An Efficient Scaling-Free Folded Hyperbolic CORDIC Design Using a Novel Low-Complexity Power-of-2 Taylor Series Approximation

Anu Verma, Khyati Kiyawat, Bishnu Prasad Das, *Senior Member, IEEE*, and Pramod Kumar Meher, *Senior Member, IEEE*

***Abstract*— Hyperbolic trigonometric functions are widely used in several engineering and scientific applications, including digital signal processing (DSP), communication systems, and many others. In this article, we propose a scaling-free hyperbolic coordinate rotation digital computer (CORDIC) algorithm and its architecture based on a novel power-of-2 coefficient low-complexity Taylor series approximation to implement sinh and cosh functions. CORDIC architectures are generally slow due to their high latency of computation. The proposed architecture reduces the latency and achieves the desired precision with only four iterations where an optimized angle set comprised of six CORDIC microrotations are mapped into a four-stage folded-pipeline structure leveraging mutually exclusive behavior of two pairs of microrotations. The proposed design is implemented on field-programmable gate arrays (FPGAs) Xilinx Zedboard using 65.38% less registers with ∼63.63% less latency and 48.97% less power consumption compared with the best of the existing designs. The proposed design is synthesized by Synopsys Design Compiler and place and route (PnR) tool using Taiwan Semiconductor Manufacturing Company (TSMC) 65-nm CMOS process. It consumes ∼76.31% less area, 68.75% less computational delay, and 68.92% less power consumption compared with the best of the existing designs. Moreover, the proposed architecture involves 46.89% less energy per output (EPO) than the best of the existing designs. The error–energy performance (EEP) and the error–area performance (EAP) of the proposed design are, respectively, ∼1.25 times and ∼2.8 times better than that of the best of the existing designs. Besides, the proposed architecture is also implemented and verified on a silicon chip in the TSMC 180-nm CMOS process for the validation of the algorithm and architecture.**

***Index Terms*— Coordinate rotation digital computer (CORDIC), hyperbolic CORDIC, hyperbolic trigonometric functions, Taylor series approximation.**

Manuscript received 16 January 2023; revised 18 April 2023; accepted 28 April 2023. Date of publication 26 June 2023; date of current version 26 July 2023. This work was supported in part by the Science and Engineering Research Board (SERB), Government of India, under Project CRG/2021/007437; and in part by the Young Faculty Research Fellowship (YFRF) of Visvesvaraya Ph.D. Scheme of Ministry of Electronics and Information Technology, Government of India. *(Corresponding author: Bishnu Prasad Das.)*

Anu Verma and Bishnu Prasad Das are with the Department of Electronics and Communication Engineering, Indian Institute of Technology (IIT) Roorkee, Roorkee, Uttarakhand 247667, India (e-mail: averma3@ec.iitr.ac.in; bishnu.das@ece.iitr.ac.in).

Khyati Kiyawat is with the Computer Science Department, University of Virginia, Charlottesville, VA 22903 USA (e-mail: khyati@virginia.edu).

Pramod Kumar Meher is with the Department of Computer Science and Engineering, C. V. Raman Global University, Bhubaneswar, Odisha 752054, India (e-mail: pkmeher@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2023.3281078.

Digital Object Identifier 10.1109/TVLSI.2023.3281078

NOMENICLATURE

MUF	Maximum usable frequency.
Cc	Clock cycles.
EPO	Energy per output.
EEP	Error–energy performance.
SDP	Slice-delay product.
LDP	LUT-delay product.
RDP	Register-delay product.
EAP	Error–area performance.
ADP	Area-delay product.

I. INTRODUCTION

THE hyperbolic trigonometric functions are extensively used in various applications [1], such as digital signal processing (DSP), communication systems, robotics, astrophysics [2], biometric authentication [3], computation of logarithmic and exponential functions [4], artificial neural networks (ANNs) [5], independent component analysis [6], and many others. Various methods and algorithms are proposed for the computation of hyperbolic trigonometric functions [7], [8], and coordinate rotation digital computer (CORDIC) algorithm is one of them. The CORDIC algorithm was proposed by Volder [9]. It performs various computational tasks, including the computation of elementary functions through simple iterative shift–add operations [10]. Although the CORDIC algorithm is simple, it requires several iterations for the computation, which increases the latency of the architecture. Various methods have been proposed to reduce the latency of computation [11], [12], [13], [14], [15], [16], [17], [18]. Scaling compensation generally becomes expensive when the number of CORDIC iterations is optimized. Lin and Wu [19] and Villalba et al. [20] have proposed techniques without the overheads of scaling factor compensation, but all these approaches either demand more design area or affect the latency and compromise with the accuracy. The CORDIC implementation using Taylor series expansion can alleviate the scale-factor overheads to a certain extent to develop scale-free CORDIC architecture with less complexity [21], [22]. Novel angle sets of rotation have been proposed for the fast processing of the CORDIC algorithm using Taylor series expansion in [23] and [24]. Most of the architectures available in the literature are implementing circular CORDIC for the computation of sin and cos functions [14], [25], [26].

The CORDIC on hyperbolic trajectories has additional challenges to be solved due to the unbound nature of hyperbolic trajectory and the range of convergence. The hyperbolic CORDIC algorithm was first given by Walther [27]. The hyperbolic CORDIC architecture proposed by Walther is limited by maximum input angle, i.e., ∼1.18 (radian). Obregon and Rios [28] have tried to extend the range of convergence by introducing negative indices, but the architecture ended up with high latency, and the demand for more I/O bits made this technique less effective. The scale-free hyperbolic CORDIC has been suggested in [29] to compute hyperbolic functions. But, design of Aggarwal et al. [29] has high latency and demands more area. Some other approaches have been proposed [30], [31], [32], for implementing hyperbolic CORDIC for the computation of exponential and logarithmic functions. The existing architectures available for the computation of hyperbolic sine and cosine functions either have high resource utilization or low accuracy and limited input range.

This article presents a scaling-free hyperbolic CORDIC algorithm based on an improved Taylor series approximation for the computation of sinh and cosh functions, which consumes less area with good design accuracy. The main contributions of this article are as follows.

1) A novel low-complexity power-of-2 coefficient Taylor series approximation of hyperbolic functions for achieving a desirable accuracy using only four out of a set of six specific CORDIC microrotations.
2) Four-stage folded pipeline CORDIC architecture, where two pairs of mutually exclusive microrotations are folded into a pair of pipeline stages using common subexpression sharing, which leads to a significant reduction of the latency and hardware complexity of the structure.
3) A low-complexity circuit is proposed for the microrotation for small residual angles.
4) Error analysis of the proposed architecture for 8- and 16-bit word lengths is presented to show the suitability of the proposed architecture for various applications.

The rest of this article is organized as follows. A brief overview of the hyperbolic CORDIC algorithm is explained in Section II. The proposed Taylor series approximation algorithm and the proposed CORDIC architecture for the computation of hyperbolic functions are described in Section III. In Section IV, we have discussed the error analysis and compared the accuracy of the proposed architecture with the existing architectures. The field-programmable gate array (FPGA) implementation results, ASIC post-layout results, and chip implementation results of the proposed architecture are discussed in Section V. Finally, Section VI concludes this article.

II. Basic Concept of Hyperbolic CORDIC and Scale-Free CORDIC Algorithm

This section briefly reviews the conventional hyperbolic CORDIC, its limitations, and the concept of a scaling-free hyperbolic CORDIC algorithm.

A. Hyperbolic CORDIC Algorithm

In 1971, Walther, in his well-known paper [27], has provided a generalized expression of a unified CORDIC algorithm for different trajectories, from which we can obtain the expression of microrotations through an elementary angle $\delta\theta_i$ in the hyperbolic coordinate system as follows:

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = K_i \begin{bmatrix} 1 & d_i \tanh \delta\theta_i \\ d_i \tanh \delta\theta_i & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{1}$$

where $K_i = (1/(1 - (\tanh \delta\theta_i)^2)^{1/2})$ is the scale factor.

The X_i and Y_i are the inputs corresponding to the X and Y coordinates, respectively, associated with the ith microrotation. Here, d_i refers to the direction of the ith microrotation. The desired angle of rotation is given by $\theta = \sum_{i=1}^{N}(d_i \delta\theta_i) + \theta_r$, where N and θ_r are the number of iterations and the residual angle, respectively. One can define the microrotations as $\delta\theta_i = \tanh^{-1}(2^{-i})$, for $i = 1, 2, \ldots, n$, and can have the conventional hyperbolic CORDIC algorithm from (1) as follows:

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = K_i \begin{bmatrix} 1 & d_i 2^{-i} \\ d_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix}, \quad K_i = \frac{1}{\sqrt{1 - (2^{-2i})}}. \tag{2}$$

The conventional CORDIC in hyperbolic trajectory as given by (2) has the following limitations.

1) In order to guarantee the convergence, the microrotations corresponding to $i = \{4, 13, 40, \ldots, k, 3k+1, \ldots\}$ need to be executed twice.
2) The region of convergence (RoC) is limited by $|\theta| \leq 1.1182$ radians, which is nearly 64°.
3) It requires a multiplication of the pair of output by scaling factor, which requires two multipliers for its hardware implementation if both the multiplications are implemented concurrently. These multiplications need to be done in a separate pipeline stage, and the duration of the minimum clock period of the overall circuit would be limited by the time required for the multiplications.
4) The number of iterations cannot be optimized for different angles of rotation, since that would, otherwise, require a generic multiplication circuit, which would impact the throughput and latency of the circuit.

B. Scaling-Free Hyperbolic CORDIC Algorithm

The scaling-free CORDIC uses fewer iterations and does not require multiplications of the outputs by a scaling-factor compensation circuit, which significantly improves performance in terms of area and speed compared with the conventional CORDIC. It is generally implemented by employing the Taylor series expansion of hyperbolic functions as follows.

The ith microrotation in hyperbolic trajectory can be expressed by the basic matrix equation

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = \begin{bmatrix} \cosh \delta\theta_i & \sinh \delta\theta_i \\ \sinh \delta\theta_i & \cosh \delta\theta_i \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix}. \tag{3}$$

Assuming $\delta\theta_i = 2^{-i}$, the Taylor series expansion of $\sinh \delta\theta_i$ and $\cosh \delta\theta_i$ can be written as follows:

$$\sinh(2^{-i}) = \sum_{n=0}^{\infty} \frac{2^{-i(2n+1)}}{(2n+1)!} \tag{4a}$$

TABLE I

ROTATIONAL ERROR IN COMPUTATION OF sinh AND cosh USING THE PROPOSED TAYLOR SERIES APPROXIMATION WITH 8-BIT WORD LENGTH

i (iteration)	$\sinh(\delta\theta_i)$	Rotational error	$\cosh(\delta\theta_i)$	Rotational error
1	$\sinh(2^{-1}) = 2^{-1} = 0.5$	2 x 10^{-2}	$\cosh(2^{-1}) = 1 + 2^{-2}(2^{-1}) = 1.125$	2.6 x 10^{-3}
2	$\sinh(2^{-2}) = 2^{-2} = 0.25$	2.6 x 10^{-3}	$\cosh(2^{-2}) = 1 + 2^{-4}(2^{-1}) = 1.0312$	1.5 x 10^{-4}
3	$\sinh(2^{-3}) = 2^{-3} = 0.125$	3.2 x 10^{-4}	$\cosh(2^{-3}) = 1$	7.8 x 10^{-3}
4	$\sinh(2^{-4}) = 2^{-4} = 0.0625$	4 x 10^{-5}	$\cosh(2^{-4}) = 1$	1.95 x 10^{-3}
5	$\sinh(2^{-5}) = 2^{-5} = 0.03125$	5.08 x 10^{-6}	$\cosh(2^{-5}) = 1$	4.88 x 10^{-4}

It corresponds to 8-bit fixed-point representation, which includes 1 sign bit, 1 integer bit, and 6 fractional bits. In this case, the lowest term of the power of 2 which can be used in sinh, and cosh expression is 2^{-5}.

$$\cosh(2^{-i}) = \sum_{n=0}^{\infty} \frac{2^{-i(2n)}}{(2n)!}. \tag{4b}$$

The factorial terms in the denominators of (4a) and (4b) cannot be expressed exactly in power-of-2 forms. Therefore, sinh and cosh function cannot be computed according to (4) by shift–add operations. For the shift–add implementation of microrotations given by (3), Aggarwal et al. [29] have approximated (3!) to 2^2 in the third-order Taylor series expansions of (4a). But, this approximation results in a large error in the computation. For the reduction of this error, the design of [29] requires more iterations. In this article, we propose an improved Taylor series approximation that provides less computational error. Therefore, the proposed hyperbolic CORDIC has achieved nearly the same accuracy as [29] in significantly less iterations and less hardware complexity.

III. PROPOSED TAYLOR SERIES APPROXIMATION AND SCALE-FREE HYPERBOLIC CORDIC ARCHITECTURE

This section describes the proposed Taylor series approximation for scaling-free hyperbolic CORDIC algorithm for low latency and improved accuracy implementation.

A. Proposed Taylor Series Approximation

The Taylor series approximations of sinh and cosh defined by (4a) and (4b), respectively, with 16-bit word size are given by

$$\sinh(2^{-i}) = 2^{-i} + \frac{2^{-3i}}{3!} + \frac{2^{-5i}}{5!} \tag{5}$$

$$\cosh(2^{-i}) = 1 + \frac{2^{-2i}}{2!} + \frac{2^{-4i}}{4!}. \tag{6}$$

The factorial terms in the denominators of the above expansions could be expressed as follows:

$$\frac{1}{3!} = \frac{1}{6} = \frac{1}{(8-2)} = \frac{1}{8}\left(1 - \frac{1}{4}\right)^{-1} \tag{7a}$$

$$\frac{1}{4!} = \frac{1}{24} = \frac{1}{(32-8)} = \frac{1}{32}\left(1 - \frac{1}{4}\right)^{-1} \tag{7b}$$

$$\frac{1}{5!} = \frac{1}{120} = \frac{1}{(128-8)} = \frac{1}{128}\left(1 - \frac{1}{16}\right)^{-1}. \tag{7c}$$

By using binomial expansion $(1-x)^{-1}$, 1/3!, 1/4!, and 1/5! as given by (7) can be expressed as follows:

$$\frac{1}{3!} \approx \frac{1}{8}\left(1 + \frac{1}{4} + \frac{1}{16} + \cdots\right) \approx 2^{-3}(1 + 2^{-2} + 2^{-4} + \cdots) \tag{8a}$$

$$\frac{1}{4!} \approx \frac{1}{32}\left(1 + \frac{1}{4} + \frac{1}{16} + \cdots\right) \approx 2^{-5}(1 + 2^{-2} + 2^{-4} + \cdots) \tag{8b}$$

$$\frac{1}{5!} \approx \frac{1}{128}\left(1 + \frac{1}{16} + \frac{1}{256} + \cdots\right) \approx 2^{-7}(1 + 2^{-4} + 2^{-8} + \cdots). \tag{8c}$$

Now, using (8), we can express (5) and (6), respectively, as follows:

$$\sinh(2^{-i}) = 2^{-i} + \left[\sum_{n=1}^{n=2} 2^{-(i(2n+1))}\left(\sum_{p} 2^{-(s_{pi})}\right)\right] \tag{9}$$

$$\cosh(2^{-i}) = 1 + \left[\sum_{n=1}^{n=2} 2^{-(i(2n))}\left(\sum_{p} 2^{-(c_{pi})}\right)\right]. \tag{10}$$

Here, s_{pi} and c_{pi} are positive integers. The range of p is different for sinh and cosh functions. It varies with the iteration number i and the input bit width. The values of s_{pi} and c_{pi} for different iterations are shown in the approximation of $\sinh(\delta\theta_i)$ and $\cosh(\delta\theta_i)$ in Tables I and II for 8- and 16-bit word lengths, respectively. The lowest power of 2, which can be used in the sinh and cosh expression, is $2^{-(f-1)}$ (f is the number of fractional bits). In Tables I and II, the rotational error is computed as follows:

$$\text{Rotational error} = |\text{Computed value} - \text{Actual value}|. \tag{11}$$

Equation (9) is the generalized expression of the proposed Taylor series of sinh function for 8- and 16-bit word size in the second column of Tables I and II, respectively. For the 8-bit word size, sinh expression is independent of the value of s_{pi}, since all its microrotations include only the first term, whereas, in the case of the 16-bit word size, the values of s_{pi} (when $n = 1$) for the second, third, fourth, and fifth terms are 3, 5, 7, and 9, respectively. Equation (10) is the generalized expression of the proposed Taylor series of cosh function for 8- and 16-bit word size in the fourth and fifth columns of Tables I and II, respectively. For the 8- and 16-bit word size, the value of c_{pi} (when $n = 1$) is 1 for the second term of the cosh function. However, the values of c_{pi} (when $n = 2$) for the third, fourth, and fifth terms are 5, 7, and 9, respectively. But, these terms are not needed for 8-bit word size. By sharing the common subexpressions of sinh and cosh in the third and sixth columns of Table II, respectively, the shift–add operations are implemented with lower hardware complexity.

The proposed Taylor series approximation of hyperbolic function introduces less error in each microrotation with only a small increase in the area compared with the best of the

TABLE II
ROTATIONAL ERROR IN COMPUTATION OF sinh AND cosh USING THE PROPOSED TAYLOR SERIES APPROXIMATION WITH 16-BIT WORD LENGTH

i (iteration)	$\sinh(\delta\theta_i)$ expressions	Sub-expression sharing of $\sinh(\delta\theta_i)$	Rotational error	$\cosh(\delta\theta_i)$ expressions	Sub-expression sharing of $\cosh(\delta\theta_i)$	Rotational error
1	$\sinh(2^{-1}) = 2^{-1}+2^{-3}(2^{-3}+2^{-5}+2^{-7}+2^{-9})+2^{-5}(2^{-7}) = 0.5209$	$\sinh(2^{-1}) = 2^{-1}(1+2^{-5}[(1+2^{-2})(1+2^{-4})+2^{-6}])$	1.95 x 10^{-4}	$\cosh(2^{-1}) = 1+2^{-2}(2^{-1})+2^{-4}(2^{-5}+2^{-7}+2^{-9}) = 1.1275$	$\cosh(2^{-1}) = (1+2^{-3})+2^{-9}[1+2^{-2}(1+2^{-2})]$	6.2 x 10^{-5}
2	$\sinh(2^{-2}) = 2^{-2}+2^{-6}(2^{-3}+2^{-5}+2^{-7}) = 0.2525$	$\sinh(2^{-2}) = 2^{-2}[(1+2^{-9})+2^{-7}(1+2^{-4})]$	3.6 x 10^{-5}	$\cosh(2^{-2}) = 1+2^{-4}(2^{-1})+2^{-8}(2^{-5}) = 1.0313$	$\cosh(2^{-2}) = 1+2^{-5}(1+2^{-8})$	4.1 x 10^{-5}
3	$\sinh(2^{-3}) = 2^{-3}+2^{-9}(2^{-3}) = 0.1252$	$\sinh(2^{-3}) = 2^{-3}(1+2^{-9})$	8.16 x 10^{-5}	$\cosh(2^{-3}) = 1+2^{-6}(2^{-1}) = 1.0078$	$\cosh(2^{-3}) = 1+2^{-7}$	1.01 x 10^{-5}
4	$\sinh(2^{-4}) = 2^{-4} = 0.0625$	$\sinh(2^{-4}) = 2^{-4}$	4.07 x 10^{-5}	$\cosh(2^{-4}) = 1+2^{-8}(2^{-1}) = 1.0019$	$\cosh(2^{-4}) = 1+2^{-9}$	6.3 x 10^{-7}
5	$\sinh(2^{-5}) = 2^{-5} = 0.03125$	$\sinh(2^{-5}) = 2^{-5}$	5.08 x 10^{-6}	$\cosh(2^{-5}) = 1+2^{-10}(2^{-1}) = 1.0004$	$\cosh(2^{-5}) = 1+2^{-11}$	3.9 x 10^{-8}

It corresponds to 16-bit fixed-point representation, which includes 1 sign bit, 1 integer bit, and 14 fractional bits. In this case, the lowest term of the power of 2 which can be used in sinh, and cosh expression is 2^{-13}.

existing design [29]. Moreover, the proposed approach requires fewer iterations for the sinh and cosh computation with the same or better accuracy compared with the other [29]. The proposed Taylor series approximation is specifically for the sinh and cosh functions and based on the magnitude of CORDIC microrotations. It cannot be used as a generic approximation. But, the approximation strategy to examine the Taylor series expansion for small arguments and successive approximation can be used in other applications. But, actual approximation could widely vary from application to application.

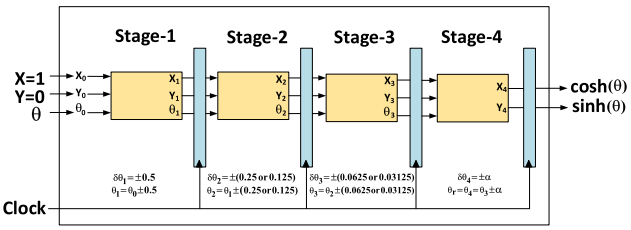


Fig. 1. Block diagram of the proposed architecture for the input range $(-\pi/4, \pi/4)$.

B. Input–Output Word-Length Requirement

If the range of input of the sinh and cosh arguments is R, then the total number of bits required for the representation of input values is given by

$$B_{\text{inp}} = \text{sign-bit} + B_i + f \tag{12}$$

where f and B_i are the numbers of bits in the fractional and integer parts

$$B_i = \lceil |\log_2(R)| \rceil. \tag{13}$$

The number of output bits required to accommodate the values of sinh and cosh can be expressed as follows:

$$B_{\text{out}} = \text{sign-bit} + B_{\text{isc}} + f \tag{14}$$

where B_{isc} is the number of integer bits required for the representation of sinh and cosh value

$$B_{\text{isc}} = \lceil |\log_2(|\sinh R|)| \rceil \text{ for sinh function} \tag{15a}$$

$$B_{\text{isc}} = \lceil |\log_2(\cosh R)| \rceil \text{ for cosh function.} \tag{15b}$$

The I/O word length used in the proposed architecture is based on (12) and (14).

C. High-Level Description of the Proposed Design

A high-level view of the proposed hyperbolic CORDIC architecture for the computation of $\cosh(\theta)$ and $\sinh(\theta)$ for any given value of θ (for $-\pi/4 < \theta < \pi/4$) is shown in Fig. 1. It has three input variables $X = 1$, $Y = 0$, and θ, as shown in Fig. 1. The proposed hyperbolic CORDIC architecture consists of four pipeline stages. The value of residual angle of each stage (which is also the same as the input angle for its following stage) gradually decreases in each successive stage and becomes negligibly small after Stage 4, such that the values of X and Y finally become equal to $\sinh\theta$ and $\cosh\theta$, respectively. The angle through which rotation is to be performed in different pipeline stages is given in Table III. Stage 1 performs the microrotation for $i = 1$, whereas each of Stages 2 and 3 either performs one of the pair of specified rotations or no rotations depending on the magnitude of the input angle of that stage. In Stage 4, a small microrotation is performed. As shown in Table IV, the different initial values are used in the architecture for different input angle ranges. When we extend the input range from $(-\pi/4, \pi/4)$ to $(-\pi, \pi)$, only initial values of inputs (X and Y) need to be changed, but the proposed architecture (as shown in Fig. 1) remains the same for all the input ranges. The number of I/O bits required to implement the proposed design depends on the input range and accuracy requirement.

As shown in Table III, the proposed approximation algorithm achieves acceptable accuracy to be used for various applications using an optimized set of only six microrotations. Besides, the proposed architecture implements the six microrotations in four pipeline stages by folding two pairs of

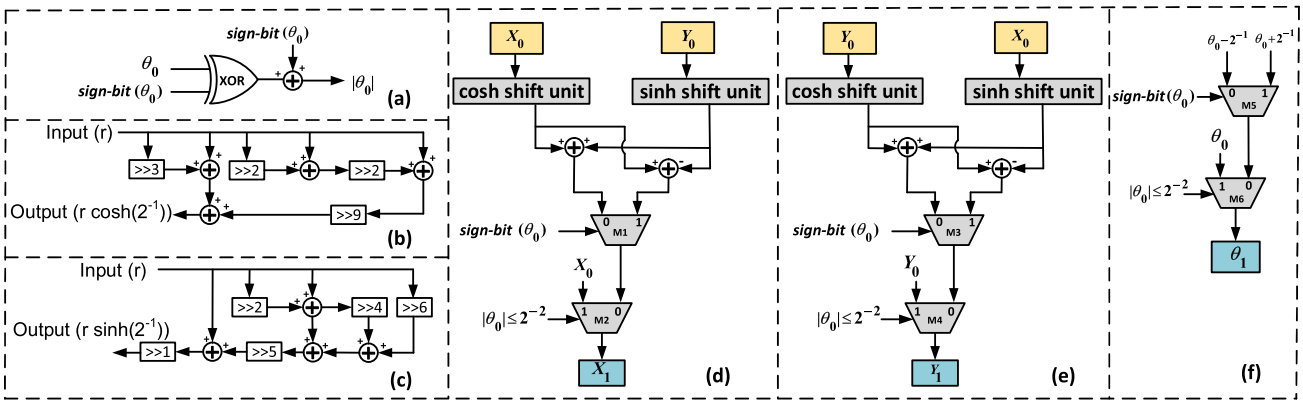


Fig. 2. Structure of Stage 1 of the proposed CORDIC architecture with shift units of 16-bit accuracy. (a) Logic circuit of computation of absolute value of θ using XOR gate. (b) Shift unit of cosh for Stage 1 performs iteration $i = 1$. (c) Shift unit of sinh for Stage 1 performs iteration $i = 1$. (d) Logic circuit of computation of output X_1. (e) Logic circuit of computation of output Y_1. (f) Logic circuit of computation of output θ_1. The structure of sinh and cosh shift units depends on the expressions of sinh and cosh shown in columns 3 and 6 of the second row of Table II, respectively, for iteration $i = 1$.

TABLE III
INPUT ANGLE OF ROTATION IN FOUR PIPELINE STAGES FOR INPUT RANGE $(-\pi/4, \pi/4)$

Stage (j)	i	$\delta\theta_i$(radian)	Rotation conditions
1	1	± 0.5	$\lvert\theta\rvert > 0.25$
2*	2	± 0.25	$\lvert\theta\rvert > 0.1875$
	3	± 0.125	$0.1875 \geq \lvert\theta\rvert > 0.0625$
3*	4	± 0.0625	$\lvert\theta\rvert > 0.0469$
	5	± 0.03125	$0.0469 \geq \lvert\theta\rvert > 0.0156$
4	small micro-rotation	$k \times (1.9 \times 10^{-3})$ (k= 0,1...,8)	$\lvert\theta\rvert \leq 0.0156$

*Each of Stage-2 and Stage-3 either perform one of the pair of specified micro-rotations or no rotations depending on the input angle of the given stage.

TABLE IV
INITIAL VALUES OF INPUTS FOR DIFFERENT RANGES

Magnitude of input angle ($\lvert\theta\rvert$)	X	Y
$(0, \pi/4)$	1	0
$(\pi/4, \pi/2)$	2.51	-2.3
$(\pi/2, 3\pi/4)$	2.51	2.3
$(3\pi/4, \pi)$	11.59	-11.55

mutually exclusive microrotations in Stages 2 and 3 to reduce the latency as well as hardware complexity. The proposed architecture can be extended for higher precision at the cost of more hardware complexity and more number of iterations according to the requirement of applications.

D. Proposed Architecture

The structure and function of different pipeline stages of the proposed structure are discussed in the following. We have used index j to represents the stage number, such that (X_{j-1}, Y_{j-1}, and θ_{j-1}) and (X_j, Y_j, and θ_j) refer to the input and output variables of the jth stage, respectively.

1) Stage 1: This stage performs the first iteration, i.e., $i = 1$, as shown in Table III. If the absolute value of input angle θ_0 is greater than 2^{-2}, then input angle is rotated by $\pm 2^{-1}$; otherwise, it does not perform any rotations and passes on the input to the second pipeline stage. Fig. 2 shows the structure for the implementation of the operations of Stage 1. The absolute value of θ_0 is determined by XORing its sign bit with all the rest of its bits and by adding the sign bit again to the XORed output, as shown in Fig. 2(a). Thus, if the sign bit is 0, the output remains the same as the input (θ_0). Otherwise, the output gets changed to its 2's complement. If the absolute value of $\theta_0 \leq 2^{-2}$, then the values of inputs X_0, Y_0, and θ_0 are assigned to the outputs X_1, Y_1, and θ_1, respectively, without any computation. Otherwise, the input vector is rotated by $\pm 2^{-1}$ (if $\theta_0 > 0$, $\delta\theta_1 = +2^{-1}$, and -2^{-1} otherwise), such that $X_1 = \cosh(2^{-1})$, $Y_1 = \sinh(2^{-1})$, and the residual angle is θ_1. As shown in Fig. 2(b) and (c), the structure of Stage 1 has two shift units; one for cosh and the other for sinh function. The structures of sinh and cosh shift units depend on the expressions of sinh and cosh shown in columns 3 and 6 of the second row of Table II, respectively, for iteration $i = 1$. A positive input angle θ_0 implies clockwise rotation, and hence, the summation of shifted X_0 words is added (or subtracted for a negative θ_0 leading to counterclockwise rotation) to the summation of shifted Y_0 words to obtain the output X_1, while, simultaneously, the summation of shifted Y_0 words is also added (or subtracted) to the summation of shifted X_0 words to obtain the output Y_1. Fig. 2(d)–(f) depicts the above operations.

2) Stages 2 and 3: The proposed structure for the computation of Stages 2 and 3 is shown in Fig. 3. As shown in Table III, Stage 2 performs the microrotation corresponding to $i = 2$ or $i = 3$ when the input angle satisfies the condition ($\theta_1 > 2^{-4}$) or does not perform any operations and passes on the input values to Stage 3. Similarly, Stage 3 of the proposed structure performs the microrotation corresponding to $i = 4$ or $i = 5$ when the input angle satisfies the condition ($\theta_1 > 2^{-6}$) or does not perform any operations and passes on the input values to Stage 4. Note that the possible microrotations specified for each stage (Stages 2 and 3) are mutually exclusive. The simplified generalized expressions and variables value used in the structure of Stages 2 and 3 are shown in Table V.

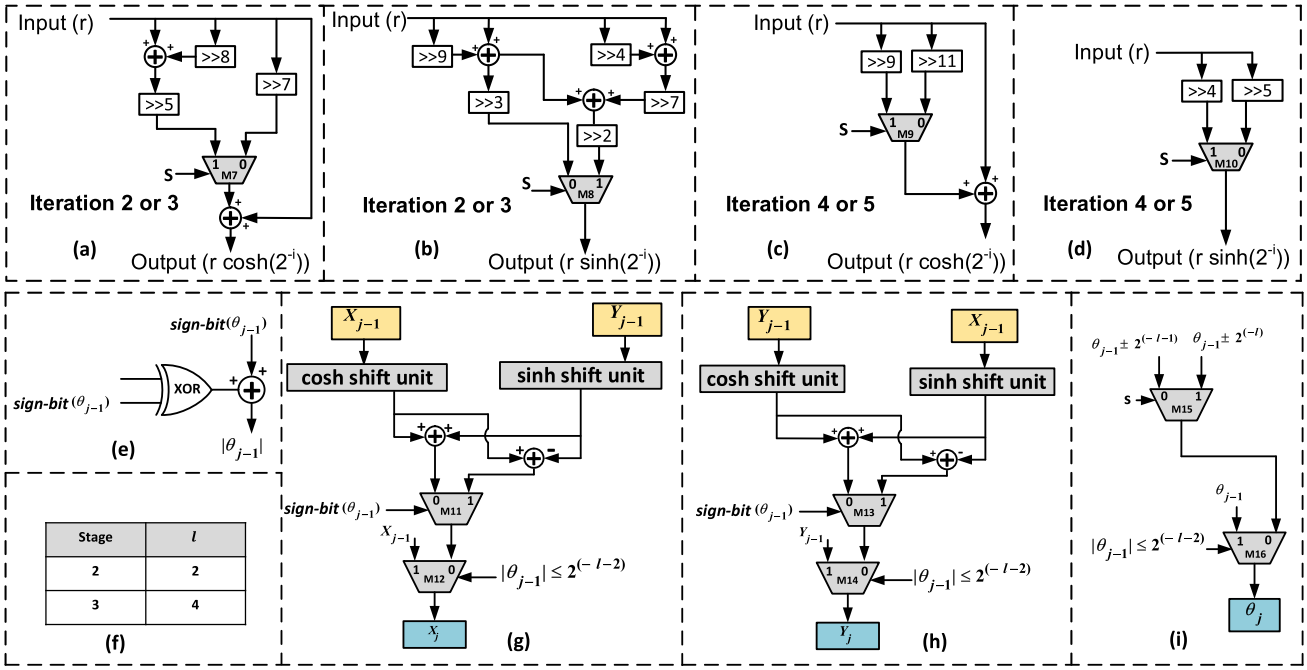


Fig. 3. Structure of Stages 2 and 3 of the proposed CORDIC architecture with shift units for 16-bit accuracy. (a) Shift unit of cosh for mutually exclusive iterations 2 and 3 for Stage 2. (b) Shift unit of sinh for mutually exclusive iterations 2 and 3 for Stage 2. (c) Shift unit of cosh for mutually exclusive iterations 4 and 5 for Stage 3. (d) Shift unit of sinh for mutually exclusive iterations 4 and 5 for Stage 3. (e) Logic circuit of computation of absolute value of θ_{j-1} using XOR gate. (f) Value of variable l for Stages 2 and 3. (g) Logic circuit of computation of output X_j. (h) Logic circuit of computation of output Y_j. (i) Logic circuit of computation of output θ_j. Here, the selection line S is $|\theta_{j-1}| > ([2^{-(l+1)} + 2^{-l}]/2)$. The structures of sinh and cosh shift units depend on the expressions of sinh and cosh shown in columns 3 and 6 of the third, fourth, fifth, and sixth rows of Table II, respectively, for iterations $i = 2$–5.

Fig. 3(a) and (b) shows the shift units of cosh and sinh, respectively, for mutually exclusive iterations 2 and 3 of Stage 2. Also, Fig. 3(c) and (d) shows the shift units of cosh and sinh, respectively, for mutually exclusive iterations 4 and 5 of Stage 3. The structures of sinh and cosh shift units depend on the expressions of sinh and cosh shown in columns 3 and 6 of the third, fourth, fifth, and sixth rows of Table II, respectively, for iterations $i = 2$–5. The absolute value of θ_{j-1} is computed by XOR gate, as shown in Fig. 3(e). Here, $j = 2$ and $j = 3$ correspond to Stages 2 and 3, respectively. To show the generalized expressions for Stages 2 and 3, a variable l is used. It needs to be noted here that for Stage 2, $l = 2$, and for Stage 3, $l = 4$, as shown in Fig. 3(f).

As shown in Fig. 3(g) and (h), a pair of inputs are fed to the 2:1 MUXes (M12 and M14) for the selection of final outputs (X_j and Y_j). If $|\theta_{j-1}| \leq 2^{-(l+2)}$, then the input values obtained from the previous stage are directly used as the output. Otherwise, the shifted words of cosh and sinh for $i = 2$ or $i = 3$ selected by 2:1 MUXes (M7 and M8) for Stage 2 and for $i = 4$ or $i = 5$ selected by 2:1 MUXes (M9 and M10) for Stage 3 are processed. The summation of shifted X_{j-1} words is added (or subtracted) to the summation of shifted Y_{j-1} words to obtain the output X_j, while the summation of shifted Y_{j-1} words is added (or subtracted) to the summation of shifted X_{j-1} words to obtain the output Y_j. In Fig. 3(i), the residual angle of the jth stage, θ_j, is computed through 2:1 MUX (M16) with selection line controlled by the condition $(\theta_{j-1} > 2^{-(l+2)})$. If the absolute value of the input angle of the stage is greater than $2^{(-l-2)}$, the rotation will be performed. Thus, if the absolute value of the input angle is greater than $[2^{-(l+1)} + 2^{-l}]/2$, then it is rotated by $\pm\, 2^{-l}$ else $\pm\, 2^{-(l+1)}$. Otherwise, the input value (θ_{j-1}) is assigned to θ_j without any rotation operation.

TABLE V

GENERALIZED EXPRESSIONS AND VARIABLES USED IN THE STRUCTURE OF STAGES 2 AND 3 AND SIMPLIFIED EXPRESSIONS

Generalized expressions and variables	Simplified expressions and variables value for Stage-2	Simplified expressions and variables value for Stage-3
j	2	3
l	2	4
θ_{j-1}	θ_1	θ_2
$\lvert\theta_{j-1}\rvert \leq 2^{(-l-2)}$	$\lvert\theta_1\rvert \leq 2^{(-4)}$	$\lvert\theta_1\rvert \leq 2^{(-6)}$
$\lvert\theta_{j-1}\rvert > ([2^{-(l+1)} + 2^{-l}]/2)$ (S)	$\lvert\theta_1\rvert > 0.1875$ (S)	$\lvert\theta_2\rvert > 0.0469$ (S)

3) Stage 4: It is the final stage of the proposed CORDIC architecture to implement a small microrotation through an angle α_k, given by

$$\alpha_k = k \times M \tag{16}$$

where $k = \{0, 1, 2, \ldots, 8\}$ and $M \simeq 2^{-9}$. Since α_k is very small, the mathematical expression for sinh and cosh functions can be given by

$$X_j = X_{j-1} + d \cdot k \times [2^{-9} \cdot Y_{j-1}] \tag{17a}$$

$$Y_j = Y_{j-1} + d \cdot k \times [2^{-9} \cdot X_{j-1}] \tag{17b}$$

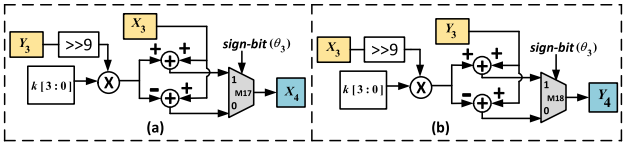


Fig. 4. Structure of small microrotations stage. (a) Computation of output X_4. (b) Computation of output Y_4. Here, the value of $k[3:0] = |\theta_3|$[MSB:MSB-3] + $|\theta_3|$[MSB-4].

where $j = 4$ (corresponding to Stage 4) and d is the direction of angle rotation. As shown in Fig. 4, (17) is implemented in the proposed architecture using two multipliers, two pairs of add/subtract, and two 2:1 MUXes (M17 and M18). The $Y_3(X_3)$ is right-shifted by 9-bit positions and then multiplied with k. The addition of the fifth MSB with the first four MSBs of the absolute value of the input angle provides the value of k to be used for multiplication. In Fig. 4(a), the final value of X_4 is computed by subtracting the summation of shifted Y_3 from X_3 if θ_3 is negative, else they are added to each other, while, simultaneously, the summation of shifted X_3 word is added with (or subtracted from) Y_3 for the computation of Y_4, as shown in Fig. 4(b). The outputs of this stage, X_4 and Y_4, are the desired $\cosh\theta$ and $\sinh\theta$, respectively.

IV. Error Performance of the Proposed Architecture

There are two types of error introduced in the rotation mode hyperbolic CORDIC architecture: rotational and residual errors. In this section, we have analyzed both these types of errors.

A. Residual Error (R_{err})

The values of cosh and sinh functions are given by the rotated vectors, X and Y, respectively. The rotation in hyperbolic trajectory can be represented as follows:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cosh(\theta) & \sinh(\theta) \\ \sinh(\theta) & \cosh(\theta) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{18}$$

Ideally, the input vector should be rotated by angle θ, but due to the residual angle, the input vector can only be rotated by $(\theta - \theta_r)$. The values of cosh and sinh functions are affected by the residual angle and are represented by X_r and Y_r, respectively

$$\begin{bmatrix} X_r \\ Y_r \end{bmatrix} = \begin{bmatrix} \cosh(\theta - \theta_r) & \sinh(\theta - \theta_r) \\ \sinh(\theta - \theta_r) & \cosh(\theta - \theta_r) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{19}$$

where θ_r can be positive or negative. The residual error is given by

$$R_{\text{err}} = \begin{bmatrix} X_{\text{err}} \\ Y_{\text{err}} \end{bmatrix} = \begin{bmatrix} X_r \\ Y_r \end{bmatrix} - \begin{bmatrix} X \\ Y \end{bmatrix}. \tag{20}$$

B. Rotational Error (Ro_{err})

The rotational error introduced in each of the pipeline stages of the proposed architecture due to the approximations in Taylor series expansion can be formulated as follows:

$$Ro_{\text{err}} = \sum_i x_{R_{\text{err}}} \text{ or } y_{R_{\text{err}}}. \tag{21}$$

For the proposed fifth-order and fourth-order Taylor series approximation of sinh and cosh, respectively, the rotational errors introduced in X and Y coordinates are given by $x_{R_{\text{err}}}$ and $y_{R_{\text{err}}}$, respectively

$$T_c = \sum_{n=1}^{n=2} \left[\sum_p 2^{-(i(2n)+c_{pi})} \right] \tag{22}$$

$$T_s = \sum_{n=1}^{n=2} \left[\sum_p 2^{-(i(2n+1)+s_{pi})} \right]. \tag{23}$$

Then

$$x_{R_{\text{err}}} = [\{1 + T_c\}X_j + \{2^{-i} + T_s\}Y_j] - X_{i+1} \tag{24}$$

$$y_{R_{\text{err}}} = [\{2^{-i} + T_s\}X_j + \{1 + T_c\}Y_j] - Y_{i+1} \tag{25}$$

where X_{i+1} and Y_{i+1} value refers to (3). The values of c_{pi} and s_{pi} depend on the accuracy requirement.

C. Total Error (To_{err})

The total approximation error of the proposed architecture is the sum of rotational error and residual error

$$To_{\text{err}} = Ro_{\text{err}} + R_{\text{err}}. \tag{26}$$

Here, the rotational and residual error both can be positive or negative. The truncation error of the architecture is generally introduced when the number of bits used is less than the number of bits required to represent the value. The fractional part of the value creates truncation error, and it can also affect the residual and rotational errors. Truncation error for the fractional part can be formulated as follows:

$$T_{\text{err}} = |(2^{-1} + 2^{-2} + \cdots 2^{-f}) - (2^{-1} + 2^{-2} + \cdots 2^{-n_f})| \tag{27}$$

where n_f = number of required fractional bits and f = number of used fractional bits.

D. Comparison of Accuracy Performance

The comparison of accuracy of the proposed design for input range $(-\pi/4, \pi/4)$ with the existing architectures of sinh and cosh functions is shown in Table VI. Fig. 5 shows total error of sinh and cosh functions of the proposed and existing architectures for input range $(0, \pi/4)$.

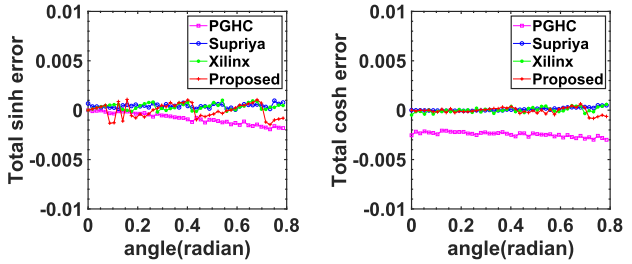
For 8-bit word length, the maximum normalized error of sinh function of the architecture [33] is less than the proposed architecture. However, the implementation of the architecture of [33] with eight I/O bits takes eight clock cycles (Cc) plus two more Cc (since iteration $i = 4$ needs to be repeated for convergence) for the first output, whereas the proposed architecture requires only three Cc for the first output. Note that Stage 4 of the proposed architecture is not required for 8-bit word length. The maximum normalized error of the proposed architecture for 8-bit word length is 95.65% and 65.59% less than the architecture [29] for sinh and cosh function, respectively, because the architecture [29] has a significantly large truncation error than the proposed architecture in each iteration.

TABLE VI

COMPARISON OF ERROR WITH THE EXISTING ARCHITECTURES OF sinh AND cosh FUNCTIONS [INPUT RANGE = $(-\pi/4, \pi/4)$]

Architecture	For 8-bit word-length		For 16-bit word-length	
	Sinh (Error*)	Cosh (Error*)	Sinh (Error*)	Cosh (Error*)
Xilinx IP [33]	0.046	0.015	0.00177	0.0044
PGHC [31],[32]	0.082	0.036	0.0024	0.0023
Aggarwal [29]	1.29	0.044	0.0012	0.0004
Proposed	0.056	0.015	0.0018	0.0006

*Error = Maximum normalized error
Normalized Error = |(computed value - actual value)/actual value|. RoC of the Xilinx IP and PGHC restricted to the input range $(-\pi/4, \pi/4)$ and (-1.18, 1.18), respectively.



(a) (b)

Fig. 5. Total error for input range $(0, \pi/4)$ for 16-bit word length. (a) Sinh error. (b) Cosh error.

For 16-bit word length, the error performance of the architecture [29] is marginally better than that of the proposed architecture and is the best among the existing designs. The architecture proposed in [31], [32], and [33] is, however, restricted to the input range $(-\pi/4, \pi/4)$. The RoC of [29] can be extended by storing some known values of hyperbolic functions in a ROM. But, the design method proposed in [29] involves significantly more area complexity. Implementing architecture of [33] with 16 I/O bits will take 16 Cc plus two more Cc (since iterations $i = 4$ and $i = 13$ need repetition for convergence) for the first output. The pipelined generalized hyperbolic architecture CORDIC (PGHC) [31], [32] is based on the conventional hyperbolic CORDIC algorithm (with scaling). The maximum input angle to this architecture is $\sim$1.18 (radian) [27]. The maximum normalized errors of sinh and cosh functions of the proposed design are slightly higher than that of [33] and [29] but involves much lower latency of computation. Moreover, it is shown later in this article (in Section V) that the proposed architecture has a much better area-delay, power-delay and accuracy performance trade-off.

V. FPGA IMPLEMENTATION AND ASIC SYNTHESIS

The proposed architecture and all other compared architectures are coded in Verilog hardware description language (HDL), simulated using Xilinx Vivado by applying fixed-point inputs, and implemented in FPGA and ASIC platforms. The details of estimated performance are discussed in this section. Note that FPGA and ASIC post-layout results of all the designs are carried out at a clock frequency of 50 MHz.

TABLE VII

COMPARISON OF FPGA IMPLEMENTATION RESULTS FOR INPUT RANGE $(-\pi/4, \pi/4)$, NUMBER OF I/O BITS = 16

Xilinx Zedboard (Zynq-7000)					
Resources	Xilinx IP [33]	PGHC [31],[32]	Aggarwal [29]	Proposed	Improvement
Slice Registers	1165	749	572	198	65.38%
Slice LUTs	1087	725	3893	562	22.48%
Occupied Slices	354	200	1126	177	11.5%
MUF (MHz)	200	200	83.33	136.98	-
*Latency (Cc)	18	16	11	4	63.63%
*Delay (ns)	90	80	132	29.2	63.5%
Dynamic Power (mW)	49	51	104	25	48.97%
EPO (pJ)	1515	1535	3384	1270.2	16.16%
EEP	0.214	0.277	0.369	0.66	$\sim 1.8\times$
SDP ($\times 10^3$) (ns)	31.86	16.00	148.63	5.17	67.69%
LDP ($\times 10^3$) (ns)	97.83	58.00	513.88	16.41	71.71%
RDP ($\times 10^3$) (ns)	104.85	59.92	75.50	5.782	90.35%
EAP	0.916	2.128	1.110	4.708	$\sim 2.21\times$

*We have calculated the latency as the number clock cycles required to produce the output. However, we have estimated the delay as the product of number of clock cycles and minimum clock period of the architecture.

TABLE VIII

COMPARISON OF FPGA IMPLEMENTATION RESULTS FOR INPUT RANGE $(-\pi, \pi)$, NUMBER OF I/O BITS = 21

Xilinx Zedboard(Zynq-7000)			
Resources	Aggarwal [29]	Proposed	Improvement
Slice Registers	671	217	67.66%
Slice LUTs	4946	861	82.59%
Occupied Slices	1403	259	81.54%
MUF (MHz)	83.33	117.65	41.18%
Latency (Cc)	11	4	63.64%
Delay (ns)	132	34	74.24%
Dynamic Power (mW)	136	33	75.73%
EPO (pJ)	4092	1589.5	61.15%
EEP	0.196	0.629	$\sim 3.21\times$
SDP ($\times 10^3$) (ns)	185.19	8.81	95.24%
LDP ($\times 10^3$) (ns)	652.87	29.27	95.52%
RDP ($\times 10^3$) (ns)	88.57	7.38	91.67%
EAP	0.57	3.86	$\sim 6.77\times$

The abbreviations of terminologies used in different tables of comparison is specified in the Nomenclature.

A. FPGA Implementation

The RTL code of the proposed design is synthesized by Xilinx Vivado 2019.2 and implemented on Xilinx Zedboard (xc7z020clg484-1). The performance of the proposed architecture is compared with the existing architectures for input angle range $(-\pi/4, \pi/4)$, as shown in Table VII. The designs of [31], [32], and [33] are based on conventional hyperbolic CORDIC with scaling. Both of these designs have the same clock period, which is the minimum of all designs. Accordingly, these designs offer the highest MUF. But, the designs of PGHC [31], [32] and Xilinx IP [33] involve the maximum latency of 16 and 18 iterations, respectively, and involve the maximum processing delay. The proposed design has the minimum latency of four Cc and involves the minimum

processing delay among all the designs. In this article, we have calculated the latency as the number Cc required to produce the output. However, we have estimated the delay as the product of number of Cc and minimum clock period of the architecture.

The proposed design occupies significantly less number of registers and lookup table (LUT) resources. Accordingly, it occupies significantly less number of slices. Compared with the design in [31] and [32], the proposed architecture consumes 22.48% less LUTs, and the design of [31] and [32] occupies ∼13% more slices compared with the proposed architecture. The designs of [29], [31], and [32] involve nearly three times and four times the number of slice registers of those required by the proposed architecture, respectively. In Tables VII and VIII, only dynamic power is reported, since the static power of all the architectures is almost the same. The reported dynamic power is the onboard power consumption of the design, which is obtained after implementation. However, EPO is computed by taking the total power consumption (including both static and dynamic power) of the designs reported at the highest frequency. The proposed architecture consumes significantly less dynamic power compared with the others. It has significantly lower SDP and LDP than the existing designs. The EPO could be computed as the product of power dissipation (total power) at minimum clock period, and the duration of minimum clock period is the lowest among all the designs. The design of [31] and [32] involves ∼20.85% more EPO compared with the proposed one. In addition, the EEP and EAP are calculated by reciprocal of the product of average of the maximum normalized error of sinh and cosh functions with the EPO and the number of occupied slices, respectively. The proposed architecture has better EEP and EAP over the other designs.

The designs proposed in [31], [32], and [33] are only applicable for input range $(-\pi/4, \pi/4)$. Therefore, the design in [29] is only compared with proposed architecture for input range $(-\pi, \pi)$ by taking fractional bits $f = 16$, as shown in Table VIII. In this case, the proposed design has fewer FPGA resources requirement, less power consumption, and less delay over the design in [29].

B. ASIC Post-Layout Results

The Verilog code of the proposed design and other existing reported designs are synthesized using Synopsys Design Compiler and place and route (PnR) using Synopsys IC Compiler. The synthesis and PnR of the proposed architecture as well as compared architectures are performed by using the 1-V Taiwan Semiconductor Manufacturing Company (TSMC) 65-nm 1P9M CMOS process. The post-layout results of the proposed architecture and other existing architecture for input range of $(-\pi/4, \pi/4)$ and $(-\pi, \pi)$ are shown in Tables IX and X, respectively.

As shown in Table IX for the input range $(-\pi/4, \pi/4)$, the design of PGHC [31], [32] has the highest MUF among all the existing designs. Accordingly, it offers the minimum clock period. But, the designs of [29] and [32] have much higher latency than the proposed design and involve higher processing delay. The proposed design has 68.92% less power consumption than that of the design in [29], whereas the EPO of the proposed design is ∼46.89% less than the design of [29] with nearly 1.25 times better EEP. The proposed design involves significantly less area and computational delay than the existing designs, because the architecture has fewer pipelined stages. The designs proposed in [31] and [32] are only applicable for input range $(-\pi/4, \pi/4)$. Therefore, the proposed architecture is only compared with the design of [29] for the input range $(-\pi, \pi)$ in Table X. The design of [29] has higher latency and less MUF than the proposed design. The proposed design has 51.51% less power consumption and 56.09% less EPO than the design of [29] with nearly 2.84 times better EEP.

TABLE IX

ASIC POST-LAYOUT RESULTS COMPARISON WITH THE EXISTING ARCHITECTURES FOR INPUT RANGE $(-\pi/4, \pi/4)$ IN THE TSMC 65-nm PROCESS, NUMBER OF I/O BITS = 16

Parameters	Aggarwal [29]	PGHC [31],[32]	Proposed	Improvement
Area (μm^2)	33696	19835	7982	76.31%
MUF (MHz)	476.19	833.33	666.66	-
Latency (Cc)	11	16	4	63.63%
Delay (ns)	23.1	19.2	6	68.75%
Power (mW)	0.5982	0.5686	0.1767	68.92%
EPO (pJ)	7.23	8.63	3.84	46.89%
EEP	172.89	49.31	217.01	∼1.25×
ADP ($mm^2 \times ns$)	0.778	0.171	0.0479	71.99%
EAP	0.0371	0.0215	0.1044	∼2.8×

TABLE X

ASIC POST-LAYOUT RESULTS COMPARISON WITH THE EXISTING ARCHITECTURES FOR INPUT RANGE $(-\pi, \pi)$ IN THE TSMC 65-nm PROCESS, NUMBER OF I/O BITS = 21

Parameters	Aggarwal [29]	Proposed	Improvement
Area (μm^2)	39721	12196	69.29%
MUF (MHz)	454.54	625	37.50%
Latency (Cc)	11	4	63.64%
Delay (ns)	13.2	6.4	72.73%
Power (mW)	0.61	0.25	51.51
EPO (pJ)	8.7	3.82	56.09
EEP	91.95	261.78	∼2.84×
ADP ($mm^2 \times ns$)	0.524	0.0781	85.09%
EAP	0.020	0.082	4.1×

C. Implemented Chip of the Proposed Architecture

The proposed architecture for the proof of concept is implemented and verified on a silicon chip for the input range $(-\pi, \pi)$ in the 1.8-V TSMC 180-nm 1P6M CMOS process. The fabricated chip microphotograph and its testing setup are shown in Fig. 6. The hyperbolic functions implemented in the chip are sinh, cosh, and tanh. The implemented tanh function is the division of sinh and cosh functions. The power consumption of the design is 3.24 mW at 100-MHz frequency. The reported power of the chip also includes shift registers power and divider block power with the proposed hyperbolic CORDIC block power. We have used one serial-in parallel-out shift register to provide inputs to the CORDIC block, one divider block for the tanh function, and two

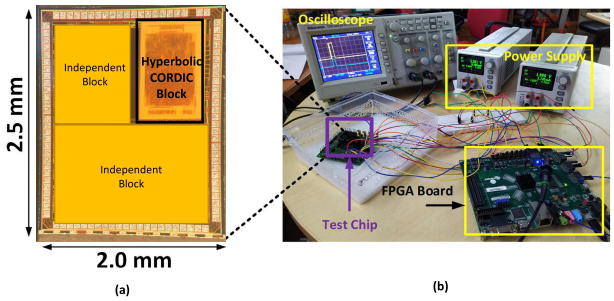


Fig. 6. (a) Chip microphotograph. (b) Chip testing setup for measurement.

TABLE XI
CHIP CHARACTERISTICS OF THE PROPOSED ARCHITECTURE IN THE TSMC 180-nm PROCESS

Process Technology	TSMC 180nm
Supply Voltage	1.8 V
Power consumption	3.24mW@100MHz
Core Area	0.12 mm^2
Input range	$(-\pi, \pi)$
Implemented Hyperbolic functions	sinh, cosh, and tanh

parallel-out serial-in shift registers to achieve outputs from the chip. The core area consumed by the proposed design block is 0.12 mm^2. The reported area of the chip includes the proposed hyperbolic CORDIC block area, shift registers area, and divider block area. The power and area consumption obtained by ASIC post-layout results of the proposed design in TSMC 180-nm process for input range $(-\pi, \pi)$ are ≈2.7 mW at 100 MHz and 0.12 mm^2, respectively. Note that the ASIC post-layout results in Section V-B, which we have discussed and compared with the existing architectures in TSMC 65-nm process, include only the proposed hyperbolic CORDIC block characteristics (it does not include any shift registers and divider block consumption). The chip characteristics of the proposed architecture are shown in Table XI.

VI. CONCLUSION

We have proposed an improved Taylor series approximation to compute hyperbolic trigonometric functions using a scaling-free hyperbolic CORDIC algorithm with less computational complexity and mapped that to a folded-pipeline architecture. The proposed structure for hyperbolic CORDIC has significantly lower latency and less hardware complexity compared with the existing architectures, with acceptable accuracy. The reduction of latency and hardware complexity is achieved by the reduction of number of microrotations to achieve the desired accuracy, and folding of two pairs of microrotations in two pipeline stages without impacting the delay and clock period with simple hardwired control. The proposed architecture could also be extended to include more number of stages to further improve the accuracy when required. For FPGA and ASIC implementations, the proposed architecture consumes significantly less FPGA logic resources and less area in case of ASIC, and it involves lower latency, less power consumption, and less EPO compared with the best of the existing architectures. The proposed architecture is also implemented and verified on a silicon chip for the input range $(-\pi, \pi)$ for the proof of concept.

ACKNOWLEDGMENT

This publication is an outcome of the work undertaken in a project under the Visvesvaraya Ph.D. Scheme of Ministry of Electronics and Information Technology, Government of India, being implemented by Digital India Corporation, New Delhi, India (formerly Media Lab Asia). Also, this work has been carried out in collaboration with Sandhaan Labs Private Limited, Bhubaneswar, Odisha, India (https://sandhaanlabs.in).

REFERENCES

[1] A. H. Bell, *The Exponential Hyperbolic Functions and Their Applications*. London, U.K.: Pitman, 1952.
[2] J.-M. Müller, *Elementary Functions: Algorithms and Implementation*. Boston, MA, USA: Birkhäuser, 1997.
[3] D. Zhang and A. K. Jain, "Biometric authentication," in *Proc. Int. Conf. Biometric Authentication*, Hong Kong, 2004, Jul. 2004.
[4] A. Boudabous, F. Ghozzi, M. W. Kharrat, and N. Masmoudi, "Implementation of hyperbolic functions using CORDIC algorithm," in *Proc. 16th Int. Conf. Microelectron.*, Tunis, Tunisia, 2004, pp. 738–741.
[5] B. Zamanlooy and M. Mirhassani, "Efficient VLSI implementation of neural networks with hyperbolic tangent activation function," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 1, pp. 39–48, Jan. 2014.
[6] Y.-H. Chen, S.-W. Chen, and M.-X. Wei, "A VLSI implementation of independent component analysis for biomedical signal separation using CORDIC engine," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 2, pp. 373–381, Apr. 2020.
[7] H. de Lassus Saint-Geniès, D. Defour, and G. Revy, "Exact lookup tables for the evaluation of trigonometric and hyperbolic functions," *IEEE Trans. Comput.*, vol. 66, no. 12, pp. 2058–2071, Dec. 2017.
[8] L. Huai, P. Li, G. E. Sobelman, and D. J. Lilja, "Stochastic computing implementation of trigonometric and hyperbolic functions," in *Proc. IEEE 12th Int. Conf. ASIC (ASICON)*, Guiyang, China, Oct. 2017, pp. 553–556.
[9] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, Sep. 1959.
[10] 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.
[11] B. Gisuthan and T. Srikanthan, "Flat CORDIC: A unified architecture for high-speed generation of trigonometric and hyperbolic functions," in *Proc. 43rd IEEE Midwest Symp. Circuits Syst.*, Lansing, MI, USA, Aug. 2000, pp. 1414–1417.
[12] T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai, "Para-CORDIC: Parallel CORDIC rotation algorithm," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 8, pp. 1515–1524, Aug. 2004.
[13] Y. H. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithm implementation," *IEEE Trans. Comput.*, vol. 42, no. 1, pp. 99–102, Jan. 1993.
[14] C. Wang, J. Luo, and J. Zhou, "A 1-V to 0.29-V sub-100-pJ/operation ultra-low power fast-convergence CORDIC processor in 0.18-μm CMOS," *Microelectron. J.*, vol. 76, pp. 52–62, Jun. 2018.
[15] J. Wu et al., "Efficient design of spiking neural network with STDP learning based on fast CORDIC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 6, pp. 2522–2534, Jun. 2021.
[16] C.-S. Wu and A.-Y. Wu, "Modified vector rotational CORDIC (MVR-CORDIC) algorithm and architecture," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 6, pp. 548–561, Jun. 2001.
[17] C.-S. Wu, A.-Y. Wu, and C.-H. Lin, "A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 50, no. 9, pp. 589–601, Sep. 2003.
[18] M. G. B. Sumanasena, "A scale factor correction scheme for the CORDIC algorithm," *IEEE Trans. Comput.*, vol. 57, no. 8, pp. 1148–1152, Aug. 2008.

[19] C.-H. Lin and A.-Y. Wu, "Mixed-scaling-rotation CORDIC (MSR-CORDIC) algorithm and architecture for high-performance vector rotational DSP applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 11, pp. 2385–2396, Nov. 2005.

[20] J. Villalba, J. A. Hidalgo, E. L. Zapata, E. Antelo, and J. D. Bruguera, "CORDIC architectures with parallel compensation of the scale factor," in *Proc. Int. Conf. Appl. Specific Array Processors*, Strasbourg, France, Jul. 1995, pp. 258–269.

[21] F. J. Jaime, M. A. Sanchez, J. Hormigo, J. Villalba, and E. L. Zapata, "Enhanced scaling-free CORDIC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 7, pp. 1654–1662, Jul. 2010.

[22] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, "Modified virtually scaling-free adaptive CORDIC rotator algorithm and architecture," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 11, pp. 1463–1474, Nov. 2005.

[23] M. Garrido, P. Källström, M. Kumm, and O. Gustafsson, "CORDIC II: A new improved CORDIC algorithm," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 2, pp. 186–190, Feb. 2016.

[24] S. S. Wadkar, B. P. Das, and P. K. Meher, "Low latency scaling-free pipeline CORDIC architecture using augmented Taylor series," in *Proc. IEEE Int. Symp. Smart Electron. Syst. (iSES)*, Rourkela, India, Dec. 2019, pp. 312–315.

[25] R. Wu, H. Chen, G. He, Y. Fu, and L. Li, "Low-latency low-complexity method and architecture for computing arbitrary Nth root of complex numbers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 6, pp. 2529–2541, Jun. 2022.

[26] Q. Zhu, Z. Zhao, K. Mao, X. Chen, W. Liu, and Q. Wu, "A real-time hardware emulator for 3D non-stationary U2V channels," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 9, pp. 3951–3964, Sep. 2021.

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

[28] 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.

[29] 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.

[30] J. Sudha, M. C. Hanumantharaju, V. Venkateswarula, and H. Jayalaxm, "A novel method for computing exponential function using CORDIC algorithm," *Proc. Eng.*, vol. 30, pp. 519–528, Jan. 2012.

[31] Y. Luo, Y. Wang, Y. Ha, Z. Wang, S. Chen, and H. Pan, "Generalized hyperbolic CORDIC and its logarithmic and exponential computation with arbitrary fixed base," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 9, pp. 2156–2169, Sep. 2019.

[32] 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, Exp. Briefs*, vol. 67, no. 11, pp. 2652–2656, Nov. 2020.

[33] *Xilinx Logic Core Product Specifications CORDICv3.0 DS249*, Xilinx, San Jose, CA, USA, 2005.

[34] A. Verma, P. Sharma, and B. P. Das, "RISC-V core with approximate multiplier for error-tolerant applications," in *Proc. 25th Euromicro Conf. Digit. Syst. Design (DSD)*, Maspalomas, Spain, Aug. 2022, pp. 239–246.

Anu Verma received the M.Tech. degree in VLSI design from Banasthali University, Rajasthan, India, in May 2018. She is currently working toward the Ph.D. degree in VLSI design at the Department of Electronics and Communication Engineering, Indian Institute of Technology (IIT) Roorkee, Roorkee, India.

Her current research interests include digital signal processing, low-power digital design, and high-performance processors for the internet-of-things (IoT) applications.

Khyati Kiyawat received the B.Tech. degree in electronics and communication engineering from the Indian Institute of Technology (IIT) Roorkee, Roorkee, India, in 2020. She is currently working toward the Ph.D. degree in computer architecture at the Computer Science Department, University of Virginia, Charlottesville, VA, USA.

Her research interests include energy-efficient computing, domain-specific accelerators, and processing-in-memory architectures.

Bishnu Prasad Das (Senior Member, IEEE) received the Ph.D. degree in electronics design and technology from the Indian Institute of Science (IISc), Bangalore, India, in 2009.

He was with Texas Instruments, Bengaluru, India, under Texas Instruments University Program during his Ph.D. program. From 2012 to 2013, he worked as a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. From 2009 to 2012, he worked as a Post-Doctoral Researcher with Kyoto University, Kyoto, Japan. He is currently an Associate Professor with the Department of Electronics and Communication Engineering, IIT Roorkee, India. His current research interests include in-memory computation architectures, hardware security, reduced instruction set computer (RISC)-V processor architecture, wearable healthcare devices, and ultralow-power circuit design.

Dr. Das was a recipient of the Young Faculty Research Fellowship from the Ministry of Electronics and Information Technology (MeitY), Government of India, in 2018. He was awarded the Faculty Fellow from Divyasampark iHUB Roorkee, India, in 2021.

Pramod Kumar Meher (Senior Member, IEEE) received the B.Sc. (Hons.) and M.Sc. degrees in physics and the Ph.D. degree in science from Sambalpur University, Sambalpur, India, in 1976, 1978, and 1996, respectively.

He was a Reader in electronics with Berhampur University, Berhampur, India, from 1993 to 1997, and a Professor of Computer Applications with Utkal University, Bhubaneswar, India, from 1997 to 2002. He was holding senior research positions with the School of Computer Engineering, Nanyang Technological University, Singapore, from 2004 to 2016. He is currently a Senior Professor with C. V. Raman Global University, Bhubaneswar. He has contributed nearly 250 technical papers to various reputed journals and conference proceedings, including nearly 90 articles in various IEEE TRANSACTIONS. He has co-edited the book *Arithmetic Circuits for Digital Signal Processing (DSP) Applications* (Wiley–IEEE Press). His current research interests include signal processing, cyber security, intelligent computing for smart IoT applications, edge computing, and analytics.

Dr. Meher was elected as a fellow of the Institution of Electrical Engineers, U.K., in 2004. He is a fellow of the Institution of Electronics and Telecommunication Engineers, India. He was a recipient of the Samanta Chandrasekhar Award for excellence in research on engineering and technology in 1999. He was a speaker of the Distinguished Lecturer Program of the IEEE Circuits Systems Society from 2011 to 2012. He has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I, and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS. He is serving as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: and the *International Journal of Circuits, Systems, and Signal Processing*.