Design and Implementation of a Low Power Area Efficient Bfloat16 based CORDIC Processor

Saras Mani Mishra*, Hanumant Singh Shekhawat, Gaurav Trivedi
Department of Electronics and Electrical Engineering,
Indian Institute of Technology Guwahati, Guwahati, India
Email: ***m.saras@iitg.ac.in***

Pidanic Jan, Zdenek Nemec
Department of Electrical Engineering,
University of Pardubice, Pardubice, Czech Republic

***Abstract*—Coordinate Rotation Digital Computer (CORDIC) algorithm has a great advantage in hardware based implementation because of its simple architecture. It employs shifter and adder for hardware implementation. The major issue with a CORDIC algorithm is the linear dependence of convergence on the number of iterations. Each iteration performs shift and addition or subtraction operations, due to this there is a trade off between area and delay. Also, the floating-point representation of angles would also increase the area and power. The main aim of this work is to implement a low power and area efficient bfloat16 based on a CORDIC algorithm. The proposed hardware module consumes $3.2\times$ and $3.38\times$ less area and power compared to a single-precision floating-point based CORDIC implementation. The result of the proposed module has been verified on a Zynq evaluation FPGA board.**

***Index Terms*—Trigonometric function, Coordinate Systems, Bfloat16, CORDIC, Floating-point Representation**

I. Introduction

With the increase in advancement and dependency on technology, the need of complex computation is also increasing. Most of these complex computations involve nonlinear operations, which should be done quickly and efficiently to match the requirements of this era. Therefore, it is important to develop a hardware module, which can perform these operations by consuming less resources. Exponential, logarithmic, trigonometric, and square-roots are some frequently used nonlinear operations in advanced systems. These operations can be performed by implementing a Taylor series on hardware at the cost of higher area and high power consumption [13] [14]. There is an alternative for this approach, which is the Coordinate Rotation Digital Computer (CORDIC) algorithm, proposed by Jack E. Volder [1] in 1959.

Traditionally, a CORDIC algorithm was implemented by employing bit serial architecture and all the iterations were executed in bits [1]. The serial bit approach slows down the computation and increases the delay of the module. The serial bit implementation approach is also known as the folded architecture approach. Another approach [2] is known as the unfolded architecture approach which provides pipeline and parallel architectures. The radix-n approach is a pipeline architecture methodology and generally it is employed for a fixed-point format [2]. Pipelined implementation has been introduced in [3] [4] to increase the efficiency and throughput. The work proposed in [4] [5] [9] shows that CORDIC architecture employed vector inputs in floating-point representation and angles are stored in look-up tables (LUT) using fixed-point representation. Lee et al. [11] have implemented a complete floating-point based CORDIC architecture.

CORDIC processors are developed to compute the functions in real time for digital computers [8]. The real time computation is required for applications such as signal processing, image processing, digital communication, robotics [10] etc. It can function as a single unit, which includes a large set of applications such as, trigonometric, exponential, and arithmetic operations.

With time, a lot of modification has been made to this algorithm and improved its performance. It is an iterative algorithm, to calculate the rotation of two vectors in circular, linear and hyperbolic coordinate systems. To calculate the trigonometric, exponential, and arithmetic operations, circular, hyperbolic, and linear modes are selected, respectively. Look-up table, adder, subtracter and shifter are used in this algorithm to perform computation in hardware. Low cost, less hardware requirement and simple hardware implementation are some major advantages of a CORDIC processor.

In this paper, we have designed a CORDIC algorithm based module that performs trigonometric and exponential functions efficiently, due to the reduced mantissa bits of bfloat16 floating-point representation. The proposed module has been implemented in bfloat16 floating-point representation as it consumes less hardware [6]. The bfloat16 based module consumes less hardware resources compared to single-precision floating-point representation. The module has been compared with a single precision based CORDIC processor.

The paper is organized as follows: the related work of the proposed module is explained in section II. In section III, the methodology of proposed work is explained. The results and discussions are explained in section IV. The proposed work has been concluded with the future work in section V.

II. Related Work

We briefed about the floating-point representations and mathematical representation of a CORDIC processor.

A. Floating-point representation

While designing VLSI architecture, we have to place the computation components and the required memory blocks in a minimum area. The fixed number of bits limit the infinite

precision in binary or fixed-point representation. The floating-point representation is given to address this issue by depicting the huge range in limited bit width. The base, significand and exponent are used to represent the floating-point number as: $significand \times base^{exponent}$. In 1985, the Institute of Electrical and Electronics Engineering (IEEE) has given a standard to represent floating-point number known as IEEE754 in the range of $\sim 1e^{-38}$ to $\sim 3e^{38}$ for 32-bit representation. There are 64-bit, and 128-bit types of IEEE754 representations also available. This shows a different range with a different bit width of exponent and mantissa. Single-precision is the most commonly used floating-point number, which is a 32-bit representation with 1-bit for sign, 8-bit exponent and 23-bits are reserved for the mantissa.

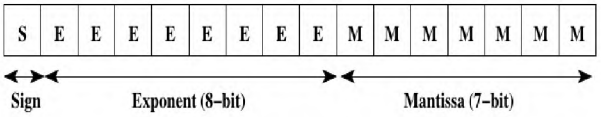

Fig. 1. Bfloat16 representation.

Recently, a Google research group of artificial intelligence (Google Brains) [7] has proposed a slightly modified version of single-precision known as bfloat16. A brain floating-point or bfloat16 is a 16-bit floating-point number with 1-bit for sign, 8-bit exponent and 7-bits are reserved for mantissa. The range of bfloat16 representation is the same as single-precision. The value of single-precision and bfloat16 number is calculated through the same equation (1).

$$Value = (-1)^S \times 2^{E-127} \times (1.M) \tag{1}$$

where, S, E and M are sign, exponent and mantissa of floating-point number, respectively.

B. CORDIC Algorithm

The CORDIC is a simple iterative convergence algorithm that offers low hardware cost. It utilizes look-up tables, shift, add and subtract operators to perform the hardware operation. General equations of the CORDIC algorithm [2] are as follows:

$$x_{n+1} = x_n cos\Omega - y_n sin\Omega \tag{2}$$
$$y_{n+1} = y_n cos\Omega + x_n sin\Omega \tag{3}$$

It can be re-written as follows:

$$x_{n+1} = cos\Omega(x_n - y_n tan\Omega) \tag{4}$$
$$y_{n+1} = cos\Omega(y_n + x_n tan\Omega) \tag{5}$$

Where $tan\Omega$ can be approximated as $tan\Omega = \pm 2^{-n}$ and n is the number of iterations. The multiplication operation is replaced by a simple shifter operator. Hence equation (4) and (5) are modified as follow:

$$x_{n+1} = K(x_n - y_n d_n 2^{-n}) \tag{6}$$
$$y_{n+1} = K(y_n + x_n d_n 2^{-n}) \tag{7}$$

To avoid the complexity in the calculation, we multiply the scaling factor (K) in the last stage of iteration. It is equals to $1/G$, where G is the gain. The accuracy of G depends on iterations and is limited to 1.64676.

$$G = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}} \tag{8}$$

As the number of iterations increase, K approaches towards 0.607252935. Hence, the modified CORDIC equations can be depicted as

$$x_{n+1} = x_n - y_n d_n 2^{-n} \tag{9}$$

$$y_{n+1} = y_n + x_n d_n 2^{-n} \tag{10}$$

$$z_{n+1} = z_n - d_n \alpha \tag{11}$$

$$where, \quad d_n = \begin{cases} -1, & z_n < 0 \\ +1, & otherwise \end{cases}$$

To generalize, the above equations for all three modes equation (9), (10) and (11) can be written as

$$x_{n+1} = x_n - m y_n d_n 2^{-n} \tag{12}$$
$$y_{n+1} = y_n + x_n d_n 2^{-n} \tag{13}$$
$$z_{n+1} = z_n - d_n \alpha \tag{14}$$

Where m is the mode selection for the coordinate system and α is the set of LUTs for different coordinates as shown in Table I.

TABLE I
MODE SELECTION AND LUTS FOR DIFFERENT COORDINATES

Mode (m)	Coordinate system	Value of α
$+1$	Circular	$\tan^{-1}(2^{-n})$
0	Linear	2^{-n}
-1	Hyperbolic	$\tanh^{-1}(2^{-n})$

Where $d_n = sign(z_n)$, its value is either $+1$ or -1. The initial conditions x_0 and y_0 are mentioned in Table II. However, $z \to 0$ with the increase in number of iterations.

TABLE II
INITIAL CONDITION FOR CORDIC OPERATIONS

Coordinate	Operations	Initial conditions
Circular	$\cos z_{in}$	$x_0 = 0.6075252935,\ y_0 = 0,\ z_{in} = input$
	$\sin z_{in}$	
Linear	Multiplication	$x_0 = input,\ y_0 = 0,\ z_{in} = input$
Hyperbolic	$\cosh z_{in}$	$x_0 = 1.2075355,\ y_0 = 0,\ z_{in} = input$
	$\sinh z_{in}$	

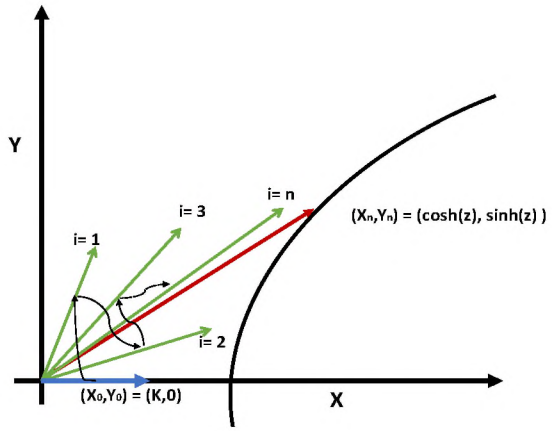

Fig. 2. Illustration of rotation convergence in CORDIC.

III. Proposed Work

In this paper, we have proposed a bfloat16 based CORDIC processor for the trigonometric, exponential and arithmetic calculations. The block diagram of proposed module is show in Fig 3. Bfloat16 based adder and subtracter are employed in the proposed work. As we know, apart from addition and subtraction, shifting plays an important role in CORDIC calculations. Shifting performed in a floating-point number is different from shifting in integer and fixed-point based numbers. To perform the shifting operation in floating-point representation, only exponent bits are shifted. The number of shifts depends on number of iterations. The given vector is rotated by an angle of α. The angle α is calculated by addition and subtraction of finite numbers of micro angles β. Addition and subtraction operations depend on the sign of z_n.

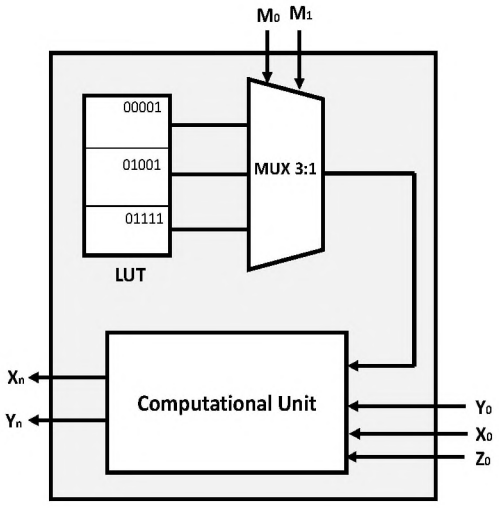

Fig. 3. Block diagram of proposed CORDIC Processor.

$$\alpha = d_0\beta_0 + d_1\beta_1 + d_2\beta_2 + ... + d_n\beta_n \tag{15}$$

Where, n is the number of iteration, β is the micro angle in each iteration and d_n is the direction of angle in each rotation.

Area and power of the proposed work is significantly reduced due to 16-bit bfloat16 floating-point number representation. We brief each operation performed by the proposed CORDIC in subsection III-A.

A. Modes and Operations

Fig 4, shows three computing modes of proposed module. These modes are explained in following subsections.

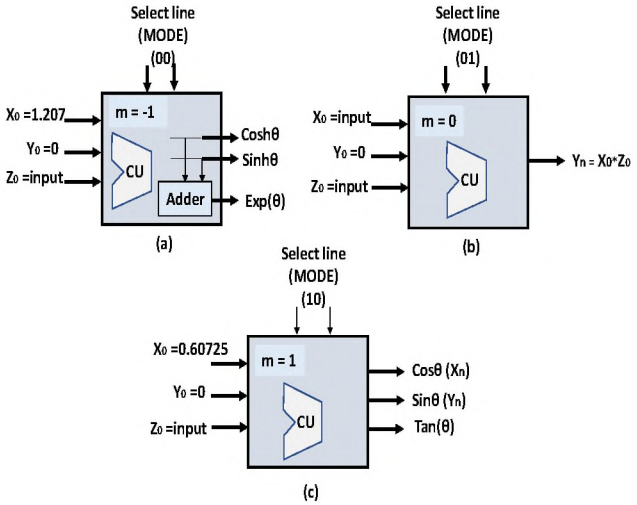

Fig. 4. Modes of operations in CORDIC Processor (a) Hyperbolic Calculation, (b) Linear computation and (c) Trigonometric expression calculation.

1) Hyperbolic and Exponential Calculation: To calculate the hyperbolic function, we have to select the hyperbolic coordinate system by choosing the mode value $m = -1$ (which is activated by the select line 00). LUT store the value of $tanh^{-1}(2^{-n})$ for hyperbolic coordinate system. All the LUT values are stored in bfloat16 floating-point representation in the memory element. The CORDIC equation for the hyperbolic coordinate system is as follow:

$$x_{n+1} = x_n + y_n d_n 2^{-n} \tag{16}$$
$$y_{n+1} = y_n + x_n d_n 2^{-n} \tag{17}$$
$$z_{n+1} = z_n - d_n tanh^{-1}(2^{-n}) \tag{18}$$

We assign the initial condition according to Table II.
As we knew that,

$$cosh\alpha = \frac{e^{\alpha} + e^{-\alpha}}{2} \tag{19}$$
$$sinh\alpha = \frac{e^{\alpha} - e^{-\alpha}}{2} \tag{20}$$

After the completion of iterations in hyperbolic coordinate system, $x_n = cosh(z_{in})$ and $y_n = sinh(z_{in})$. The exponent function is computed by employing x_n and y_n as the input of

an adder and subtracter, as shown in equation (21) and (22), respectively.

$$x_n + y_n = exp(z_{in}) \quad (21)$$
$$x_n - y_n = exp(-z_{in}) \quad (22)$$

2) Linear Calculation: Linear functions such as square root and multiplication are calculated in linear mode. It is activated by selecting the mode value to $m = 0$ (which is activated by the select line 01). The LUT stores values of 2^{-n} for linear coordinate system. Two inputs are provided at X_n and Z_n, and $Y_n = 0$ is the initial condition for linear calculations, as mentioned in Table II.

3) Trigonometric Calculation: Trigonometric functions are computed in a circular coordinate system by selecting the mode value to $m = 1$ (which is activated by the select line 10). The initial conditions are assigned at the time of mode selection, as mentioned in Table II. The LUT contains values of $tan^{-1}(2^{-n})$ and starts at the address 01111. All the values are stored in bfloat16 representation, which also reduces the area of LUT. The following equations (mentioned as follows) are calculated in hardware by employing a bfloat16 based adder unit.

$$x_{n+1} = x_n - y_n d_n 2^{-n} \quad (23)$$
$$y_{n+1} = y_n + x_n d_n 2^{-n} \quad (24)$$
$$z_{n+1} = z_n - d_n tan^{-1}(2^{-n}) \quad (25)$$

The above equations are calculating the trigonometric functions, such as $x_n = cos(z_{in})$, $y_n = sin(z_{in})$ and $tan(z_{in}) = \frac{y_n}{x_n}$.

B. CORDIC Computation Modules

The CORDIC processor module contains an adder and shifter with a look-up table to perform the computation. In each iteration the shift and arithmetic operations are performed using a computation block, shown in Fig 5. This computation block contains the adder and shifter unit as a submodule.

1) Bfloat16 Addition/Subtraction Unit: Bfloat16 based floating-point adder is a combination of $8 - bit$ shifter and $8 - bit$ kogge-stone adder. The inputs of adder module is bfloat16 based floating-point values and operation to be performed by assigning 'ADDbar_SUB' as 1 for subtraction and 0 for addition. The output of module is a 16-bit bfloat16 floating-point number. The delay of this module is computed from equation (26).

$$t_{add/sub} = t_{AND} + t_{OR} + (log_2 N)t_{AND} + (log_2 N)t_{OR} + t_{xor} \quad (26)$$

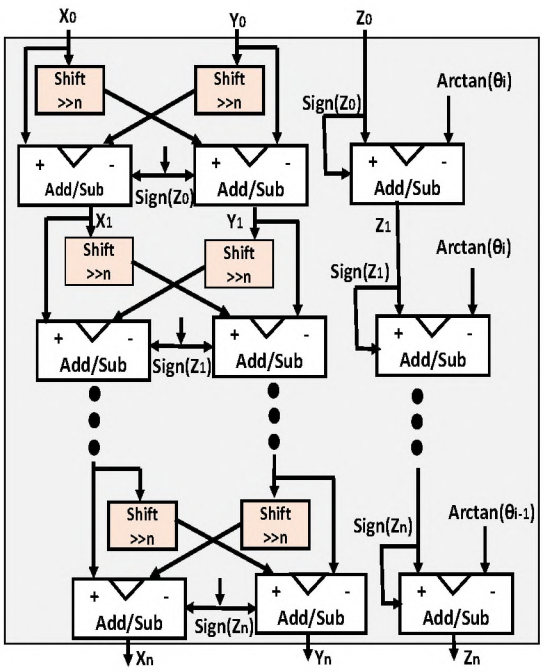

Fig. 5. CORDIC Computation Unit.

2) Bfloat16 based Number Shifter: The shift operation is performed on the basis of the number of iterations. In equation (16), operation 2^{-n} can be performed by the shifter. In the shifter, the exponent bit of bfloat16 floating-point number is shifted by a number of iteration. The 8-bit length of the exponent gives a high range of shift operations. The delay of this module is computed from equation (27).

$$t_{shifter} = t_{not} + 2t_{nand} \quad (27)$$

The total delay of the computation unit is defined as the sum of equation (26) and (27), which is given as equation (28)

$$T = t_{add/sub} + t_{shifter} \quad (28)$$

C. Look-up Table (LUT)

Look-up tables, in the CORDIC algorithm, are used to store the values of inverse of tanh, tan and 2^{-n} for hyperbolic, circular and linear modes respectively. Values are stored from the address of 00001 to 0111. Different LUT values for different modes are chosen from the given address by the mode select line.

IV. Result and Discussion

In this section, the results of proposed module is compared with a single-precision based CORDIC processor. The proposed module is synthesized using Synopsys Design Vision

TABLE III
ANGLE CALCULATION USING LUT.(HERE 32° IS COMPUTED)

$\tan(Q_n)$	Table value	Process steps	Direction (d_n)
$2^{-0}=1$	45	$45.0(>32)$	-1
$2^{-1}=0.5$	26.6	$45.0-26.6=18.4(<32)$	$+1$
$2^{-2}=0.25$	14.0	$18.4+14.0=32.4(>32)$	-1
$2^{-3}=0.125$	7.1	$32.4-7.1=25.3(<32)$	$+1$
$2^{-4}=0.0625$	3.64	$25.3+3.6=28.9(<32)$	$+1$
$2^{-5}=0.031251$	1.84	$28.9+1.8=30.7(<32)$	$+1$
$2^{-6}=0.015625$	0.9	$30.7=0.9=31.6(<32)$	$+1$
$2^{-7}=0.0078125$	0.44	$31.6+0.44=32.04(<32)$	$+1$

using a TSMC 180nm process. The synthesis results are compared with the single-precision based CORDIC processor. Table IV presents the area and power comparison of the proposed module with a single precision based CORDIC processor. The results shows that area and power of the proposed module is $3.2\times$ and $3.38\times$ better than the FP32 based module.

TABLE IV
COMPARISON OF AREA AND POWER AFTER SYNTHESIS USING TSMC180NM

Module		CORDIC Processor	
Floating-point representation		FP32	Proposed
Area (μm^2)	Combinational	94.502	25.957
	Sequential	17.485	9.009
	Total	111.987	34.966
Power (uW)	Static	524.400	165.800
	Dynamic	369.769	98.340
	Total	894.169	264.140

The results of proposed module is verified on a Zynq evaluation FPGA board. Fig 6 shows the result achieved using FP32 and bfloat16 based modules. The FP32 based module provides the result with a precision of $4-bit$ after decimal. On the other, hand bfloat16 based results are with the precision of $2-bit$ after decimal. However, the resource utilization, as shown in in Table V describes that the FP32 based module consumes $3.36\times$ more resources compared to the proposed module.

TABLE V
RESOURCE UTILIZATION OF CORDIC PROCESSOR

Resource	Single-precision		Bfloat16	
	Available	Utilization	Available	Utilization
LUT	53200	2662	53200	699
FF	106400	296	106400	153
IO	200	130	200	66
BUFG	32	1	32	1
Total Resources	159832	3089	159832	919
Comparative Utilization	3.36X		X	

The precision of bfloat16 based CORDIC is acceptable to work in the machine learning models, during training of spiking neural networks (SNN), learning of ANN models, and weight optimization in machine learning algorithms [12] [15]. The given applications of the proposed unit is a suitable choice

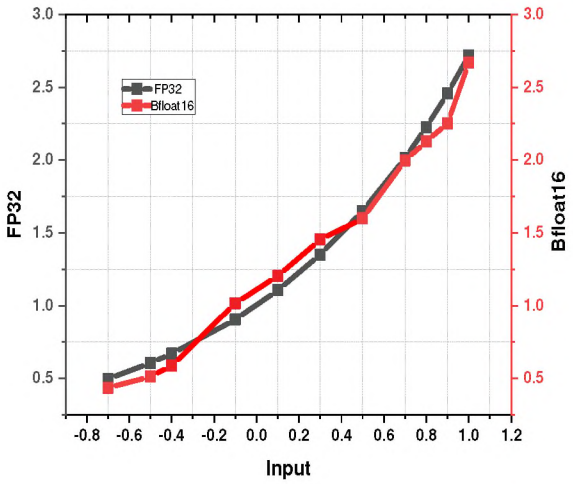

Fig. 6. Comparison of bfloat16 and FP32 based CORDIC results

for hardware implementation of low power and area efficient applications.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a bfloat16 based CORDIC processor. The proposed module can calculate trigonometric, hyperbolic, and linear computations without employing the multiplication unit. The lack of a multiplication unit leads toward area and power saving. The bfloat16 representation has only a 7-bit mantissa and 8-bit exponent which further reduces the resource utilization of the adder or subtracter module. The result of the proposed module is verified on a Zynq evaluation FPGA board. The area and power of the proposed design is compared with the single-precision based module, which utilizes $3.2\times$ and $3.38\times$ more area and power, respectively. Future work with the proposed module will be application in neuromorphic areas. As neuromorphic applications require high parallel computation, which increases area and power. The use of the proposed module will lead towards less resource utilization.

ACKNOWLEDGMENT

The work has been supported from the programme INTER-EXCELLENCE (LTAIN19100) funds of the Ministry of Education, Youth and Sports, Czech Republic, project LTAIN19100 Artificial Intelligence Enabled Smart Contactless Technology Development for Smart Fencing and Ministry of Science and Technology through project DST/INT/Czech/P-11/2019 entitled as “Smart Contactless Technology Development for Smart Fencing”. The support provided for the projects by respective ministries in Czech Republic and India is gratefully acknowledged.

REFERENCES

[1] Volder, Jack E. ”The CORDIC trigonometric computing technique.” IRE Transactions on electronic computers 3 (1959): 330-334..

[2] J. R. Cavallaro and F. T. Luk, "CORDIC Arithmetic for an SVD Processor," Journal of Parallel and Distributed Computing, vol. 5, no. 3, 1988.
[3] Baker, P. W. "Suggestion for a fast binary sine/cosine generator." IEEE Transactions on Computers 25.11 (1976): 1134-1136.
[4] de Lange, Alfons AJ, and Ed F. Deprettere. "Design and implementation of a floating-point quasi-systolic general purpose CORDIC rotator for high-rate parallel data and signal processing." IEEE Symposium on Computer Arithmetic. 1991.
[5] Ercegovac, Milos D., and Tomas Lang. "Redundant and on-line CORDIC: Application to matrix triangularization and SVD." IEEE Transactions on Computers 39.6 (1990): 725-740.
[6] A. Tiwari, G. Trivedi, and P. Guha, Design of a low power bfloat16 pipelined mac unit for deep neural network applications, in 2021 IEEE Region 10 Symposium (TENSYMP). IEEE, 2021, pp. 18.
[7] BFLOAT16-hardware numerics definition. https://software.intel.com/sites/default/files/managed/40/8b/bf16-hardware-numerics-definition-white-paper.pdf. Accessed: 2019-03-22.
[8] Lakshmi, Boppana, and Anindya Sundar Dhar. "CORDIC architectures: A survey." VLSI design 2010 (2010).
[9] Kota, Kishore, and Joseph R. Cavallaro. "Numerical accuracy and hardware tradeoffs for CORDIC arithmetic for special-purpose processors." IEEE Transactions on Computers 42.7 (1993): 769-779.
[10] Meher, Pramod K., et al. "50 years of CORDIC: Algorithms, architectures, and applications." IEEE Transactions on Circuits and Systems I: Regular Papers 56.9 (2009): 1893-1907.
[11] Lee, Jeong-A., and Ed FA DEPRETTERE. "A low-cost floating point vectoring algorithm based on CORDIC." IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences 83.8 (2000): 1654-1662.
[12] Wang, Yi, et al. "Energy efficient spiking neural network processing using approximate arithmetic units and variable precision weights." Journal of Parallel and Distributed Computing 158 (2021): 164-175.
[13] P. Nilsson, A. U. R. Shaik, R. Gangarajaiah, and E. Hertz, Hard- ware implementation of the exponential function using taylor series, in 2014 NORCHIP. IEEE, 2014, pp. 14.
[14] Exponential taylor methods: Analysis and implementation, Computers & Mathematics with Applications, vol. 65, no. 3, pp. 487 499, 2013, efficient Numerical Methods for Scientific Applications.
[15] Wu, Jiajun, et al. "Efficient design of spiking neural network with STDP learning based on fast CORDIC." IEEE Transactions on Circuits and Systems I: Regular Papers 68.6 (2021): 2522-2534.