalized hyperbolic functions. Here, we will replace the hyperbolic functions used in hyperbolic CORDIC with generalized hyperbolic functions and thus develop the iterative formulas of GH CORDIC.

After observing iterative formulas of hyperbolic CORDIC (1) and (2), it seems that we only need to replace inverse hyperbolic tangent $\tanh^{-1}(\cdot)$ with inverse generalized hyperbolic tangent $\text{gtanh}^{-1}(\cdot) = (\tanh^{-1}(\cdot))/(\ln(b))$. In reality, this is true for HV CORDIC. Since HV CORDIC drives$y$ to 0, the change of angle function has no influence on the trend that (2b) will converge to 0. Thus, we have the iterative formulas of GH CORDIC of vector mode (GHV CORDIC) given by

$$x^{i+1} = x^i - \text{sign}(y^i)(2^{-i}y^i) \tag{19a}$$
$$y^{i+1} = y^i - \text{sign}(y^i)(2^{-i}x^i) \tag{19b}$$
$$z^{i+1} = z^i + \text{sign}(y^i)\frac{\tanh^{-1}(2^{-i})}{\ln(b)} \tag{19c}$$

where i starts with 1 and is an integer.

As for HR CORDIC, further modification is needed in addition to direct replacement. Since HR CORDIC drives z to 0, if the plus-minus sign of the inverse generalized hyperbolic tangent is opposite to that of the inverse hyperbolic tangent, (1c) will no longer converge to 0. Therefore, when b in $(\tanh^{-1}(\cdot))/(\ln(b))$ is less than 1, i.e., when the plus-minus sign of $(\tanh^{-1}(\cdot))/(\ln(b))$ is opposite to that of $\tanh^{-1}(\cdot)$, sign function $\text{sign}(z^i)$ in (1) needs to be replaced by $-\text{sign}(z^i)$. In this way, (1c) will continue to converge to 0. Consequently, we have the iterative formulas of GH CORDIC of rotation mode (GHR CORDIC) given by

$$x^{i+1} = x^i + \alpha(2^{-i}y^i) \tag{20a}$$
$$y^{i+1} = y^i + \alpha(2^{-i}x^i) \tag{20b}$$
$$z^{i+1} = z^i - \alpha\frac{\tanh^{-1}(2^{-i})}{\ln(b)} \tag{20c}$$
$$\alpha = \begin{cases} \text{sign}(z^i), & \text{if } b > 1 \\ -\text{sign}(z^i), & \text{if } 0 < b < 1 \\ = \text{sign}((b-1)z^i) \end{cases} \tag{20d}$$

where i starts with 1 and is an integer.

To summarize, we simplify and unify expressions (19) and (20) to represent the iterative formulas of GH CORDIC in

$$x^{i+1} = x^i + \alpha(2^{-i}y^i) \tag{21a}$$
$$y^{i+1} = y^i + \alpha(2^{-i}x^i) \tag{21b}$$
$$z^{i+1} = z^i - \alpha\frac{\tanh^{-1}(2^{-i})}{\ln(b)} \tag{21c}$$
$$\alpha = \begin{cases} \text{sign}\left(z^i\right), & \text{if } b > 1 \\ -\text{sign}\left(z^i\right), & \text{if } 0 < b < 1 \end{cases} = \text{sign}\left((b-1)z^i\right) \tag{21d}$$

where i starts with 1 and is an integer.

Up until now, we have developed the iterative formulas of GH CORDIC expressed by (21). Similarly, by replacing hyperbolic functions in Table I with generalized hyperbolic functions, we can get the convergent functions for GH CORDIC as shown in Table III. The scale-factor of GH CORDIC is consistent with that of conventional hyperbolic CORDIC expressed by (3), as the iterative index of GH CORDIC is consistent with that of conventional hyperbolic CORDIC (see Section III-C) and the change of angle function has no influence on this constant.

TABLE III
CONVERGENT FUNCTIONS FOR GH CORDIC

CORDIC	**OUTPUTS**
GHV	$x_n = K_h\sqrt{x_0^2 - y_0^2}$
	$y_n = 0$
	$z_n = z_0 + gtanh^{-1}(y_0/x_0)$
GHR	$x_n = K_h(x_0\, gcosh\, z_0 + y_0\, gsinh\, z_0)$
	$y_n = K_h(y_0\, gcosh\, z_0 + x_0\, gsinh\, z_0)$
	$z_n = 0$

Observing (21), we can find that the method to generalize the conventional hyperbolic CORDIC is similar to that in [36] to scale the computation range of Circular CORDIC. However, [36] only directly replaces the angle $\tan^{-1}(2^{-i})$ with $\tan^{-1}(2^{-i})/(pi/2)$ to calculate $\cos(pi/2x)$ and $\sin(pi/2x)$, where $x \in [-1, 1]$, without using an extra constant multiplier. It does not give the theoretical derivation of that method and has not fully explored that method to scale the computation range of Circular CORDIC. For instance, if replacing $pi/2$ with a negative, the Circular CORDIC will not converge as the sign function also needs to change in this case according to our rigorous study; [36] also lacks the analysis of the characteristic of convergence for the modified Circular CORDIC. Therefore, the work of this paper can guide the deeper research of that method used in [36] to fully scale the computation range of Circular CORDIC.

C. Analysis for Characteristic of Convergence

Although the iterative formulas of GH CORDIC have been presented in Section III-B, the corresponding iterative sequence number remains unresolved. First, we analyze the characteristic of convergence for conventional hyperbolic CORDIC. Then, we further analyze the characteristic of convergence for GH CORDIC and determine its convergent iterative sequence number.

Suppose the angle function of CORDIC is θ_i. According to the theorem of CORDIC [37], each θ_i should be less than twice its next consecutive term θ_{i+1}. That is

$$\frac{\theta_i}{\theta_{i+1}} \leq 2. \tag{22}$$

Otherwise, some iterations need to be repeated to make the CORDIC converge. For instance, the angle function θ_i of Circular CORDIC is $\tan^{-1}(2^{-i})$. Since $(\tan^{-1}(2^{-i}))/(\tan^{-1}(2^{-(i+1)})) \leq 2$ always holds true for $i = 0, 1, 2, \ldots$, the iterative sequence number for Circular CORDIC is set as $i = 0, 1, 2, \ldots$. However, for the hyperbolic CORDIC whose angle function θ_i is $\tanh^{-1}(2^{-i})$, inequality (22) is not satisfied for $i = 1, 2, 3, \ldots$. In fact, the value of $\theta_i/(\theta_{i+1})$ for hyperbolic CORDIC is a little bit bigger than 2. Therefore, in order to make the hyperbolic CORDIC converge, those iterations whose iterative sequence number i equals $4, 13, 40 \ldots$, should repeat once [17].

Finally, we analyze the GH CORDIC whose angle function θ_i is $\text{gtanh}^{-1}(2^{-i})$, i.e., $(\tanh^{-1}(2^{-i}))/\ln(b)$ [see (16) and (21)]. Considering the equation

$$\frac{\text{gtanh}^{-1}(2^{-i})}{\text{gtanh}^{-1}(2^-(i+1))} = \frac{\tanh^{-1}(2^{-i})}{\tanh^{-1}(2^{-(i+1)})}$$

we can see that the value of $(\theta_i)/(\theta_{i+1})$ for GH CORDIC equals that for conventional hyperbolic CORDIC. Thus, it can be concluded that the characteristics of convergence for conventional hyperbolic CORDIC and GH CORDIC are the same. Meanwhile, the iterative sequence number for GH CORDIC is also set as $i = 1, 2, 3, \ldots$, and those iterations whose iterative sequence number i equals $4, 13, 40 \ldots$, should be executed twice too, just like the conventional hyperbolic CORDIC.

D. Analysis for Convergence Range

According to the theorem of CORDIC [37], the convergence range of GH CORDIC depends on the max value of the summation of the rotation angles, which can be defined as

$$\begin{aligned}\theta_{\max} &= \sum_{i=1}^{n} |\theta_i| = \sum_{i=1}^{n} |\text{gtanh}^{-1}(2^{-i})| \\ &= \sum_{i=1}^{n} \frac{\tanh^{-1}(2^{-i})}{|\ln(b)|} \\ &= \frac{1}{|\ln(b)|} \sum_{i=1}^{n} \tanh^{-1}(2^{-i}). \end{aligned} \tag{23}$$

In (23), the repeated iterations ($i = 4, 13, 40 \ldots$) are accumulated twice. For instance, when $i = 4$

$$\theta_{\max} = \theta_{\max} + \frac{\tanh^{-1}(2^{-4})}{|\ln(b)|} + \frac{\tanh^{-1}(2^{-4})}{|\ln(b)|}.$$

Therefore, the convergence range of GHV CORDIC can be given by

$$\begin{aligned}\left|\text{gtanh}^{-1}(\frac{y_0}{x_0})\right| = \left|\frac{\tanh^{-1}\left(\frac{y_0}{x_0}\right)}{\ln(b)}\right| &\leq \theta_{\max} \\ &= \frac{1}{|\ln(b)|} \sum_{i=1}^{n} \tanh^{-1}(2^{-i})\end{aligned}$$

that is

$$\left|\tanh^{-1}\left(\frac{y_0}{x_0}\right)\right| \leq \sum_{i=1}^{n} \tanh^{-1}(2^{-i}). \tag{24}$$

TABLE IV
CONVERGENCE RANGE OF GH CORDIC

CORDIC	**GHV**	**GHR**
Convergence Range	$\left\|tanh^{-1}\left(\frac{y_0}{x_0}\right)\right\| \leq 1.1178$	$\|z_0\| \leq \frac{1.1178}{\|ln(b)\|}$

Meanwhile, the convergence range of GHR CORDIC is given as follows:

$$|z_0| \leq \theta_{\max} = \frac{1}{|\ln(b)|} \sum_{i=1}^{n} \tanh^{-1}(2^{-i}). \tag{25}$$

Now, we give an example of the convergence range for GH CORDIC. Consider the iterative sequence number as $i = 1, 2, 3, \ldots, 12$, i.e., $n = 12$ in (23)–(25). The convergence range for this case is outlined in Table IV.

IV. GENERALIZED HYPERBOLIC CORDIC-BASED APPROACH

This section presents the GH CORDIC-based approach to computing logarithms and exponentials with an arbitrary fixed base and tests it using MATLAB. In order to make the proposed methodology clearer, we set the iterative sequence number of GH CORDIC as $i = 1, 2, 3, \ldots, 12$ and make analysis. In the following parts of this section, we first give the top-level computing flow and then analyze the convergence range of the proposed methodology. Finally, we verify the proposed methodology via MATLAB with extensive vector matching to prove its correctness. Software testing indicates that the accuracy of the proposed methodology is consistent with that of the conventional hyperbolic CORDIC-based approach to computing natural logarithms and exponentials.

A. Top-Level Computing Flow

The flow of using GH CORDIC to compute logarithms and exponentials with an arbitrary fixed base is the same as that of using hyperbolic CORDIC to extract natural logarithms and exponentials. According to the convergent functions shown in Table III and the two identities (17) and (18), the entire computing flows are shown in Fig. 2. Comparing Fig. 2 to Fig. 1, we can find that they are very similar, except Fig. 1 uses additional constant multipliers to convert bases. Essentially, the state-of-the-art method (see Fig. 1) calculates natural logarithms and exponentials and uses constant multipliers to convert the natural base to an arbitrary fixed base, while the proposed methodology (see Fig. 2) achieves the goal by directly specifying the parameter b. It is obvious that the proposed methodology can save a constant multiplier compared with the state of the art.

B. Convergence Range Analysis

In Section III-D, we have clarified the convergence range of GH CORDIC. Here, it is essential to figure out the convergence range of GH CORDIC-based approach to extracting logarithms and exponentials with an arbitrary fixed base.

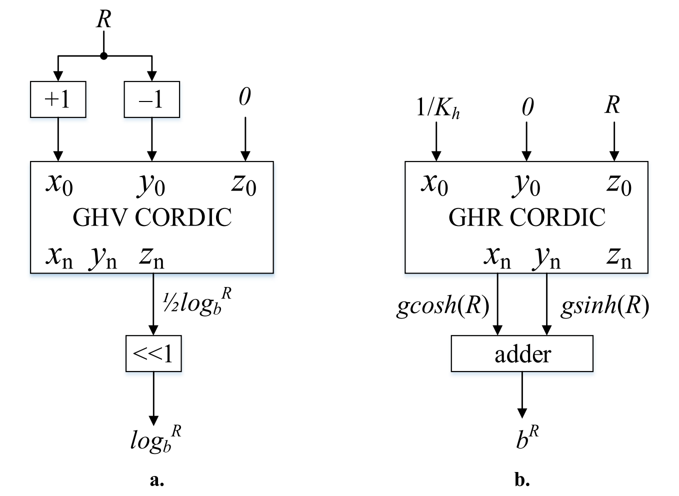


Fig. 2. Computing flow of the proposed approach. (a) Computing flow of $\log_b^R$. (b) Computing flow of b^R.

First, we analyze the convergence range with respect to computing logarithms to an arbitrary fixed base. In this situation, the inputs of GHV CORDIC are $x_0 = (R+1)$ and $y_0 = (R-1)$ as shown in Fig. 2. Considering the convergence range of GHV CORDIC expressed by (24), we could derive the following constraint:

$$\left|\tanh^{-1}\left(\frac{R-1}{R1}\right)\right| \le \sum_{i=1}^{n} \tanh^{-1}(2^{-i})$$

that is

$$R \in \left[\frac{1-\tanh\left(\sum_{i=1}^{n}\tanh^{-1}(2^{-i})\right)}{1+\tanh\left(\sum_{i=1}^{n}\tanh^{-1}(2^{-i})\right)}, \frac{1+\tanh\left(\sum_{i=1}^{n}\tanh^{-1}(2^{-i})\right)}{1-\tanh\left(\sum_{i=1}^{n}\tanh^{-1}(2^{-i})\right)}\right]. \tag{26}$$

Consider the iterative sequence number as $i = 1, 2, 3, \ldots, 12$. According to Table IV, we have $\sum_{i=1}^{12} \tanh^{-1}(2^{-i}) = 1.1178$. Then, (26) becomes

$$\text{R} \in [0.107, 9.352].$$

Next, the convergence range with respect to computing exponentials with an arbitrary fixed base needs to be addressed. The input of GHR CORDIC is $z_0 = R$ as shown in Fig. 2. Considering the convergence range of GHR CORDIC expressed by (25), we have

$$|R| \le \frac{1}{|\ln(b)|} \sum_{i=1}^{n} \tanh^{-1}(2^{-i}). \tag{27}$$

Accordingly, if the max iterative sequence number of GHR CORDIC is 12 and referring to Table IV, (27) becomes

$$|R| \le \frac{1.1178}{|\ln(b)|}.$$

In summary, the convergence range of the proposed methodology is given by (26) and (27). Note that using the techniques proposed in [34], such convergence range can be largely expanded. Observing (26), we know that different choices of parameter b have no influence on the input range of R when calculating $\log_b^R$. However, according to (27), it is evident that different selections of parameter b result in different input ranges of R when extracting b^R.

TABLE V

STATISTICS OF SOFTWARE TESTING MODEL

Item	**Logarithm (GHV)**	**Exponential (GHR)**
computing flow	Fig. 2a	Fig. 2.b
iterative index	$i = 1, 2, 3, \ldots, 12$	
iterative formula	(21a), (21b) and (21c)	
$\boldsymbol{b}$	$b > 0$ and $b \neq 1$	
sign function ($\boldsymbol{\alpha}$)	$-sign(y^i)$	$sign(z^i)$ for $b > 1$; $-sign(z^i)$ for $0 < b < 1$
input range of $\boldsymbol{R}$	$[0.107, 9.352]$	$[-\frac{1.1178}{\lvert\ln(b)\rvert}, \frac{1.1178}{\lvert\ln(b)\rvert}]$
$\boldsymbol{1/K_h}$	-	1.207497046771112

C. Software Testing

In preparation for software testing, the scale-factor needs to be determined. We set the iterative sequence number as $i = 1, 2, 3, \ldots, 12$. Therefore, according to (3), we have

$$K_h = 0.828159375357508.$$

From Fig. 2, $2/K_h$ is a constant input to GHR CORDIC. Thus, it needs to be precomputed as follows:

$$2/K_h = 1.207497046771112.$$

On the grounds of the iterative formulas (21) for GH CORDIC and the top-level computing flow shown in Fig. 2, we code the proposed methodology using MATLAB and test the relative error. The statistics with respect to the testing model is summarized in Table V.

For software testing, we choose the average mean of the relative error to represent accuracy. The relative error is defined as

$$\text{Err}_r = \left|\frac{T-C}{T}\right|. \tag{28}$$

In (28), T means the true value of logarithm or exponential derived by MATLAB's inbuilt functions. C represents the result derived from the proposed methodology. Suppose that the number of logarithms or exponentials we will test is denoted as *Num*. Then the average mean of relative error is given by

$$\text{Avg\_Err}_r = \frac{\sum_{j=1}^{\text{Num}} \text{Err}_r}{\text{Num}}. \tag{29}$$

First, we test the proposed methodology for computing logarithms to an arbitrary fixed base. There are two steps to complete this task. For the first step, the parameter b is set as Euler's number, and we generate 50 million random R as the inputs to calculate ln (R). In this scenario, it is equivalent to using conventional hyperbolic CORDIC to compute natural logarithms. After running MATLAB, the average mean of

TABLE VI
DETAILS OF SOFTWARE TESTING

DUT / Item	Logarithm (GHV)		Exponential (GHR)	
	$b = e$ (HV)	b ∈ (0,1) ∪ (1,100]	$b = e$ (HR)	b ∈ (0,1) ∪ (1,100]
No. of b	1	500	1	500
No. of R for each b	50,000,000	100,000	50,000,000	100,000
Test totals	50,000,000	50,000,000	50,000,000	50,000,000
Avg_Err_r	3.8023×10^{-4}	3.8026×10^{-4}	1.2228×10^{-4}	1.2228×10^{-4}

a. DUT: design under test.
b. For the convenience of software testing, the upper bound of random b is set as 100. Actually, there is no upper bound for this parameter as shown in Table V.

relative error turns out to be 3.8023×10^{-4}. For the second step, we generate 500 random number specified as parameter b. For each parameter b, we generate 100 thousand random R as the inputs to compute $\log_b^R$. Thus, we will test 50 million logarithms to an arbitrary fixed base. It turns out that the average mean of relative error for this scenario is 3.8026×10^{-4}, which is equal to that of hyperbolic CORDIC-based approach to calculating natural logarithms. Therefore, we can conclude that the accuracy of GH CORDIC-based approach to extracting logarithms to arbitrary fixed base equals that of conventional hyperbolic CORDIC-based approach to calculating natural logarithms.

Then, we test the proposed methodology for computing exponentials with an arbitrary fixed base. Two steps are needed to complete this task. In step one, the parameter b is set as Euler's number and we generate 50 million random R as the inputs to compute $\exp(R)$. In this scenario, it is equivalent to using conventional hyperbolic CORDIC to compute natural exponentials. The average mean of relative error derived by MATLAB turns out to be 1.2228×10^{-4}. In step two, we generate 500 random numbers specified as parameter b. For each parameter b, we generate 100 thousand random R as the inputs to compute b^R. It turns out that the corresponding average mean of relative error for this scenario is 1.2228×10^{-4} too, exactly equivalent to that of hyperbolic CORDIC-based approach to calculating natural exponentials. Therefore, we can conclude that the accuracy of GH CORDIC-based approach to extracting exponentials with arbitrary fixed base equals that of conventional hyperbolic CORDIC-based approach to calculating natural exponentials.

Table VI summarizes the details of the software testing. It shows that the proposed methodology is correct. It also reveals that the accuracy of the proposed methodology (see Fig. 2) equals that of the hyperbolic CORDIC-based approach to computing natural logarithms and exponentials (through extensive software testing, it turns out that this conclusion is applicable to different values of n).

V. HARDWARE IMPLEMENTATION AND COMPARISON

In this section, 16-bit fixed-point implementation for the proposed methodology is carried out under the TSMC 40-nm CMOS technology. The example circuits are described using Verilog HDL based on the software testing model shown in Table V. In order to achieve high processing speed, we adopt pipelined structures. Since GH CORDIC differs from the conventional hyperbolic CORDIC, we also propose the corresponding iterative architectures of each stage in three different scenarios. To clarify the advantages of the proposed approach (see Fig. 2) over the state of the art [28]–[31], 16-bit fixed-point implementation for the hyperbolic CORDIC-based solution (see Fig. 1) is modeled in a pipelined manner as well. In the following parts of this section, details of hardware design, hardware timing analysis, error analysis, implementation results, and comparison with the state of the art are presented. In addition, we have also compared our approach with the prior art of approximation methods in this section.

A. Details of Hardware Design

The top-level architectures for the proposed approach have been clarified in Fig. 2. It is evident that the key point lies in the architecture of each cascaded stage of GH CORDIC for a pipelined implementation. Comparing (21) with (1) and (2), we can find out that there are two major differences between the proposed GH CORDIC and the conventional one. The first one is the prestored constants, changing from $\tanh^{-1}(2^{-i})$ to $(\tanh^{-1}(2^{-i}))/\ln(b)$. The second one is the judge condition, or the sign function. Conventional hyperbolic CORDIC of rotation mode holds an invariable sign function $\text{sign}(z^i)$. However, the proposed GH CORDIC of rotation mode has two different sign functions depending on the parameter b. Accordingly, we propose three iterative architectures for the GH CORDIC with respect to three different scenarios. As shown in Fig. 3, two iterative architectures are built for the GHR CORDIC and the other one is built for the GHV CORDIC. If the prestored constants are not considered, iterative architectures described by Fig. 3(a) and (c) remain the same as that of the conventional hyperbolic CORDIC. Although the sign function of GHR for $b < 1$ differs from that for $b > 1$, the hardware overhead for implementing the judge condition keeps the same for both cases as they all judge the sign bit of z_i and then select the add or subtraction operations. For instance, if the given $b > 1$ and the sign bit of z_i is 1, the data x_i and y_i go to adders while z_i goes to a subtractor [see Fig. 3(a)]; if the given $b < 1$ and the sign bit of z_i is 1, the data x_i and y_i go to subtractors while z_i goes to an adder [see Fig. 3(b)]. Both hardware overhead is only a 1-bit comparator. It should be noted that parameter b is determined before implementing the designs. In summary, the proposed GH CORDIC requires six adders or subtractors in hardware, the same as the conventional hyperbolic CORDIC.

Note that in the pipelined architecture of GH CORDIC, for those iterations whose index equals $4, 13, 40 \ldots,$ the corresponding stage needs to be cascaded twice [17]. This is consistent with the conventional hyperbolic CORDIC. Therefore, according to Table V, there is one repeated iteration whose index is 4 because the iterative index range is set as $i = 1, 2, 3, \ldots, 12$. To sum up, 13 cascaded stages are required.

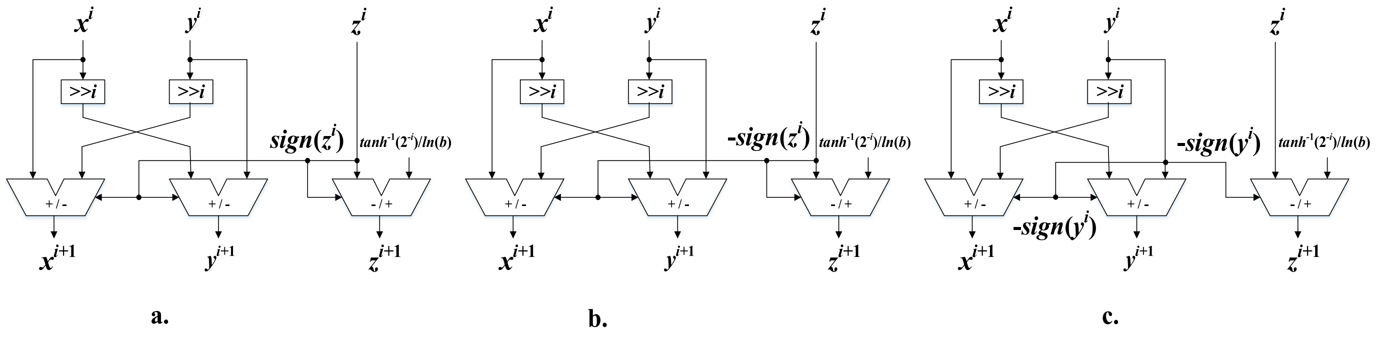


Fig. 3. Iterative architectures of each stage for GH CORDIC.

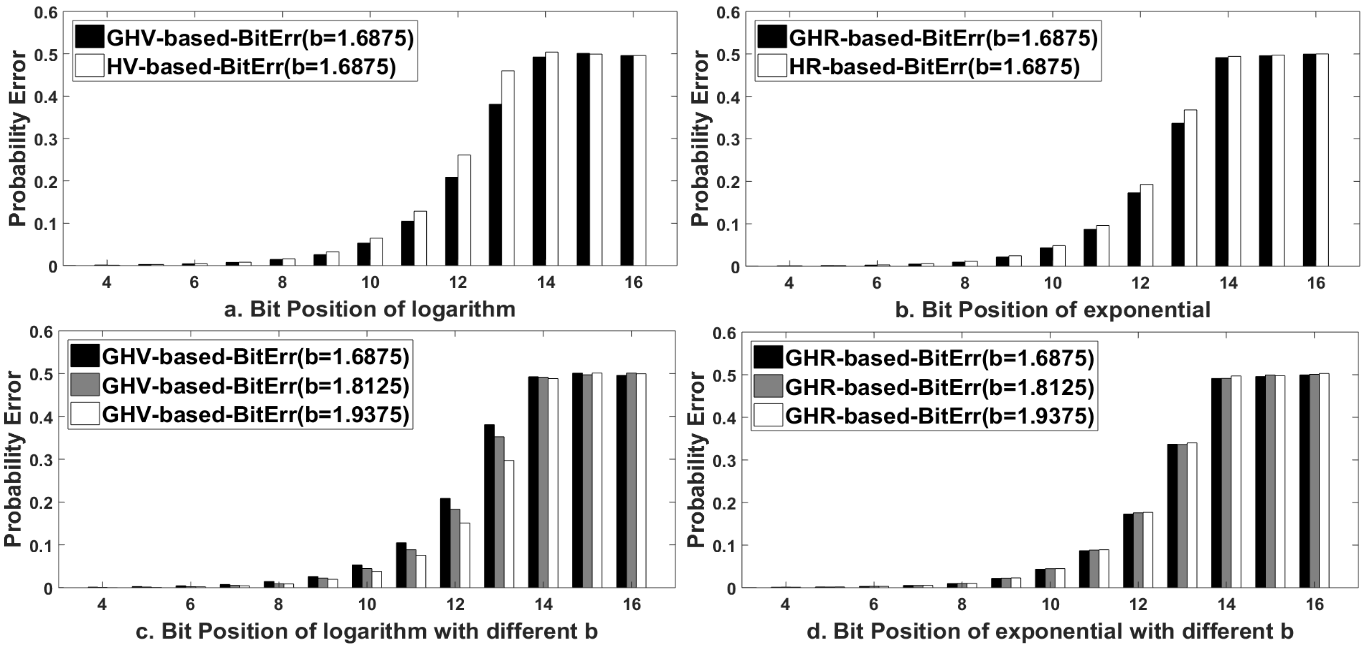


Fig. 4. Bit-position-error experiments of the proposed methodology and the state-of-the-art method [28]–[31]. (a) GHR for $b > 1$. (b) GHR for $0 < b < 1$. (c) GHV.

Before describing the example circuits using Verilog HDL, it is necessary to determine the word length. According to the average mean of relative error provided in Table VI, we finally set 12 fractional bits for GHV-based circuits and 13 fractional bits for GHR-based circuits. The Avg_Err$_r$ for GHV-based model is 3.8×10^{-4}, while the minimum number that 12 fractional bits can represent is $1/(2^{12}) = 2.44\times10^{-4}$, which is just lower than the Avg_Err$_r$. Similarly, the minimum number that 13 fractional bits can represent is $1/(2^{13}) = 1.2207\times10^{-4}$, just lower than 1.2228×10^{-4}, the Avg_Err$_r$ for GHR-based model. More bits setting for fractional parts will be useless since they are not accurate. As for the integral parts, 3 bits and 2 bits are assigned to the GHV-based and GHR-based circuits, respectively. The input range of R for GHV-based model is $[0.107, 9.352]$ according to Table V; therefore, 3 integral bits setting for it narrows the input range of R to be $[0.107, 8)$. Considering the adding 1 operation before the GHV CORDIC [see Fig. 2(a)] and the maximum range that 3 integral bits can represent, the input range of R for computing $\log_b^R$ is further limited between $[0.107, 7)$ so that overflow is avoided. Similarly, 2 integral bits setting for GHR-based model restricts the input range of R to be $(-3, 3)$ for computing b^R. We can select suitable b to let $[-1.1178/|\ln(b)|, 1.1178/|\ln(b)|]$, i.e., the input range of R for GHR-based model (see Table V) adapt to $(-3, 3)$. Moreover, a sign bit is added in front of every datum. The word length setting for the benchmark circuits (see Fig. 1) is similar to the example circuits for the proposed methodology. Table VII illustrates the word length setting for both example and benchmark circuits. Actually, the word length setting for HV CORDIC is consistent with that of [38]. Since we set 16-bit word length for both example circuits and benchmark circuits, they are comparable in terms of hardware overhead.

Up until now, we have clarified the details of hardware design, including the iterative architectures of GH CORDIC and the word length setting for hardware implementation.

B. Timing Analysis

We make the timing analysis for the example circuits and the benchmark circuits here. As mentioned above, CORDIC used

TABLE VII
WORD LENGTH SETTING FOR HARDWARE IMPLEMENTATION

function	**CORDIC**	**Item**	**Sign bit**	**Integral bit**	**Fractional bit**	**Total bit**
$\boldsymbol{log_b^R}$	Fig. 1a (HV)	R, 0, log_b^e	1	3	12	16
	Fig. 2a (GHV)	R, 0	1	3	12	16
$\boldsymbol{b^R}$	Fig. 1b (HR)	$1/K_h$, R, 0, $ln(b)$	1	2	13	16
	Fig. 2b (GHR)	$1/K_h$, R, 0	1	2	13	16

in our implementation has 13 cascaded stages including one repeated iteration. For each stage, one clock cycle is needed. Thus, CORDIC requires 13 clock cycles. As shown in Fig. 2, an additional one clock cycle is required to complete the add operation outside the GH CORDIC. Therefore, the proposed methodology consumes 14 clock cycles in total. As for the state-of-the-art approach shown in Fig. 1, two extra clock cycles are required to complete add and multiplication operations outside the hyperbolic CORDIC, resulting in 15 clock cycles in total. Numerical examples showing the execution steps with respect to the clock cycle are outlined in Table VIII.

The critical path of the proposed methodology is a shift-add operation while that of the state-of-the-art method is a multiplication operation. Generally, the latency with respect to multiplication is larger than that of a shift-add operation. Thus, the proposed methodology promises a higher frequency. Suppose the frequency of the proposed methodology is F_p GHz and that of the state-of-the-art method is F_s GHz. We have the latency expressions shown as follows:

$$t_p = \frac{14}{F_p}\text{ns} \tag{30a}$$

$$t_s = \frac{15}{F_s}\text{ns}. \tag{30b}$$

Since the maximum F_p is supposed to be greater than the maximum F_s, the minimum latency t_p with respect to the proposed methodology is anticipated to be less than the minimum latency t_s of the state-of-the-art method. Because the above analysis is independent of silicon technology, the proposed methodology will show better hardware performance than the state-of-the-art method as long as they are implemented using the same technology.

C. Error Analysis

In Section IV-C, we have justified the correctness of the proposed methodology by software testing. Here, we test if the proposed approach will jeopardize the accuracy when compared with the state-of-the-art method [27]–[29], [39]. First, the base b (see Figs. 1–3) is specified as 1.6875 since it is essential to choose a certain value for this parameter in the hardware implementation. In other words, the constants $\{(\tanh^{-1}(2^{-i}))/(\ln(1.6875))\}_{i=1:12}$ are prestored in a memory. Then, four groups of 100 000 random numbers generated by MATLAB are taken as the inputs of the example circuits and the benchmark circuits accordingly. Finally, the computed results from ModelSim are compared with the true results calculated by MATLAB to determine the accuracy of the circuits. In order to visualize the computing precision of the circuits, we perform the "*bit position error*" [39] experiment, which computes the probability of error for each bit. The results of the bit-position-error comparison are shown in Fig. 4(a) and (b). The black bar and the white bar represent bit-position-error of the proposed GH CORDIC-based approach and the conventional hyperbolic CORDIC-based approach, respectively, to calculating logarithms and exponentials. It is apparent that the bit error of 3 least significant bits approaches 0.5, which means those bits are not accurate. However, in terms of the probability of error, the GH CORDIC-based approach outperforms the state-of-the-art method slightly at the remaining bits. Therefore, we can infer that the proposed methodology will not damage the accuracy when comparing with the state-of-the-art method.

In addition to the comparison with the state-of-the-art method, it is also necessary to explore the bit-position-error of the proposed methodology with respect to different base b. This is achieved by specifying different b in Verilog HDL code and then testing their accuracies as abovementioned process. Fig. 4(c) and (d) depict the bit-position-error for $b = 1.6875$, 1.8125, and 1.9375. Clearly, the precision up to 13 out of 16 bits is still somewhat accurate. Through the tests for numerous values of b, the following conclusion is obtained: the proposed GHV-based methodology for computing logarithms shows better performance of bit-position-error when the value of $|\ln(b)|$ increases; the proposed GHR-based methodology for computing exponentials shows better performance of bit-position-error when the value of $|\ln(b)|$ decreases. The bit-position-error plotted by Fig. 4(c) and (d) concurs with this conclusion.

D. Implementation Results and Comparison With the State of the Art

Using 16-bit fixed-point input data, the example circuits of the proposed methodology (see Fig. 2) and the benchmark circuits of the state-of-the-art method (see Fig. 1) are coded in Verilog HDL and implemented under the TSMC 40-nm CMOS technology. The architecture of each cascaded stage of GH CORDIC is shown in Fig. 3. The parameter base b is randomly set as 0.6456. In order to evaluate the energy efficiency, sampling rate per watt [39] is adopted as a criterion. Assuming the frequency is f GHz, then the sampling rate equals $1000\times f$ MSPS (million samples per second). We further assume that the power is w mW. Thus, the sampling rate per watt can be given by

$$\frac{(1000\times f)\text{MSPS}}{(w)\text{mW}} = \frac{10^6\times f}{w}\text{MSPS/W}. \tag{31}$$

We synthesize the circuits by Design Compiler and present the implementation results in Table IX. It shows that the highest frequency of the state-of-the-art method is 2.94 GHz. When tackling the computation of logarithms to base 0.6456 at this frequency, compared with the conventional HV-based circuits, the proposed GHV-based circuits can save 27.98%

TABLE VIII
NUMERICAL EXAMPLES SHOWING THE EXECUTION STEPS WITH RESPECT TO CLOCK CYCLE

Clock Cycle	**Operation**	$\boldsymbol{log_3^R\|_{R=6}}$ **(HV: Fig. 1a)**	$\boldsymbol{log_3^R\|_{R=6}}$ **(GHV: Fig. 2a)**
1	Add/Sub	$\begin{cases} R+1 \to x_0 = 7 \\ R-1 \to y_0 = 5 \end{cases}$	$\begin{cases} R+1 \to x_0 = 7 \\ R-1 \to y_0 = 5 \end{cases}$
2~14	CORDIC	$z_n \ll 1 \to ln(R) = 1.79176$	$z_n \ll 1 \to log_3^R = 1.631$
15	Multiply	$ln(R) \times log_3^e \to log_3^R = 1.631$	——
Clock Cycle	**Operation**	$\boldsymbol{3^R\|_{R=0.8}}$ **(HR: Fig. 1b)**	$\boldsymbol{3^R\|_{R=0.8}}$ **(GHR: Fig. 2b)**
1	Multiply	$R \times ln(3) \to z_0 = 0.87889$	——
2~14	CORDIC	$\begin{cases} cosh(z_0) \to x_n = 1.4117 \\ sinh(z_0) \to y_n = 0.9965 \end{cases}$	$\begin{cases} gcosh(R) \to x_n = 1.4117 \\ gsinh(R) \to y_n = 0.9965 \end{cases}$
15	Add	$x_n + y_n \to 3^R = 2.4082$	$x_n + y_n \to 3^R = 2.4082$

TABLE IX
IMPLEMENTATION RESULTS FROM DESIGN COMPILER

Computation	**Freq./Clock Cycle**	**Technique**	**Area($\boldsymbol{\mu m^2}$)**	**Power($\boldsymbol{mW}$)**	**Latency($\boldsymbol{ns}$)**	**MSPS/W**
$\boldsymbol{log_b^R\|_{b=0.6456}}$	2.94GHz/0.34ns	HV [29]-[31]	14236.656138	17.8462	5.10	1.6474×10^5
		GHV	10253.308904	8.7994	4.76	3.3411×10^5
		Saving/Improving	**27.98%/-**	**50.69%/-**	**6.67%/-**	**-/2.0281×**
	3.33GHz/0.3ns	GHV	13129.569758	11.6185	4.20	2.8690×10^5
$\boldsymbol{b^R\|_{b=0.6456}}$	2.94GHz/0.34ns	HR [28], [30]	13230.940904	15.6512	5.10	1.8785×10^5
		GHR	11498.928112	9.3834	4.76	3.1332×10^5
		Saving/Improving	**13.09%/-**	**40.05%/-**	**6.67%/-**	**-/1.6680×**
	3.33GHz/0.3ns	GHR	15894.110556	12.6662	4.20	2.6317×10^5

a. HV-based and HR-based techniques are the state-of-the-art method for logarithms and exponentials with arbitrary fixed base (see Fig. 1).
b. GHV-based and GHR-based techniques are the proposed methodology for logarithms and exponentials with arbitrary fixed base (see Fig. 2).
c. Power=Internal Power + Switching Power + Leakage Power.
d. MSPS/W: Million Samples Per Second Per Watt.

area, 50.69% power consumption, and 6.67% latency. When addressing the calculation of exponentials with base 0.6456 at this frequency, compared to the conventional HR-based circuits, the proposed GHR-based circuits can save 13.09% area, 40.05% power consumption, and 6.67% latency. In addition, as mentioned in Section V-B, the shorter critical path of the proposed methodology promises a higher maximum frequency than the state-of-the-art method. The maximum frequency of the example circuits is up to 3.33 GHz. Thus, the proposed methodology is capable of yielding 1.13 times the processing speed and 0.8235 times the latency than the state-of-the-art method. In other words, the example circuits can save up to 17.65% latency when compared to the benchmark circuits.

According to the frequency and power reported by Design Compiler and formula (31), the sampling rate per watt is calculated to indicate the energy efficiency of the example and benchmark circuits. From the last column of Table IX, we can see that at the highest frequency of the state-of-the-art method (2.94 GHz), the proposed GH CORDIC-based methodology can process average 330 billion logarithms or exponentials per second per watt, i.e., 330 billion logarithms or exponentials per joule. However, the state-of-the-art method can only tackle average 175 billion logarithms or exponentials per joule. In other words, the energy efficiency of the proposed methodology is 1.85 times than that of the state-of-the-art method.

TABLE X
PARAMETER SETTING FOR HARDWARE IMPLEMENTATION

GHV: $\boldsymbol{log_2^R}$			**[8], [10]:** $\boldsymbol{log_2^{(R)}}$		
Item	High Precision	Low Precision	Item	High Precision	Low Precision
Iterative Index	$i = 1,2,\ldots 25$	$i = 1,2 \ldots 9$	**No. of Regions**	6564	26
Range of R	$[0.11, 9.35]$		**Range of R**	$[0.11, 9.35]$	
Data Width	32	15	**Data Width**	32	15

It should be noted that the hardware overheads of the benchmark circuits may be different for different bases because of the existence of a constant multiplier (see Fig. 1). However, whatever the fixed base is, the improvements of abovementioned hardware criteria for the example circuits will always exist due to the proposal's pure reduction of a constant multiplier.

TABLE XI
COMPARISON WITH THE APPROXIMATION METHOD

Precision	**Method**	**Area(μm^2)**	**Power(mW)**	**Latency(ns)**	**ALP**	**MAE**
High Precision	GHV	60132.643799	66.6396	10.64	4.26×10^7	4.49×10^{-8}
	[8], [10]	831641.484935	449.5292	2.58	9.64×10^8	4.13×10^{-8}
Low Precision	GHV	8043.628397	9.5970	3.08	2.38×10^5	3.07×10^{-3}
	[8], [10]	5428.416072	6.0268	1.20	3.93×10^4	2.86×10^{-3}

a. The circuits are synthesized under the TSMC 40-nm CMOS technology.
b. Implementation results with minimized latency are presented above.
c. ALP: a figure of merit to characterize the overall hardware overheads.
d. MAE: mean absolute error.

In summary, the proposed methodology outperforms the state-of-the-art method at all hardware criteria, including area, power consumption, latency, and energy efficiency.

E. Comparison With the Approximation Method

In addition to the comparison with the conventional hyperbolic CORDIC-based approach, we also make the comparison with the state-of-the-art approximation methods. Liu *et al*. [8] propose a novel linear approximation algorithm to enable the maximum absolute error the same in all segments for the computation of binary logarithm. When implementing this algorithm model for low-precision computation, Liu *et al.* [8] and Ha and Lee [10] substitute several adders for a multiplier to reduce hardware overheads by only considering some most significant bits for each region. However, such an implementation scheme not only sacrifices the accuracy of the algorithm model but also is impractical when the number of regions is too big, i.e., when high precision is required. Therefore, to facilitate effective comparison, we directly employ a multiplier to implement the linear approximation algorithms used in [8] and [10]. Table X summarizes the parameter settings for the hardware implementation. In order to see the tradeoff between the proposed approach and the approximation method, we have carefully designed two comparative experiments: high-precision comparison and low-precision comparison.

To characterize the accurate performance of the circuits, mean absolute error (MAE) is employed. To evaluate the overall hardware overheads, we adopt a figure of merit, which is characterized by *area*, *latency*, and *power* (ALP)

$$\text{ALP} = \text{area} \times \text{latency} \times \text{power}. \tag{32}$$

The comparison results are outlined in Table XI. It shows that, at the high precision around 4.3e^{-8}, ALP of the approximation method is almost 23 times larger than the proposed iterative approach. Therefore, the proposed GH CORDIC-based approach is preferred when high precision is required such as in commercial applications [15]. While at the low precision around 3.0e^{-3}, ALP of the proposed iterative approach is almost six times larger than the approximation method. Thus, the approximation method is preferred when only low precision is required such as in error-tolerant image processing [40].

VI. CONCLUSION

In this paper, following Walther's work [34], [35], we have proposed the theory of GH CORDIC, which shows stronger computation capacities than the conventional hyperbolic CORDIC. Subsequently, GH CORDIC-based methodology for the computation of logarithms and exponentials with an arbitrary fixed base has been presented as an improvement for the state-of-the-art method [28]–[31] that requires both a conventional hyperbolic CORDIC and a constant multiplier. The correctness of the proposed approach has been validated by software testing with extensive vector matching. At the same time, software testing has also verified the theory of GH CORDIC. Example circuits and benchmark circuits with 16-bit fixed-point input data have been implemented under the TSMC 40-nm CMOS technology. As shown in Sections V-C and V-D, hardware implementation reveals that the proposed methodology outperforms the state-of-the-art method at all hardware criteria, including area, power consumption, latency, and energy efficiency without compromising accuracy. When compared to the approximation method [8], [10], Section V-E indicates that the proposed iterative methodology has significant advantages when high precision is required.

The proposed GH CORDIC-based methodology for computing logarithms and exponentials provides a promising solution to many applications, such as binary base-required 3-D graphics [5], decimal base-required commercial applications [15], [16], and even other arbitrary fixed base-required applications like calculating compound interest. In addition, the proposed GH CORDIC can be configured and assembled to complete more tasks as its computation capacity is greater than the conventional hyperbolic CORDIC. Therefore, there remains an additional potential application value to be discovered. We also have applied lots of the conventional hyperbolic CORDIC-based research achievements to the proposed GH CORDIC, such as the method to expand the convergence range of hyperbolic CORDIC [37] and the method to wipe out the scale-factor of hyperbolic CORDIC [27]. It turns out they are all applicable to the proposed GH CORDIC. Thus, it can be anticipated that the proposed GH CORDIC will have a big potential to supersede the conventional one, given that it is more powerful and it inherits lots of research achievements based on the conventional hyperbolic CORDIC. As for the study value, the work of this paper can guide the deeper research of the method used in [36] to fully scale the computation range of Circular CORDIC because the study is not deep and complete from a new perspective.

References

[1] M. Combet, H. Van. Zonneveld, and L. Verbeek, "Computation of the base two logarithm of binary numbers," *IEEE Trans. Electron. Comput.*, vol. EC-14, pp. 863–867, Dec. 1965.

[2] E. L. Hall, D. D. Lynch, and S. J. Dwyer, "Generation of products and quotients using approximate binary logarithms for digital filtering applications," *IEEE Trans. Comput.*, vol. C-19, no. 2, pp. 97–105, Feb. 1970.

[3] K. H. Abed and R. E. Siferd, "CMOS VLSI implementation of a low-power logarithmic converter," *IEEE Trans. Comput.*, vol. 52, no. 11, pp. 1421–1433, Nov. 2003.

[4] K. H. Abed and R. E. Siferd, "VLSI implementation of a low-power antilogarithmic converter," *IEEE Trans. Comput.*, vol. 52, no. 9, pp. 1221–1228, Sep. 2003.

[5] B. G. Nam, H. Kim, and H. J. Yoo, "A low-power unified arithmetic unit for programmable handheld 3-D graphics systems," *IEEE J. Solid-State Circuits*, vol. 42, no. 8, pp. 1767–1778, Aug. 2007.

[6] S. Paul, N. Jayakumar, and S. P. Khatri, "A fast hardware approach for approximate, efficient logarithm and antilogarithm computations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 2, pp. 269–277, Feb. 2009.

[7] D. De Caro, M. Genovese, E. Napoli, N. Petra, and A. G. M. Strollo, "Accurate fixed-point logarithmic converter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 61, no. 7, pp. 526–530, Jul. 2014.

[8] C.-W. Liu, S.-H. Ou, K.-C. Chang, T.-C. Lin, and S.-K. Chen, "A low-error, cost-efficient design procedure for evaluating logarithms to be used in a logarithmic arithmetic processor," *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1158–1164, Apr. 2016.

[9] M. Zhu, Y. Ha, C. Gu, and L. Gao, "An optimized logarithmic converter with equal distribution of relative errors," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 9, pp. 848–852, Sep. 2016.

[10] M. Ha and S. Lee, "Accurate hardware-efficient logarithm circuit," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 8, pp. 967–971, Aug. 2017.

[11] F. De Dinechin and B. Pasca, "Floating-point exponential functions for DSP-enabled FPGAs," in *Proc. IEEE Int. Conf. Field-Program. Technol.*, Dec. 2010, pp. 110–117.

[12] M. Langhammer and B. Pasca, "Single precision logarithm and exponential architectures for hard floating-point enabled FPGAs," *IEEE Trans. Comput.*, vol. 66, no. 12, pp. 2031–2043, Dec. 2017.

[13] C. S. Turner, "A fast binary logarithm algorithm [DSP Tips & Tricks]," *IEEE Signal Process. Mag.*, vol. 27, no. 5, pp. 124–140, Sep. 2010.

[14] J.-A. Pineiro, M. D. Ercegovac, and J. D. Bruguera, "Algorithm and architecture for logarithm, exponential, and powering computation," *IEEE Trans. Comput.*, vol. 53, no. 9, pp. 1085–1096, Sep. 2004.

[15] D. Chen, L. Han, and S. B. Ko, "Decimal floating-point antilogarithmic converter based on selection by rounding: Algorithm and architecture," *IET Comput. Digit. Techn.*, vol. 6, no. 5, pp. 277–289, Sep. 2012.

[16] D. Chen, L. Han, Y. Choi, and S.-B. Ko, "Improved decimal floating-point logarithmic converter based on selection by rounding," *IEEE Trans. Comput.*, vol. 61, no. 5, pp. 607–621, May 2012.

[17] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna, "50 years of CORDIC: Algorithms, architectures, and applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 9, pp. 1893–1907, Sep. 2009.

[18] Y. Wang and V. Dinavahi, "Real-time digital multi-function protection system on reconfigurable hardware," *IET Gener., Transmiss. Distrib.*, vol. 10, no. 10, pp. 2295–2305, Jul. 2016.

[19] Y. Song, T. Zeng, and D. Zeng, "A hardware Gaussian noise generator and evaluation," in *Proc. IET Int. Radar Conf.*, Guilin, China, Apr. 2009, pp. 1–4.

[20] D. Huang, D. Z. Zeng, T. Long, and J. Y. Yu, "Design of a correlated Lognormal distributed sequence generator based on Virtex-IV series FPGA," in *Proc. Int. Conf. Comput. Appl. Syst. Model. (ICCASM)*, Taiyuan, China, Oct. 2010, pp. V2-340–V2-343.

[21] B. G. Sileshi, C. Ferrer, and J. Oliver, "Accelerating hardware Gaussian random number generation using Ziggurat and CORDIC algorithms," in *Proc. IEEE SENSORS*, Valencia, Spain, Nov. 2014, pp. 2122–2125.

[22] T. Yuan, G. Xu, Y. Zou, J. Han, and X. Zeng, "A configurable SVM hardware accelerator for embedded systems," in *Proc. 12th IEEE Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT)*, Guilin, China, Oct. 2014, pp. 1–3.

[23] S. Huang, Z. Xie, J. Han, and X. Zeng, "A flexible low-power machine learning accelerator for healthcare applications," in *Proc. 13th IEEE Int. Conf. Solid-State Integr. Circuit Technol. (ICSICT)*, Hangzhou, China, Oct. 2016, pp. 613–615.

[24] J.-C. Wang, L.-X. Lian, Y.-Y. Lin, and J.-H. Zhao, "VLSI design for SVM-based speaker verification system," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 7, pp. 1355–1359, Jul. 2015.

[25] M. Heidarpour, A. Ahmadi, and R. Rashidzadeh, "A CORDIC based digital hardware for adaptive exponential integrate and fire neuron," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 11, pp. 1986–1996, Sep. 2016.

[26] K. Babionitakis, Y. Dagres, K. Nakos, and D. Reisis, "A VLSI architecture for minimizing the transmission power in OFDM transceivers," in *Proc. 10th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, vol. 1, Dec. 2003, pp. 308–311.

[27] S. Aggarwal, P. K. Meher, and K. Khare, "Scale-free hyperbolic CORDIC processor and its application to waveform generation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 2, pp. 314–326, Feb. 2013.

[28] T. Lang and E. Antelo, "High-throughput CORDIC-based geometry operations for 3D computer graphics," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 347–361, Mar. 2005.

[29] G. Helvacioğlu, A. B. K. Y. K. Alp, and C. Kasnakoğlu, "Reduced CORDIC based logarithmic convertor," in *Proc. 25th Signal Process. Commun. Appl. Conf. (SIU)*, Antalya, Turkey, May 2017, pp. 1–4.

[30] Á. Vazquez, J. Villalba, and E. Antelo, "Computation of decimal transcendental functions using the CORDIC algorithm," in *Proc. 19th IEEE Symp. Comput. Arith.*, Portland, OR, USA, Jun. 2009, pp. 179–186.

[31] L. Bangqiang, H. Ling, and Y. Xiao, "Base-N logarithm implementation on FPGA for the data with random decimal point positions," in *Proc. IEEE 9th Int. Colloq. Signal Process. Appl.*, Kuala Lumpur, Malaysia, Mar. 2013, pp. 17–20.

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

[33] J. E. Volder, "The birth of CORDIC," *J. VLSI Signal Process.*, vol. 25, no. 2, pp. 101–105, 2000.

[34] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. 38th Spring Joint Comput. Conf.*, Atlantic City, NJ, USA, May 1971, pp. 379–385.

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

[36] J. Valls, T. Sansaloni, A. Perez-Pascual, V. Torres, and V. Almenar, "The use of CORDIC in software defined radios: A tutorial," *IEEE Commun. Mag.*, vol. 44, no. 9, pp. 46–50, Sep. 2006.

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

[38] D. R. Llamocca-Obregón and C. P. Agurto-Ríos, "A fixed-point implementation of the expanded hyperbolic CORDIC algorithm," *Latin Amer. Appl. Res.*, vol. 37, no. 1, pp. 83–91, 2007.

[39] Y. Luo, Y. Wang, H. Sun, Y. Zha, Z. Wang, and H. Pan, "CORDIC-based architecture for computing Nth root and its implementation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4183–4195, Dec. 2018.

[40] L. Chen, J. Han, W. Liu, and F. Lombardi, "Algorithm and design of a fully parallel approximate coordinate rotation digital computer (CORDIC)," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 3, no. 3, pp. 139–151, Jul./Sep. 2017.

Yuanyong Luo received the B.S. degree in applied physics from Jilin University, Changchun, China, in 2016. He is currently working toward the Ph.D. degree at the School of Electronic Science and Engineering, Nanjing University, Nanjing, China.

From 2014 to 2015, he hosted an engineering project, the system of aerial photography with three cooperative and unmanned aerial vehicles, at Jilin University. His current research interests include digital integrated circuit design and VLSI implementation of signal processing algorithms, machine learning algorithms, and digital communications.

Yuxuan Wang received the B.S. and M.Sc. degrees in electronic science and engineering from Nanjing University, Nanjing, China, where he is currently working toward the Ph.D. degree at the School of Electronic Science and Engineering.

His current research interests include VLSI computing IP optimization and the coordinated acceleration of software and hardware in machine learning.

Yajun Ha (S'98–M'04–SM'09) received the B.S. degree from Zhejiang University, Hangzhou, China, in 1996, the M.Eng. degree from the National University of Singapore, Singapore, in 1999, and the Ph.D. degree from Katholieke Universiteit Leuven, Leuven, Belgium, in 2004, all in electrical engineering.

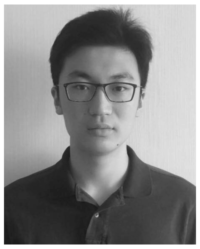
He was an Assistant Professor with the National University of Singapore. He was a Scientist and the Co-Director of the I2R-BYD Joint Laboratory, Institute for Infocomm Research, Singapore, and an Adjunct Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore. He is currently a Professor with ShanghaiTech University, Shanghai, China. His current research interests include reconfigurable computing, ultralow-power digital circuits and systems, embedded system architecture and design tools for applications in robots, smart vehicles, and intelligent systems. He has published around 100 internationally peer-reviewed journal/conference papers on these topics.

Dr. Ha has served a number of positions in the professional communities. He has been a Program Committee Member for a number of well-known conferences in the fields of FPGAs and design tools, such as Design Automation Conference (DAC), Design Automation Test in Europe, Asia and South Pacific-DAC (ASP-DAC), International Symposium on Field-Programmable Gate Arrays, International Conference on Field-Programmable Logic and Applications, and International Conference on Field-Programmable Technology (FPT). He was a recipient of two IEEE/ACM Best Paper Awards, the Shanghai Thousands of Talents Program Award, and the Shanghai Oriental Scholar Program. He has served as the General Co-Chair for ASP-DAC 2014, the Program Co-Chair for FPT 2010 and FPT 2013, the Chair for the Singapore Chapter of the IEEE Circuits and Systems (CAS) Society in 2011 and 2012, a member of the ASP-DAC Steering Committee, and a member of the IEEE CAS VLSI and Applications Technical Committee. He served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS from 2016 to 2019, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS from 2011 to 2013, and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS from 2013 to 2014. He has been an Associate Editor of the *Journal of Low Power Electronics* since 2009.

Zhongfeng Wang (F'16) received the B.E. and M.S. degrees from Tsinghua University, Beijing, China, and the Ph.D. degree from the Department of Electrical and Computer Engineering, the University of Minnesota, Minneapolis, MN, USA, in 2000.

He was with National Semiconductor Corporation, Santa Clara, CA, USA. He was an Assistant Professor with the School of Electrical Engineering and Computer Science (EECS), Oregon State University, Corvallis, OR, USA. He served as a leading VLSI Architect at Broadcom Corporation for nearly nine years. He joined Nanjing University, Nanjing, China, in 2016, as a Distinguished Professor. During his tenure at Broadcom, he has contributed significantly on 10 G/ps and beyond high-speed networking products. Additionally, he has made critical contributions in designing FEC coding schemes for 100- and 400-G/ps Ethernet standards. So far, his technical proposals have been adopted by many international networking standards. He has published over 200 technical papers, edited one book ("VLSI"), and filed tens of U.S. patent applications and disclosures. His current research interests include digital communications, machine learning, and efficient VLSI implementation.

Dr. Wang is a world-recognized Expert on VLSI for Signal Processing Systems. He has served as a Technical Program Committee Member (or Co-Chair), the Session (or Track) Chair, and a Review Committee Member for tens of international conferences. In 2013, he served on the Best Paper Award Selection Committee for the IEEE Circuits and System Society. He was a recipient of the IEEE Circuits and Systems Society VLSI Transactions Best Paper Award in 2007. Since 2007, he has had five papers ranked among top 20 most downloaded manuscripts in the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS. Since 2004, he has been serving as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I—PART I: REGULAR PAPERS, THE IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS, and the IEEE Transactions on Very Large Scale Integration (VLSI) Systems for numerous terms. He was a Guest Editor for a special issue of the IEEE JOURNAL ON EMERGING AND SELECTED TOPICS IN CIRCUITS AND SYSTEMS in 2016.



Siyuan Chen is currently working toward the B.S.E. degree in electrical and computer engineering and computer science at Duke University, Durham, NC, USA. His current research interests include digital integrated circuits design, VLSI for digital signal processing, and design-for-testability.

Hongbing Pan received the B.S. degree in applied physics and the Ph.D. degree in microelectronics and solid-state electronics from Nanjing University, Nanjing, China, in 1994 and 2005, respectively.

From 2006 to 2012, he was an Associate Professor with the Institute of VLSI Design, Nanjing University, where he has been a Professor with the School of Electronic Science and Engineering since 2013. He has authored more than 40 articles. His current research interests include VLSI design, CMOS sensors, reconfigurable computing, and artificial intelligence.