Received 10 August 2020; revised 13 November 2020; accepted 30 November 2020. Date of current version 26 January 2021.

Digital Object Identifier 10.1109/OJCAS.2020.3042743

RECON: Resource-Efficient CORDIC-Based Neuron Architecture

GOPAL RAUT [1] (Graduate Student Member, IEEE), SHUBHAM RAI [2] (Graduate Student Member, IEEE), SANTOSH KUMAR VISHVAKARMA [1] (Member, IEEE), AND AKASH KUMAR [2] (Senior Member, IEEE)

[1]Department of Electrical Engneering, Indian Institute of Technology Indore, Indore 453 552, India

[2]Chair for Processor Design, Center for Advancing Electronics Dresden, Technische Universität Dresden, 01169 Dresden, Germany

This article was recommended by Associate Editor Y. Li.

CORRESPONDING AUTHOR: S. K. VISHVAKARMA (e-mail: skvishvakarma@iiti.ac.in)

This work was supported by the University Grant Commission (UGC), Government of India and the HiPEAC
Collaboration Grant, European FP7 ICT Cooperation Program.

ABSTRACT Contemporary hardware implementations of artificial neural networks face the burden of excess area requirement due to resource-intensive elements such as multiplier and non-linear activation functions. The present work addresses this challenge by proposing a resource-efficient *Co-ordinate Rotation Digital Computer* (CORDIC)-based neuron architecture (RECON) which can be configured to compute both multiply-accumulate (MAC) and non-linear activation function (AF) operations. The CORDIC-based architecture uses linear and trigonometric relationships to realize MAC and AF operations respectively. The proposed design is synthesized and verified at 45nm technology using Cadence Virtuoso for all physical parameters. Implementation of the signed fixed-point 8-bit MAC using our design, shows 60% less area, latency, and power product (ALP) and shows improvement by 38% in area, 27% in power dissipation, and 15% in latency with respect to the state-of-the-art MAC design. Further, *Monte-Carlo* simulations for process-variations and device-mismatch are performed for both the proposed model and the state-of-the-art to evaluate expectations of functions of randomness in dynamic power variation. The dynamic power variation for our design shows that worst-case mean is $189.73\mu W$ which is 63% of the state-of-the-art.

INDEX TERMS AF, CORDIC, configurable architecture, MAC, neural network.

I. INTRODUCTION

AN ARTIFICIAL neural network (ANN) has been a game-changer in computing paradigms within the last decade. The main advantage of an ANN over other prediction techniques is in its capability to learn hidden relationships in data with unequal variability [1]. Efficient VLSI architectures based on a fully connected neural network (FCNN) have been proposed in the literature, targeting diverse applications [2]–[4]. However, FCNNs are compute-intensive and often require high computational power [5], [6].

A typical neural network (NN) is a graph network consisting of several layers, with each layer having multiple nodes called *neurons*. In an NN, each neuron performs two basic mathematical operations: sum-weighted input feature using *Multiply-Accumulate* (MAC), and non-linear *Activation Function* (AF) over calculated sum as shown in Fig. 1. The performance and accuracy of a neural network primarily depend upon the bit precision of computation [3]. Specifically, in the case of the hardware implementation of neural networks, higher precision gives higher accuracy but often comes with high area and power overheads.

ANNs can be designed and implemented on different platforms like CPU, GPU, FPGA, or ASIC. The drawback of a general-purpose platform such as a CPU is the low utilization of their resources that reflects in high power consumption and low performance. Moreover, it fails to exploit the underlying parallelism of a neural network. GPUs exploit the underlying parallelism but often leads to high power consumption. On the other end of the spectrum, ASICs have specialized hardware structures for MAC and

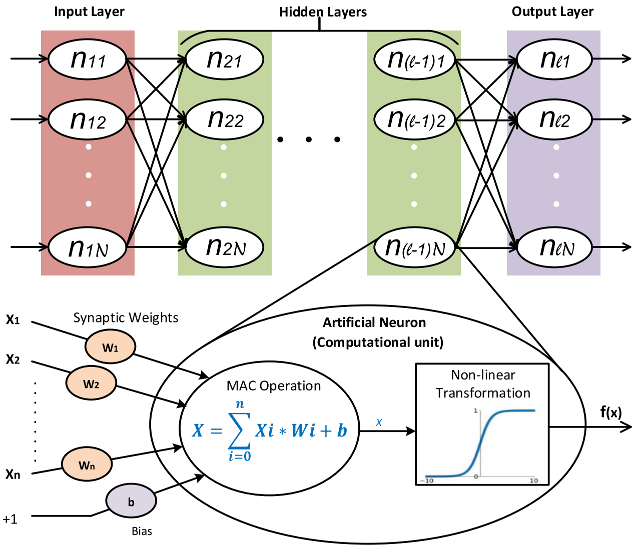


FIGURE 1. Fully connected artificial neural network architecture with neuron showing MAC and activation function computation.

AF operation, and thus they achieve high resource utilization and lower power consumption [7]. However, ASIC-based hardware accelerators are constrained by their inability to support different types of neural networks due to their fixed design architecture [8]. This trade-off gets even more complicated when configurable architectures are required. The FPGAs offer configurable hardware designs but need more chip areas with higher power consumption as compared to ASICs [5], [9], [10].

A flexible design architecture that scales well with the size of the neural network without compromising on accuracy, provides a nice balance between area and functionality. An area and power-efficient configurable architecture are desirable at all technology nodes [11]. Keeping this in mind, we propose a *Resource Efficient Coordinate Rotation Digital Computer (CORDIC)-based neuron architecture* (RECON) that provides compelling application opportunities and enables efficient yet configurable computations required in a neural networks.

The CORDIC algorithm used in RECON employs an iterative convergence approach and uses minimal resources [12]. It is a desirable choice where the cost-to-performance ratio is critical [13]. The CORDIC architecture uses only shift-and-add operations [14] and can perform several computing tasks such as *linear*, *trigonometric*, *hyperbolic*, and *exponential* functions. Hence, CORDIC-based architecture can be tuned to calculate many transcendental algebraic functions such as multiplication, division, hyperbolic tangent, and sigmoid functions [14].

A. MOTIVATION

In the case of hardware implementation of an ANN, increasing computational complexity has an unfavorable impact on area and performance. Moreover, an ASIC design is not adaptable, and any configurability such as flexibility in bit precision, or configurability in the type of activation function such as *sigmoid*, hyperbolic tangent (*tanh*), comes with additional overhead. Computation with higher bit precision (32-bit or 64 bit) is also expensive in terms of an area and power [15]. Thus, a reduction in hardware complexity (i.e., bit precision) and faster response without compromising accuracy is highly desirable [16].

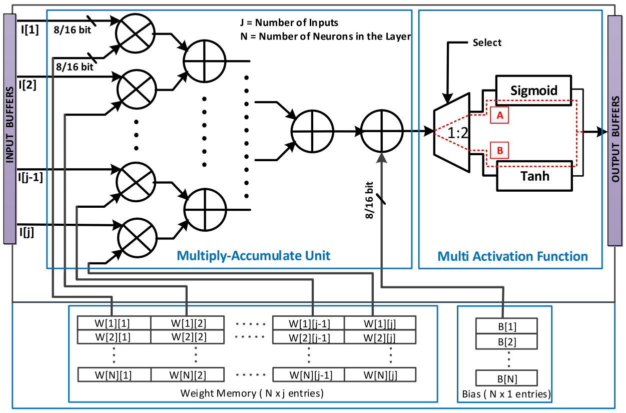


FIGURE 2. The neuron architecture having multiply-accumulate unit with parallel multiplier for j^{th} input and *n* neurons in the layer followed by multiple activation functions (path-A and path-B).

The conventional ASIC-based configurable architecture of neurons with multiply-accumulate unit and multiple activation functions proposed in [17]–[19] is shown in Fig. 2. Each MAC unit consists of several multipliers followed by an adder tree. The output of the MAC unit is then fed to the multiplexer which selectively feeds to a particular activation function depending upon an application. The architecture shown in Fig. 2 has some major drawbacks like area overhead due to array of multipliers and use of separate hardware path (path-A and path-B) for individual activation function. It also leads to increased data propagation delay (due to the MUX) and power dissipation (static power dissipation) due to the unused hardware because only one activation function is activated at a particular time.

To overcome the above limitations, we propose RECON which uses CORDIC algorithm. It uses a fixed-point signed arithmetic computation. The proposed logic has two benefits– Firstly, it is scalable in terms of area and power as a single block can compute both MAC and activation function. Secondly, it allows configurable activation function *(sigmoid/tanh)* and hence has reduced critical path delay and lesser power dissipation. The proposed design also maximizes hardware reuse as unlike conventional ASIC-based designs which uses *MUX* which leaves out hardware blocks unused.

B. CONTRIBUTIONS

In this work, we investigate design strategies and optimizations for RECON which can realize both MAC and activation function computation. The major contributions are summarized in the following points:

- We propose a CORDIC-based design of an unsigned/signed computational unit which can compute both MAC and non-linear activation function.

- We demonstrate how CORDIC is configured within RECON to operate in linear or hyperbolic rotation mode to solve arithmetic (for MAC) and trigonometric operations (for AF) respectively.
- Optimization of the proposed CORDIC-based architecture in terms of area, power, and delay. We further employ power-gating approach and evaluate it with 45nm technology node to demonstrate power-savings. We analyze and discuss the impact of technology scaling on circuit's physical parameters like area, power and delay.

Experimental evaluations show that RECON as a MAC unit, has a lower area and power footprint as compared to the contemporary state-of-the-art designs proposed in [20]–[23]. Moreover, our proposed design architecture can support both signed and unsigned number representations. Similarly, in an activation function configuration, RECON returns the best area and power numbers as compared to other designs [24], [25].

C. ORGANIZATION

The rest of this article is organized as follows. Section II details about the related work and introduces preliminaries required for this work. Section III introduces the CORDIC architecture and various rotational modes required for mathematical operations. The proposed design and functioning of the configurable CORDIC architecture for MAC and activation function are given in detail in Section IV. Section V gives the experimental setup followed by simulation results and discussion in Section VI. The concluding remarks are given in Section VII.

II. BACKGROUND AND RELATED WORKS

In this section, we discuss related works which targeted ASIC implementation for ANNs.

A. RELATED WORK FOR MAC UNIT

The basic multiply-accumulate (MAC) unit consists of multiplier, adder, and accumulator blocks. The multiply-accumulate computational unit is shown in Fig. 2. The input and output arithmetic relation in the preceding layer of a neural network is shown below:

$$a_j^l = f\left(\sum \omega_{jN}^l a_N^{l-1} + b_j^l\right) \tag{1}$$

where f is the activation function (*sigmoid*/*tanh*) of computational unit k, corresponding to overall neurons in $(l-1)^{th}$ layer. To formulate this equation in matrix form, we assume a weight matrix ω^l corresponding to layer l. Elements of weight matrix ω^l are weights to the inputs of neurons from l^{th} layer with j^{th} row and N^{th} column. The term, b_j^l is the bias here.

Most of the earlier work proposed a dedicated ASIC design for the MAC unit. In [20], [26] authors designed a MAC unit using a *Vedic-multiplier* and register-based accumulator. Such kind of architecture is effective for low precision. However, it is not scalable as increasing bit-precision for higher accuracy, leads to a substantial increase in the critical path and the propagation delay. In order to carry out inference for a deep neural network, the authors in [22], [27], [28] resort to a shift-and-add-based multiplication instead of bulky multipliers for MAC computation. Though the architecture was area and power-efficient, it suffered due to low throughput. In [22], the authors proposed calculating one multiplication using shift and addition operation. The proposed approach requires n-shift left and $n-1$ addition for n-bit precision calculation. However, the design is limited for unsigned number calculation as it uses left shift operation.

Configurable n-bit (where n can be any value between 1 and 16) precision MAC unit was proposed using a digital in-memory computing concept [29]. The architecture used an XNOR-based bit-wise multiplier, a full-adder, and an SRAM cell for computation. The circuit design was efficient in terms of area and power but throughput was very low as it needs n-clocks for n-bit precision. Hence, it was not efficient for high-precision architecture. The *Wallace Tree* multiplier based MAC design was analyzed in terms of area, delay, and power in [21]. The proposed architecture was designed using only *AND* and *OR* basic gates. However, it leads to an increase in the critical path and high area utilization. The *double MAC* design proposed in [17], [30], implements two multiplication in a single clock cycle but suffers due to high resource utilization.

B. RELATED WORK FOR ACTIVATION FUNCTION

The hyperbolic tangent (*tanh*) and *sigmoid* are generally the most used non-linear activation functions in hardware implementations. Activation functions like *sigmoid* or *tanh* provide a smooth transition between excitation and inhibition, which improves the neural response [31], [32].

Various ASIC implementations of non-linear activation functions proposed in [18], [33]–[35], employ a combinational logic-based design approach consisting of memory elements, *MUX/DeMUX-trees* and logic gates. All these combinational logic-based design uses quantization states to realize activation functions. The quantization states for a particular bit precision are stored in memory elements and then selected using MUX tree. For example, in case of 8-bit precision, the number of quantization states is $2^8 = 256$ which are stored in memory elements. These designs benefit with high throughput but are not very practical for higher bit precisions as the area increases exponentially. The authors in [33] extended the combinational logic design-based approach to realize configurable AF. The configurable AF is implemented using additional hardware resources such as multiplier and sign converters to realize both *sigmoid* and *tanh* activation functions.

Instead of using memory elements, authors in [36] used power series to compute non-linear activation functions. While the design was able to cope up with area overheads with bit precisions, it suffered due to large delay that led to low performance. One of the first works to investigate CORDIC architecture for activation function implementation was [14]. The CORDIC-based implementation provides resources-efficient and configurable activation function computation. Hence, configurable activation functions are realized using CORDIC design in [15], [24], [37]. The various activation function area realized using CORDIC are given by the following equations:

$$f_1(z) = sigmoid(z) = \frac{1}{1+e^{-z}} = \frac{1+tanh(z/2)}{2} \tag{2a}$$

$$f_2(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2b}$$

$$= 1 - 2\ sigmoid(-2z) = 2\ sigmoid(2z) - 1 \tag{2c}$$

The authors in [25] proposed direct computation of a single *sigmoid* activation function with CORDIC architecture. For negative values, they used $2's$ complement arithmetic computation which requires an extra computation step that leads to area and performance overhead. Similarly, the authors used an additional multiplication step to the underlying architecture for calculating *tanh* function [24]. The scaling of inputs by two implies a double rotation technique in polar coordinates. However, it requires an extra multiplier and a subtractor for *tanh* calculations. It scales the *sigmoid* input by two for *tanh* activation function realization using equations (2a) and (2b). These extra steps increase the delay and area of the overall design.

Another approach using a near-threshold CORDIC design technique for low-power applications was proposed in [38]. The authors designed an internal sub-block logarithmic shifter and adder of the CORDIC core using dynamic logic. However, this work was not targeted towards a semi-custom ASICs in general.

To address these limitations for both MAC and activation function, the proposed design, RECON uses a single CORDIC architecture with a power gating technique for the realization of an individual neuron.

III. CORDIC ALGORITHM

The COordinate Rotation DIgital Computer (CORDIC) algorithm realizes various mathematical functions by rotating a vector coodinates. The underlying principle allows solving the trigonometric relationships involved in plane coordinate rotation and conversion from rectangular to polar coordinates [39]. The CORDIC architecture is configured to operate in three rotation modes – circular, linear, or hyperbolic rotation. In this connection, a unified algorithm for linear and hyperbolic CORDIC is an extension of the basic CORDIC algorithm for a circular trajectory as explained in [40], [41]. It is a convergence method for evaluating linear, hyperbolic, (and other) functions using simple logic blocks—Multiplexer, shifters, adders, memory-based (ROM) pre-calculated constants. The propagation delay of a conventional CORDIC is the sum of the delay of a multiplexer, adder/subtractor, barrel-shifter, and feedback resistor which involves one CORDIC unit for each micro-rotation. The CORDIC uses a pseudo rotation calculation which is a scaled version of real rotation. The real circular rotation co-ordinate calculation equations are

TABLE 1. Used adder and subtractor functional similarity in CORDIC design architecture.

Components	Sum / Difference	C_{out}/B_{out}
Adder	S = X $\oplus$ Y $\oplus$ C_{in}	C_{out} = X $\oplus$ Y $\cdot$ C_{in} + XY
Subtractor	D = X $\oplus$ Y $\oplus$ B_{in}	B_{out} = $\overline{X \oplus Y}$ $\cdot$ B_{in} + $\overline{X}$ Y

$$\begin{aligned} X_{(i+1)} &= X_i \cdot cos\ \alpha_i - Y_i \cdot sin\alpha_i \\ Y_{(i+1)} &= Y_i \cdot cos\ \alpha_i + X_i \cdot sin\alpha_i \\ Z_{(i+1)} &= Z_i - \alpha_i \end{aligned} \tag{3}$$

Here, (X_i, Y_i) is a set of coordinate components representing an i^{th} state. In terms of polar coordinates, an angle α is used where the new coordinates (X_{i+1}, Y_{i+1}) can be easily reached. The above equations describe a real rotation of the plane vector v_i to v_{i+1} at each iteration. In these equations, $sin\ \alpha_i$ and $cos\ \alpha_i$ are replaced with $sinh\alpha_i$ and $cosh\alpha_i$ respectively for hyperbolic circular rotation. The CORDIC algorithm uses pseudo rotation for function calculation. Taking $cos\ \alpha_i$ (or $cosh\alpha_i$ in case of hyperbolic rotation mode) (i.e., the K_i) term as common in Eq. (3) and scaling the equations in Eq. (3) by K_i (scaling factor $= 1/K_i$), CORDIC equations for all mode of trajectories are then formulated as:

$$\begin{aligned} X_{(i+1)} &= X_i - Y_i \cdot tan\alpha_i \\ Y_{(i+1)} &= Y_i + X_i \cdot tan\alpha_i \\ Z_{(i+1)} &= Z_i - \alpha_i \end{aligned} \tag{4}$$

where $\alpha_i = 2^{-i}\ or\ tan^{-1}(2^{-i})\ or\ tanh^{-1}(2^{-i})$ depending whether CORDIC operates in linear, circular or hyperbolic mode respectively. α_i is the rotation angle in radians for the each iteration. The scaling factor $1/K_i$ is unique for an individual mode of operation.

In this connection, adder/subtractor block is designed for output calculation using the equations shown in Table 1.

The Eq. (4) then converges to the following equations:

$$X_{i+1} = X_i - m \cdot d_i \cdot Y_i \cdot 2^{-i} \tag{5a}$$

$$Y_{i+1} = Y_i + d_i \cdot X_i \cdot 2^{-i} \tag{5b}$$

$$Z_{i+1} = Z_i - d_i \cdot E_i \tag{5c}$$

Here mode $m \in \{1, 0, -1\}$ indicates a circular, linear, and hyperbolic coordinate system, respectively. E_i is the memory constant at each i^{th} iteration which is equal to 2^{-i}, $tan^{-1}(2^{-i})$ and $tanh^{-1}(2^{-i})$ for linear, circular and hyperbolic rotation mode respectively.

In general, the CORDIC algorithm needs n iterations for n significant digits of the fractional part. For higher precision, higher rate of shifting is required which demands more clock cycles for maintaining the computation accuracy. The optimized CORDIC architecture for all modes of operation is shown in Fig. 3. Fig. 3 shows three separate flows for calculation of X_n, Y_n and Z_n. Shift registers are used to right-shift

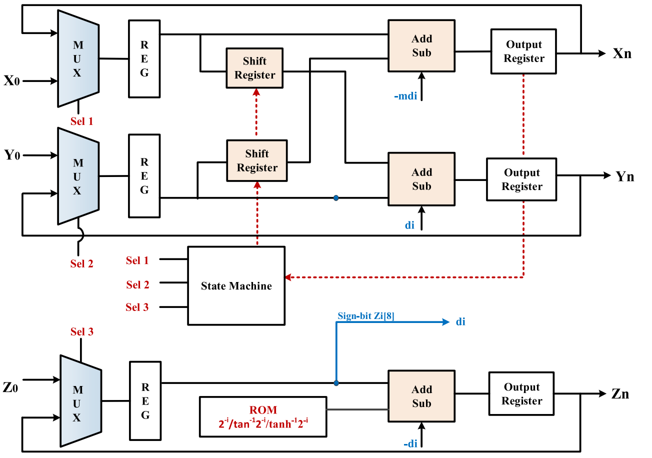


FIGURE 3. Signed 8-bit precision architecture for the basic CORDIC-based design.

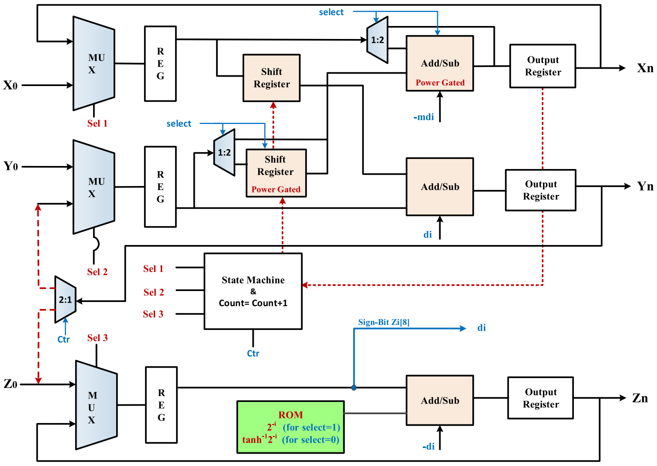


FIGURE 4. The efficient recursive sign 8-bit precision RECON architecture configured by *select* and *ctr* line for MAC and AF computation.

X_n and Y_n. The sign bit of Z_i determines the direction signal d_i. The functionality of the add/sub-block used in CORDIC architecture depends on the d_i direction. The direction signal is important as it helps to converge the computation iteratively. The d_i represents the rotation direction for i^{th} iteration such that the output at Z converges to 0.

IV. RECON ARCHITECTURE

Design of neural network accelerators requires calculations that involve multiplication, accumulation and trigonometric function such as *tanh*, *sigmoid* etc. The proposed CORDIC-based design enables such computation by using pre-computed constants with shift-and-add operation for fast computation and minimum resources utilization. As compared to a conventional architecture which has dedicated blocks for MAC and activation function calculation, RECON focuses on re-utilization of logic architecture (via iteration) for both MAC as well as activation function calculation. The architecture uses linear and hyperbolic rotation mode for multiply-accumulation and non-linear activation function calculation respectively. RECON can further support both signed and unsigned computations. The optimized CORDIC architecture with power gating technique is shown in Fig. 4. The proposed architecture and its operation are described in the following subsections.

A. MULTIPLY-ACCUMULATE COMPUTATION USING RECON

The multiply-accumulation computation technique used in this work depends on the shift-and-add multiplication approach. It has two principal features. Firstly, arithmetic right-shift is used instead of the left shift operation. This technique allows RECON to support both signed and unsigned numbers. Secondly, for mathematical computations (such as addition and multiplication) for an n-bit precision, the iterative convergence using the CORDIC algorithm returns the same accuracy as one gets using conventional combinational logic.

In the present work, we focus on fixed-point number representation for multiply-accumulate computation, as the floating-point representation (IEEE-754 notation) in MAC has a higher complexity and power consumption [16]. Moreover, the choice of bit-precision in case of weight/bias constants is one of the important considerations as it directly impacts the area and power of the hardware implementation. In order to reach an acceptable precision that gives a good trade-off between area and accuracy, we have trained the neural network (as shown in Appendix A) for different precision. We got the best accuracy-area trade-off for 8-bit precision among the possible combinations.

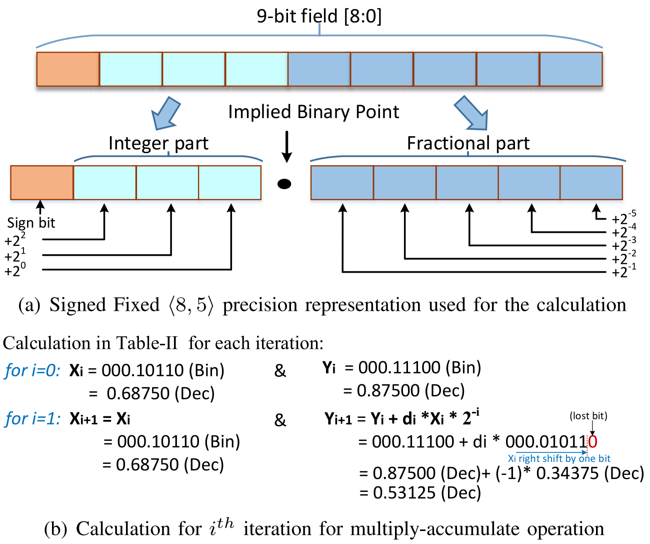
We have used a 9-bit format that reserve 1 bit for the sign, 3 bits for the integer part, and 5 bits for the fractional part. Fixed $\langle 8,5\rangle$ represents an 8-bit fixed-point number of which five rightmost bits are fractional. The 8-bit signed fixed-point representation with binary point is shown in Fig. 5 (a). The most significant bits (MSB) [7:5] are assigned for the integer part and hence the maximum multiplication output range of -7.968 to $+7.968$ is achieved using this representation.

The MAC operation is realized using RECON as shown in Fig. 4. In linear mode, the mode variable, m is considered as 0 and E_i is considered as 2^{-i}. The equation (5) then translates the general output as shown below

$$X_{i+1} = X_i \tag{6a}$$
$$Y_{i+1} = Y_i + d_i \cdot X_i \cdot 2^{-i} \tag{6b}$$
$$Z_{i+1} = Z_i - d_i \cdot 2^{-i} \tag{6c}$$

Here, Y_{i+1} computes the multiply-accumulate operation when $Z_0 \rightarrow 0$. The computation is valid as long as the binary point is allowed to float. We have considered fixed $\langle 8,5\rangle$ representation hence binary point is floated for 5 iterations. This calculation assumes that both X_i (input) and Z_i (weight) are fractions within the range of $\{-1,+1\}$. After 5 iterations, Eq. (6) translates as follows:

$$x_n = x_0 \quad \& \quad z_n \cong 0 \quad \textit{whereas},$$
$$y_n = y_0 + x_0 * z_0 \tag{7}$$



(a) Signed Fixed ⟨8, 5⟩ precision representation used for the calculation

Calculation in Table-II for each iteration:

for i=0: X_i = 000.10110 (Bin) = 0.68750 (Dec) & Y_i = 000.11100 (Bin) = 0.87500 (Dec)

for i=1: $X_{i+1} = X_i$ = 000.10110 (Bin) = 0.68750 (Dec) & $Y_{i+1} = Y_i + d_i * X_i * 2^{-i}$ = 000.11100 + di * 000.010110 = 0.87500 (Dec)+ (-1)* 0.34375 (Dec) = 0.53125 (Dec)

(b) Calculation for i^{th} iteration for multiply-accumulate operation

FIGURE 5. Data representation with binary point and arithmetic calculation.

where x_0, y_0 z_0 represent the input, bias value and corresponding weight respectively. This is also shown in Fig. 4. This implementation is similar to the standard shift and add multiplication.

From Eq. (7), the CORDIC algorithm for multiplication $x_0 \times z_0$ is derived using a series representation for weights shown below:

$$\begin{aligned} x_j = x_{jN} * \omega_N \quad &= x_{jN} * \sum_{i=1}^{j} a_i * 2^{-i} \\ = \sum_{i=1}^{j} x_{jN} * a_i * 2^{-i} &= \sum_{i=1}^{j} a_i * x_{jN} * 2^{-i} \end{aligned} \tag{8}$$

The equation states that x_j is composed of a shifted version of input x_0 with respect to weight z_0. This implementation is based on the standard shift and add multiplication. The unknown coefficient a_i may be found by driving ω to zero one bit at a time. If the i^{th} bit of input ω_N is non-zero, x_i is first right-shifted by i bits and added to the current value of y_j. When ω has been driven to zero all bits have been examined and x_j contains the signed product of input vector and weight.

The calculation as shown in Eq. (8) is carried out using RECON in linear mode. The sign bit of Z is used to calculate the value of d_i. The computation has to perform till $Z_0 \to 0$ for exact calculation that demands high bit precision computation for approaching zero [42]. The inputs and output of each iteration of a MAC operation is shown in Table 2. Calculation for one of the iteration is shown in Fig. 5 (b). A single MAC operation, thus takes 5 iterations to compute.[1]

TABLE 2. Iteration-level calculation shown for MAC computation using CORDIC in linear mode for fixed ⟨8, 5⟩ representation.

i	**d_i**	**$X_{i+1} \to X_0$**	**$Y_{i+1} \to Y_i + d_i\ X_i * 2^{-i}$**	**$Z_{i+1} \to 0$**
clock	d_i for $(i+1)^{th}$	Initial Conditions/Inputs		
iteration	iteration	input data	bias	weight
initial	–	0.68750	0.18750	0.90625
0	+1	0.68750	0.87500	-0.09375
1	-1	0.68750	0.53125	0.40625
2	+1	0.68750	0.68750	0.15625
3	+1	0.68750	0.75000	0.03125
4	+1	0.68750	0.78125 (0.81054)	-0.03125

The output at $Y_{i+1}[8:0]$ is the MAC output after 5 iterations which is then used for activation function calculation. The exact calculation using Eq. (7), returns a value of Y_n as 0.81054 as shown in Table 2. Since we are using quantization for 8 bits precision, the value returned after 5 iterations is 0.78125. We can see that the value returned is just 3% less than the exact(64-bit floating-point calculation) results.

In order to have an additional saving in static power dissipation, the add/sub and the shift register blocks are power-gated as they are not required for the MAC calculation (shown in Fig. 4).

B. ACTIVATION FUNCTION COMPUTATION USING RECON

Section II-B describe the general specific equation (Eq. (2)) based on mode selection which can calculate many functions. The generalized unified CORDIC rotation matrix for all modes of operation including the hyperbolic trajectory of operation as an extension of Eq. (5) is given as:

$$\begin{bmatrix} X_{(i+1)} \\ Y_{(i+1)} \end{bmatrix} = K_i \begin{bmatrix} 1 & -m \cdot d_i \cdot 2^{-i} \\ d_i \cdot 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} X_{(i)} \\ Y_{(i)} \end{bmatrix}$$

As this section focuses on the hyperbolic rotation mode CORDIC architecture, the CORDIC coordinate equations for hyperbolic calculation are given as:

$$X_n = K_i \cdot (X\ coshZ + Y\ sinhZ) \tag{9a}$$
$$Y_n = K_i \cdot (Y\ coshZ + X\ sinhZ) \tag{9b}$$
$$Z_n = Input\ at\ Z_0 \longrightarrow 0 \tag{9c}$$

The generated MAC output after 5 iterations at Y_{out} is considered as the input for activation function at Z_{in} through feedback and it is controlled by *ctr* pin shown in Fig. 4. In order to evaluate the *sinh(Z)* and *cosh(Z)* shown in Eq. (9), we choose $m = -1$ and $K_i = 0.8281$ as the scaling factors in pseudo rotation which is compensated by applying (i) $1/K_i = 1.2075$ at $X_0[8:0]$, (ii) $Y_0[8:0] = 0$ and (iii) MAC output as input at $Z_0[8:0]$. Most significant bits (MSBs), $Y[8]$ and $Z[8]$ are used for generating '*di*' signal. The direction, d_i is chosen in the range $\in \{-1, 1\}$ based on the sign of the previous output, i.e., current input in the each iteration. The signal '*di*' is fed to the adder/subtractor block which decides whether addition or subtraction has to be done. The hyperbolic calculation as shown in Eq. (9), hence, transforms as shown below:

$$X_{i+1} = X_i + d_i \cdot Y_i \cdot 2^{-i} \tag{10a}$$
$$Y_{i+1} = Y_i + d_i \cdot X_i \cdot 2^{-i} \tag{10b}$$

[1] Since the binary point is allowed to float for five iteration of right shift, we stop calculating for the value for Y after 5 iterations.

TABLE 3. Iteration-level calculation shown for activation function using CORDIC in hyperbolic mode.

i	$\mathbf{d_i}$	$\mathbf{X_{i+1}\rightarrow}$ **cosh Z**	$\mathbf{Y_{i+1}\rightarrow}$ **sinh Z**	$\mathbf{Z_{i+1}\rightarrow 0}$
clock iteration	for $(i+1)^{th}$ iteration	Initial Conditions/Inputs		
		1/K_i	reset	Input AF
initial	–	1.2075	0.0000	0.78125
1	1	1.2075	0.6037	0.2319
2	1	1.3584	0.9056	-0.0208
3	-1	1.2452	0.7358	0.1022
4	1	1.2911	0.8136	0.0397
5	1	1.3172	0.8554	0.0084

$$Z_{i+1} = Z_i - d_i \cdot tanh^{-i}\left(2^{-i}\right) \tag{10c}$$

The CORDIC module is used in hyperbolic rotation mode for realizing *sinh* and *cosh* functions using Eq. (10). We take an example calculation for *sinh(30)* and *cosh(30)* as shown in Table 3. The final output of the MAC operation shown in Table 2, is taken as input for the activation function calculation. The final desired outputs for hyperbolic calculations (*sinh(Z)* and *cosh(Z)*) are calculated in another 5 clock cycles (as shown grayed out in Table 3). After 5 iterations, *Y* is again reset at the 6th clock cycle for next evaluation. This gives X_n and Y_n as *cosh* and *sinh* functions respectively of the previous evaluation. The *sinh* and *cosh* functions are further used to calculate *tanh* or *sigmoid* function as activation function calculation. Z_n gives the output of activation function applied to the MAC output of the succeeding neuron which is the final resultant.

The generated trigonometric hyperbolic functions are used for producing exponential function as required from Eq. (2) are shown in Eq. (11a). The 8-bit CORDIC output is applied to the adder for producing exponential output as shown in Fig. 6. The realization of the *tanh* function in the proposed architecture can be represented in terms of *sigmoid* function as shown in the below equations.

$$e^z = sinh(z) + cosh(z) \tag{11a}$$

$$f_1(z) = tanh(z) = \frac{sinh(z)}{cosh(z)} = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{11b}$$

$$f_2(z) = sigmoid(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z} \tag{11c}$$

The proposed work further explores the relationship between *tanh* and *sigmoid* functions using Eq. (11b). The configuration between *sigmoid* or *tanh* activation function is based on the *select_af* line. The *tanh* function is realized using the Eq. (11b). When *select_af* = *1*, the CORDIC output is directly transferred to the divider through the *MUX* to generate *tanh* function. The subtraction and the shift operation are the two basic operations that are used within the divider circuit [43] for calculating *tanh*. Additionally, in the proposed architecture, the execution of *tanh* function does not require the additional adder block as compared to [24]. However, this unused block can dissipate static power. Addressing this issue and considering the trade-off between leakage current and the speed of operation, *Power Gating* (PG) technique is used to minimize the leakage power and to improve the performance. We have implemented adders with the PG technique used in the proposed architecture as shown in Fig. 6.

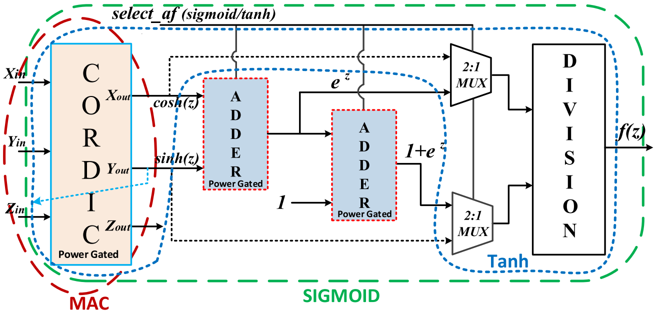


FIGURE 6. Block-level architecture for RECON. The red dotted line shows blocks required for MAC computation. The blue and green dotted lines represent the blocks used for *tanh* and *sigmoid* function computation.

The *sigmoid* function is realized using the Eq. (11c) when *select_af* = *0*. For calculation of *sigmoid*, additional adders and dividers are used as given in Eq. (11c). The *select_af* = *0* is used to activate the adder logic block for *sigmoid* calculation. In comparison to [37], which used an additional step for calculating 2's complement for calculating negative exponents, our *sigmoid* function does not need negative exponents as shown in Eq. (11c). This gives an additional saving in terms of area and delay.

The output from the MAC unit is then applied as feedback along with the initialization of the predefined parameters. The overall design allows reconfiguration, and hence the user can configure the activation function depending upon application requirement. This is the key difference between the previous approaches and the proposed architecture, leading to significant performance enhancement.

C. RECON FOR NEURON COMPUTATION

Fig. 6 shows the complete RECON architecture. Three partitions can be seen depending upon the configuration. The red encircle represents the CORDIC unit which is used in linear mode for MAC computation. The blue and green encircle represent the blocks required for *tanh* and *sigmoid* function respectively. It can be noticed that only in the case of sigmoid function calculation, all blocks are needed. While, RECON provides a configurable design to implement both MAC and activation function, the proposed design components can also be used independently depending upon the user requirement.

The overall flow for a neuron is shown in Fig. 7. In this CORDIC-based architecture, the *select* signal is set to 0 to compute the multiply-accumulate operation. The power gated blocks (Add/Sub and shift register) connected with the *select* signal, is isolated from the power supply for saving static power dissipation. When the value of *select* is 0, *m* is set to 0 so that the CORDIC operates in linear mode. The value of the MAC operation is calculated after the five clock cycle as explained before. This is represented by the count variable in Fig. 7. The output after the MAC computation

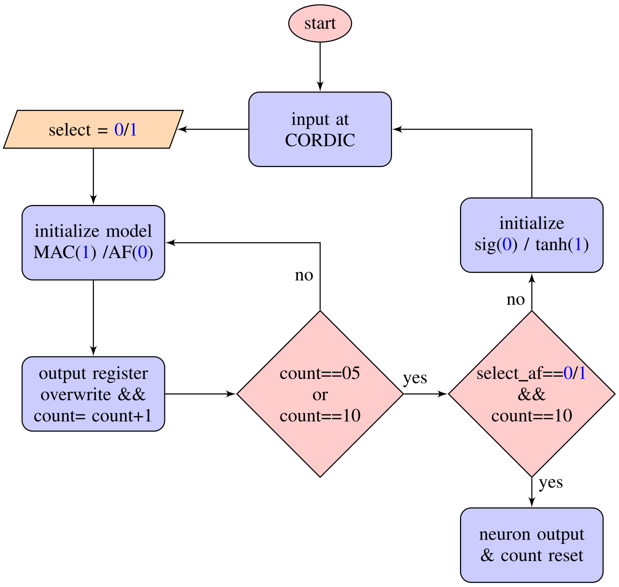

start
input at CORDIC
select = 0/1
initialize model MAC(1) /AF(0)
initialize sig(0) / tanh(1)
no
no
output register overwrite && count= count+1
count==05 or count==10
yes
select_af==0/1 && count==10
yes
neuron output & count reset

FIGURE 7. Complete flow for RECON architecture to iteratively compute MAC and activation function.

TABLE 4. Hardware implementation result for RECON.

Architecture Design	Resources Utilization LUT	FF	On-chip Power (mW) Logic	Signal	Clock	Critical Delay(ns)	PDP
Available Res.	17600	35200	–	–	–	–	–
CORDIC Arch.	17	22	0.10	0.12	0.09	2.49	0.78
Single Neuron	29	41	0.23	0.31	0.28	3.28	2.69
Network 4:4:2	262	318	1.86	2.47	2.12	11.06	71.34

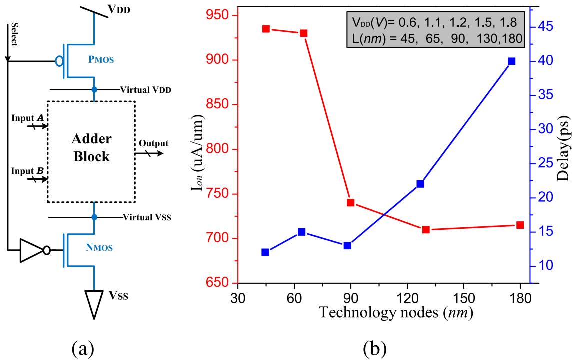


FIGURE 8. (a) Power gating technique applied for the adder block. The *select* signal is used to isolate the circuits from power supply. (b) Inverter Circuit *ON* current and delay variations with respect to technology scaling.

is then fed as a feedback to the same architecture for initialization to compute the activation function. The generated MAC output at Y_n is the input for activation function at Z_0. Here, *select_af* signal is initialized to the type of activation function to be used. The architecture employs a configurable activation function based on the application requirement. The output of the activation function is calculated in next five clock, i.e., *count=10*. At *count=10*, the complete computation of a single MAC followed by activation function is done. At *count=10*, the count register is reset and the final output of a neuron is the input for the next layer neuron. The complete iterative computation for the neuron architecture is shown in Appendix B.

V. EXPERIMENT

A. EXPERIMENTAL SETUP

To evaluate the proposed design architecture, the neuron unit having a multiply-accumulate unit and configurable activation function are represented in *HDL* using Verilog hardware description language. The RTL for our RECON architecture is synthesized and results are produced by *Design Compiler*-Synopsys [44]. The netlist file is extracted from *Encounter*-Cadence and the generated .cdl is used for RTL digital design extraction into CMOS design using the *v2lvs*-Mentor Graphics [45]. The extracted design is simulated in *Cadence-Virtuoso* [46] for performance-parameter validation at different process corners, temperature, and mismatch. Following experiments are done to validate our proposed design:

1) The first experiment shows the results for the FPGA prototyping of the proposed design.
2) In the second experiment, we evaluate the effect of power gating technique. We use a logic gate with wide-gate sized transistors so that the performance is not compromised. The coarse-grain technique is used in the design for better efficiency, less circuit complexity, and moderate switching time.
3) In order to evaluate the impact of process variation and mismatch (which increases significantly in 65*nm* and lower CMOS technology), in the third experiment, the circuit is simulated at all PVT variations.
4) In the fourth experiment, the proposed design is compared with the state-of-the-art for both MAC and AF configuration at technology nodes of 45nm.
5) In the last experiment, we carry out Monte-Carlo simulation to model the probability of different outcomes of dynamic power. It helps to calculate random variations in dynamic power dissipation due to device-mismatch in the characteristics of identically designed devices, occurring during the manufacturing of ICs.

VI. RESULTS AND DISCUSSION

A. HARDWARE IMPLEMENTATION

In order to validate the proposed design, it is implemented on FPGA *ZyboXC7z010-board*. We implement a three layer *4:4:2* fully connected artificial neural network using 8-bit precision (i.e., 8-bit weight, bias, input). The single neuron consists of a MAC followed by sigmoid AF design using the proposed CORDIC based architecture. The post implementation hardware resources utilization is given in Table 4.

B. GATE SIZING

At lower technology nodes, static power dissipation is the biggest concern. In order to implement power-gating technique, appropriate gate-sizing is an important step. For CMOS-based inverter circuit with *coarse-grain* power gating

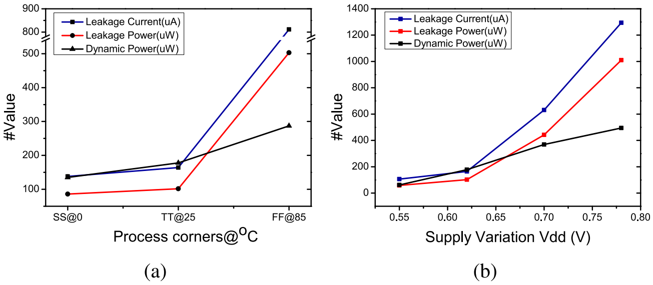


FIGURE 9. **Performance parameter variation due to process, voltage and temperature (PVT) variation effects on proposed 8-bit precision integrated architecture *@45nm* technology node (a) the different process corner and temperature at power supply = 0.62V (b) for TT corner with power supply variation.**

technique, the *ON* current and delay variations with respect to technology scaling is shown in Fig. 8 (b). We determine the power gate size for a larger slew rate with a lower response time and the same is used for our RECON circuit simulations. The add/sub, shift register logic blocks used in Fig. 4 Is designed using the same power getting technique. These logic blocks are isolated when they are not in use in a specific computation task such as multiply-accumulation or *tanh* calculation. We save 30% static power dissipation with minimal area overhead compared to non power-gated architecture. Based on simulation results and merit, the power gate size should be around 3× large as compared to its standard size to maintain similar performance.

C. OPERATING VOLTAGE

Circuit design with a lower power supply is beneficial as it minimizes power dissipation. These savings are due to scaling in the technology model that has an impact on physical parameters such as mobility (μ_0) and saturation velocity (V_{sat}). However, it comes with an increased propagation delay. Equation 12 shows the relation between the power supply and the propagation delay in the CMOS circuit.

$$T_d = \frac{C_L \cdot V_{dd}}{(V_{dd} - V_t)^{\alpha}} \tag{12}$$

The CMOS logic based circuit design is extracted using cadence virtuoso and post-layout circuit simulation of the proposed design is carried out for current and voltage variation at different process corners and supply variation. The leakage power increases with the increasing supply voltage, and it exceeds the dynamic power for supply greater than 0.7V at 45nm technology. The parameters calculated at three different process corners *FF*, *TT* and *SS* are shown in Fig. 9(a). It observed that the static power is more than the dynamic power in the fast-fast (*FF*) process corner with temperature 85°C. The circuit simulated at different supply voltages with typical-typical (*TT*) process corner and observed variations in simulation results are shown in Fig. 9(b). It seems from the figure that an operating voltage of around 0.62 returns the lowest value for leakage current and dynamic power.

TABLE 5. **Comparison for the proposed design and the state-of-the-art for MAC computation @45*nm* TT process corner for 8-bit precision.**

Parameters	Wallace tree [22]	shift-add [23]	Vedic mult. [21]	Proposed (CORDIC)
Area (μm^2)	838	501	1144	307
St. Power (μW)	3.88	2.86	4.93	4.72
Dy. Power (μW)	496.34	119.53	484.83	139.19
Delay (ns)	6.54	4.27	7.66	3.62
Power-Delay-Product	3246.06	510.39	3713.79	503.87
No. of Clocks	2	8	2	5
Integer	unsigned	unsigned	signed	signed/unsigned

TABLE 6. **Comparison for the proposed design and the state-of-the-art CORDIC-based design for sigmoid activation computation @45*nm* TT process corner for 8-bit precision.**

Configurable AF design simulation Parameters	Magnitude Scaling [25]	Negative Input [26]	Proposed without PG	Proposed with PG
Area (μm^2)	541	483	377	428
St. Power (μW)	176.00	121.70	101.74	77.94
Dy. Power (μW)	299.40	209.50	189.30	189.30
Delay (ns)	6.21	5.93	4.83	5.12
Power-delay-Product	1,859.27	1,242.33	829.10	969.22
No. of Clock	6	6	5	5

D. RECON IN MAC CONFIGURATION

We compare our RECON architecture in MAC configuration with the state-of-the-art designs [20]–[22]. For RECON, the area of the corresponding module performing the MAC is shown in Fig. 6. For the sake of comparison, we have implemented the designs proposed in [20]–[22] with 8-bit precision at 45nm since they were proposed at different technology node. The post-synthesis performance parameters are populated in Table 5.

It can be observed from the Table 5 that our proposed design has the least area as compared to other proposed designs [20]–[22]. While the design in [22] shows the least static and dynamic power, it requires more number of clock cycles along with a larger delay to compute the results as compared to our proposed design. Hence, our proposed design shows a lower *power-delay-product* (PDP) as compared to the design proposed in [22]. To evaluate the overall hardware overheads, we adopt a figure of merit, which is characterized by area, latency, and power product (ALP = area × latency × power) and is 60% less as compared to the architecture proposed in state-of-the-art [22].

An important point to note here is that while other designs focused primarily on unsigned values, our proposed design architecture can support both signed and unsigned numbers. It is particularly relevant considering hardware implementations for neural architecture, as it has a direct impact on the accuracy.

E. RECON IN AF CONFIGURATION

Just as we have evaluated RECON in MAC configuration, we compare our design in *sigmoid* AF configuration with other works proposed in [24], [25] which employed CORDIC-based architecture for AF computation at both 45nm. Results and comparison of proposed architecture with and without power gating and state-of-the-art are shown in Table 6.

TABLE 7. Sigmoid activation function using proposed design and combinational logic-based design [18] at 180*nm* and 45*nm* TT process corner.

		@180nm				@45nm			
Sigmoid AF Bit Precision	**No. of Clocks**	**Combinational Area** (μm^2)	**Dynamic Power** (μW)	**Static Power** (μW)	**Critical Path Delay** (ns)	**Combinational Area** (μm^2)	**Dynamic Power** (μW)	**Static Power** (μW)	**Critical Path Delay** (ns)
8_bit Combinational	1	1637.5	77.5	18.7	7.04	631.7	33.7	19.1	4.76
8_bit CORDIC	5	2283.0	527.4	75.2	6.22	377.5	189.3	101.7	4.38
12_bit Combinational	1	17735.0	804.7	190.0	8.80	4899.5	198.3	110.9	5.10
12_bit CORDIC	5	2492.5	656.0	89.5	7.74	748.9	267.3	137.7	5.00
16_bit Combinational	1	83796.5	35708.5	665.3	15.86	23408.2	8542.7	3960.0	9.53
16_bit CORDIC	5	2973.0	919.7	125.3	8.98	896.2	272.2	149.3	5.22

While, it is to be noted that power gating technique is not applicable for *sigmoid* AF calculation, the overheads of using power gating technique are applicable, as shown in Table 6. It can be observed that our design achieves better numbers for area, power, and delay as compared to other CORDIC based designs [24], [25].

In order to carry out evaluation for our proposed design for AF computation, we also compare it with combinational logic design based AF computation as proposed in [18]. We synthesise both the designs at two technology nodes of 45nm and 180nm at different precisions to have a fair comparison. The reason we are choosing the sigmoid AF configuration because it uses all the components in our proposed architecture as shown by green encircle in Fig. 6.

The combinational logic design-based AF proposed in [18] uses the concept of quantization states to implement non-linear activation functions. Realization of quantization states is done through memory elements and selection using MUX tree and combinational logic gates. The number of quantization states depends upon the bit precision used. Hence, for 8-bit precision, the number of quantization states is equal to $2^8 = 256$. The selection then further requires 8:1 multiplexer tree.

The results of our comparison are shown in Table 7. It can be seen that at low precision, i.e., 8-bit, design proposed in [18] fairs better as compared to RECON in terms of area and power. However, we can observe an exponential increase at higher bit precisions. Moving from 8-bit to 12-bit results in 10× increase in area and power for the design proposed in [18]. The reasons can be ascertained to the fact that for 12-bit precision, number of quantization states is equal to $2^{12} = 4096$ which leads to an exponential increase in the memory elements required. Apart from memory elements, a bigger 16:1 muliplexer tree is also required which further leads to additional overheads. Similarly, for 16-bit precision, the number of memory elements required increases to $2^{16} = 65536$ with the same 16:1 multiplexer tree. Hence, the area increase from 12-bit to 16-bit is just 5×.

It concludes that CORDIC based architecture is an admirable choice for higher precision neural network design architecture. The results shown in Table 7 demonstrate that for the CORDIC-based architecture, the progression in physical parameters such as area and power with respect to the increment in bit precision guarantees much better scaling as compared to [18]. However, the number of*clk* edges required is more in our proposed design as compared to the combinational design. This is obvious considering the same CORDIC block is used iteratively for calculating MAC and activation function in different modes. Hence, CORDIC based design is favorable for applications where the energy and area requirements are less. Hence, the proposed design offers an excellent choice for IoT applications.

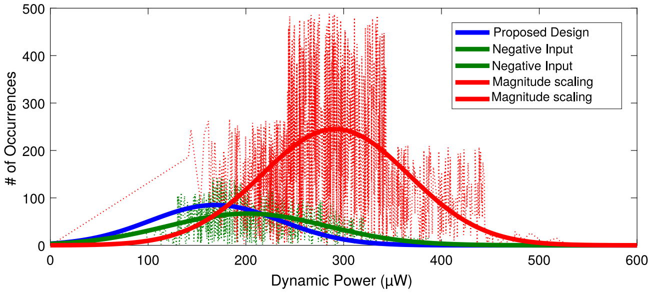


FIGURE 10. 1000 Monte Carlo simulation with process variation and mismatch for mean dynamic power variation of signed 8-bit sigmoid activation function.

F. PROCESS VARIATION AND MISMATCH

At lower technology nodes, process variation and mismatch is also an important issue in addition to power dissipation, stability, and reliability. We have simulated the circuit at *45nm* technology node. The *Monte-Carlo* simulation for dynamic power variation due to process and mismatch is carried out for 1000 samples for the proposed architecture (with *select=0 &* select_af=0) and the state-of-the-art as shown in Fig. 10. The mean dynamic power and standard deviation of the proposed architecture are 189.30μW and 58.2μW respectively. The mean dynamic power of the proposed work is 90% compared to architecture proposed by [15] (shown in green color) and 63% as proposed by [24] (shown in red color). The standard deviation for power variation in the case of RECON is 58.2μW which is less compared to 66.15μW [15] (shown in green color) and 78μW [24] (shown in red color) respectively. It shows that the proposed architecture is more reliable in terms of power variation due to process and mismatch.

VII. CONCLUSION

In this work, we have proposed a resource-efficient and configurable CORDIC-based design, RECON for a neuron

TABLE 8. Accuracy comparison of different DNN architecture at different fixed point bit-precision computation for MNIST and CIFAR-10 data-set.

Fixed Point Data Precision Representation	Training Accuracy (%)		
	LeNet		CaffeNet
	MNIST	CIFAR-10	MNIST
32-bit	99.1	81.7	56.9
16-bit	98.7	81.2	56.8
8-bit	98.2	80.7	56.7
4-bit	97.6	79.6	06.0
2-bit	85.9	48.0	00.1

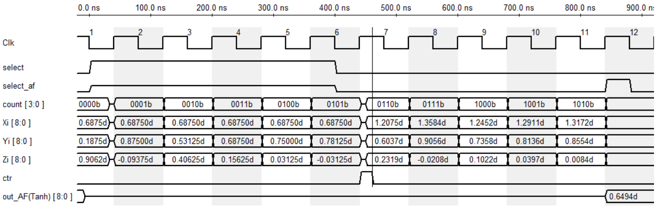


FIGURE 11. Waveform for complete neuron architecture computation with each iteration.

architecture. The CORDIC-based design allows configuration and the same block can compute both MAC and multiple activation functions. Additionally, our proposed design can also compute both signed and unsigned computations. The proposed architecture is area and power efficient as compared to the state-of-the-art designs for both MAC and activation function computation. Using extensive evaluation, we show that our proposed design gives better returns at higher bit precision as compared to other designs.

APPENDIX A

In order to finalize the number of digits after the decimal point in fixed-point number representation, we carry out experiments using different artificial neural networks. Accuracy comparison at different bit-precision is shown in Table 8. It can be seen that their is a very low ($\leq 2\%$) loss of accuracy between usage of 8-bit and 32-bit fixed-point computation. This leads to reduction in memory consumption by factor of 4. Hence, Hence, we have taken 8-bit fixed-point with $\langle 8, 5 \rangle$ representation in computation with fractional for RECON.

APPENDIX B

In order to show the iterative computation the complete neuron architecture, complete waveform is shown in Fig. 11. Here, we have selected the input (X_i), weight (Z_i), bias (Y_i) for the MAC computation and *Tanh* function as an activation function.

ACKNOWLEDGMENT

The authors would like to thank the Special Manpower Development Program Chip to System Design, Department of Electronics and Information Technology (DeitY) under the Ministry of Communication and Information Technology, Government of India for providing necessary research facilities.

REFERENCES

[1] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure–activity relationships," *J. Chem. Inf. Model.*, vol. 55, no. 2, pp. 263–274, 2015.

[2] Du, Ke-Lin, and M. N. S. Swamy, "Neural network circuits and parallel implementations," in *Neural Networks and Statistical Learning*. London, U.K.: Springer, 2019, pp. 829–851.

[3] Y. Umuroglu *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2017, pp. 65–74.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arXiv:1409.1556.

[5] R. Pottathuparambil and R. Sass,"An FPGA-based neural network for computer vision applications," in *Proc. Workshop Comput. Vis. Low Power Reconfig. Architect. (FPL)*, Crete, Greece, 2011, pp. 1–7.

[6] J. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2009, pp. 329–338.

[7] E. Nurvitadhi *et al.*, "Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC," in *Proc. IEEE Int. Conf. Field Program. Technol. (FPT)*, 2016, pp. 77–84.

[8] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[9] F. Kastner, B. Janßen, F. Kautz, M. Hübner, and G. Corradi, "Hardware/software codesign for convolutional neural networks exploiting dynamic partial reconfiguration on PYNQ," in *Proc. IEEE IPDPSW*, 2018, pp. 154–161.

[10] E. Wu, X. Zhang, D. Berman, I. Cho, and J. Thendean, "Compute-efficient neural-network acceleration," in *Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays*, 2019, pp. 191–200.

[11] A. Arthurs, J. Roark, and J. Di, "Ultra-low voltage digital circuit design: A comparative study," in *Proc. IEEE Faible Tension Faible Consommation*, 2012, pp. 1–4.

[12] 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.

[13] L. Vachhani, K. Sridharan, and P. K. Meher, "Efficient CORDIC algorithms and architectures for low area and high throughput implementation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 1, pp. 61–65, Jan. 2009.

[14] M. Qian, "Application of cordic algorithm to neural networks VLSI design," in *Proc. Multiconf. Comput. Eng. Syst. Appl.*, vol. 1, 2006, pp. 504–508.

[15] M. Ercegovac, D. Kirovski, and M. Potkonjak, "Low-power behavioral synthesis optimization using multiple precision arithmetic," in *Proc. IEEE Design Autom. Conf.*, 1999, pp. 568–573.

[16] S. Anwar, K. Hwang, and W. Sung, "Fixed point optimization of deep convolutional neural networks for object recognition," in *Proc. IEEE ICASSP*, 2015, pp. 1131–1135.

[17] A. S. Abraham and S. Anand, "An ASIC design of an optimized multiplication using twin precision," in *Proc. IEEE ICICCS*, 2017, pp. 455–461.

[18] S. J. V. Rani and P. Kanagasabapathy, "Multilayer perceptron neural network architecture using VHDL with combinational logic sigmoid function," in *Proc. IEEE Int. Conf. Signal Process. Commun. Netw.*, 2007, pp. 404–409.

[19] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized mlp architectures of neural networks," *Int. J. Artif. Intell. Expert Syst.*, vol. 1, no. 4, pp. 111–122, 2011.

[20] A. S. K. Vamsi and S. Ramesh, "An efficient design of 16 bit MAC unit using vedic mathematics," in *Proc. IEEE Int. Conf. Commun. Signal Process. (ICCSP)*, 2019, pp. 319–322.

[21] H. Wong, V. Betz, and J. Rose, "Comparing FPGA vs. custom CMOS and the impact on processor microarchitecture," in *Proc. 19th ISFPGA*, 2011, pp. 5–14.

[22] G. B. Joseph and R. Devanathan, "Algorithms for multiplierless multiple constant multiplication in online arithmetic," *Circuits Syst. Signal Process.*, vol. 37, no. 11, pp. 5127–5142, 2018.

[23] G. Raut, V. Bhartiy, G. Rajput, S. Khan, A. Beohar, and S. K. Vishvakarma, "Efficient low-precision cordic algorithm for hardware implementation of artificial neural network," in *Proc. Int. Symp. VLSI Design Test*, 2019, pp. 321–333.

[24] V. Tiwari and N. Khare, "Hardware implementation of neural network with sigmoidal activation functions using CORDIC," *Microprocess. Microsyst.*, vol. 39, no. 6, pp. 373–381, 2015.

[25] M. Alçın, İ. Pehlivan, and İ. Koyuncu, "Hardware design and implementation of a novel ANN-based CHAOTIC generator in FPGA," *Optik*, vol. 127, no. 13, pp. 5500–5505, 2016.

[26] M. Yuvaraj, B. J. Kailath, and N. Bhaskhar, "Design of optimized MAC unit using integrated VEDIC multiplier," in *Proc. Int. Conf. Microelectron. Devices Circuits Syst. (ICMDCS)*, 2017, pp. 1–6.

[27] W. Xu, Z. Zhang, X. You, and C. Zhang, "Efficient deep convolutional neural networks accelerator without multiplication and retraining," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2018, pp. 1100–1104.

[28] D. A. Gudovskiy and L. Rigazio, "ShiftCNN: Generalized low-precision architecture for inference of convolutional neural networks," 2017. [Online]. Available: arXiv:1706.02393.

[29] H. Kim, Q. Chen, T. Yoo, T. T.-H. Kim, and B. Kim, "A 1–16b precision reconfigurable digital in-memory computing macro featuring column-MAC architecture and bit-serial computation," in *Proc. IEEE 45th Eur. Solid-State Circuits Conf. (ESSCIRC)*, 2019, pp. 345–348.

[30] D. Nguyen, D. Kim, and J. Lee, "Double MAC: Doubling the performance of convolutional neural networks on modern FPGAs," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, 2017, pp. 890–893.

[31] J. Kadmon and H. Sompolinsky, "Transition to chaos in random neuronal networks," *Phys. Rev. X*, vol. 5, no. 4, 2015, Art. no. 041030.

[32] M. Wedlake and H. L. Kwok, "A CORDIC implementation of a digital artificial neuron," in *Proc. IEEE Pac. Rim Conf. Commun. Comput. Signal Process. (PACRIM)*, vol. 2, 1997, pp. 798–801.

[33] C.-H. Chang, H.-Y. Kao, and S.-H. Huang, "Hardware implementation for multiple activation functions," in *Proc. IEEE Int. Conf. Consum. Electron. Taiwan (ICCE-TW)*, 2019, pp. 1–2.

[34] S. Gomar, M. Mirhassani, and M. Ahmadi, "Precise digital implementations of hyperbolic TANH and sigmoid function," in *Proc. IEEE 50th Asilomar Conf. Signals Syst. Comput.*, 2016, pp. 1586–1589.

[35] 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.

[36] 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.

[37] C. B. Bidhul, N. Hampannavar, S. Joseph, P. Jayakrishnan, and S. Kumaravel, "Comparison of architectures of a coarse-grain reconfigurable multiply-accumulate unit," in *Proc. IEEE Int. Conf. Green Comput. Commun. Conservation Energy (ICGCE)*, 2013, pp. 225–230.

[38] P.-Y. Chou *et al.*, "Near-threshold cordic design with dynamic circuitry for long-standby IoT applications," in *Proc. 31st IEEE Int. Syst. Chip Conf. (SOCC)*, 2018, pp. 250–253.

[39] J. Volder, "The CORDIC computing technique," presented at the the Western Joint Comput. Conf., Mar. 1959, pp. 257–261.

[40] T. Lang and E. Antelo, "CORDIC-based computation of ARCCOS and ARCSIN," in *Proc. IEEE Int. Conf. Appl. Specific Syst. Architect. Process.*, 1997, pp. 132–143.

[41] 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.

[42] T.-Y. Sung and H.-C. Hsin, "Fixed-point error analysis of CORDIC arithmetic for special-purpose signal processors," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 90, no. 9, pp. 2006–2013, 2007.

[43] N. Sorokin, "Implementation of high-speed fixed-point dividers on FPGA," *J. Comput. Sci. Technol.*, vol. 6, no. 1, pp. 8–11, 2006.

[44] *Synopsys*. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler-graphical.html

[45] *Communities Mentor*. [Online]. Available: https://communities.mentor.com/docs/doc-3114

[46] *Cadence*. [Online]. Available: https://www.cadence.com/en_us/home/tools/custom-ic-analog-rf-design/circuit-design/virtuoso-schematic-editor.html

GOPAL RAUT (Graduate Student Member, IEEE) received the B.Engg. degree in electronic engineering and the M.Tech. degree in VLSI design from the G. H. Raisoni College of Engineering Nagpur, India, in 2015. He is currently pursuing the Ph.D. degree with the Electrical Engineering Department, Indian Institute of Technology Indore, where he is currently with the Nanoscale Devices, VLSI Circuit and System Lab, Electrical Engineering Department. His research focus is compute-efficient and configurable VLSI circuit design for IoT applications.

SHUBHAM RAI (Graduate Student Member, IEEE) received the B.Engg. degree in electrical and electronic engineering and the M.Sc. degree in physics from the Birla Institute of Technology and Science Pilani, India, in 2011. He is currently pursuing the Ph.D. degree with Technische Universität, Dresden, Germany. His research focus is on circuit design for reconfigurable nanotechnologies and their logical applications.

SANTOSH KUMAR VISHVAKARMA (Member, IEEE) received the Ph.D. degree from the Indian Institute of Technology Roorkee, India, in 2010. He is currently an Associate Professor with the Department of Electrical Engineering, Indian Institute of Technology Indore, India, where he is leading the Nanoscale Devices and VLSI Circuit and System Design Lab. From 2009 to 2010, he was with the University Graduate Center, Kjeller, Norway, as a Postdoctoral Fellow under the European Union Project "COMON." His current research interests include nanoscale devices, reliable SRAM memory designs, and configurable circuits design for IoT applications.

AKASH KUMAR (Senior Member, IEEE) received the joint Ph.D. degree in electrical engineering in embedded systems from the Eindhoven University of Technology, Eindhoven, and the National University of Singapore, Singapore, in 2009. He is currently a Professor with Technische Universität Dresden, Germany, where he is directing the Chair for Processor Design. From 2009 to 2015, he was with the National University of Singapore. His current research interests include design, analysis, and resource management of low-power and fault-tolerant embedded multiprocessor systems.