Circuits, Systems, and Signal Processing (2023) 42:6089–6115
https://doi.org/10.1007/s00034-023-02387-2

Designing a Performance-Centric MAC Unit with Pipelined Architecture for DNN Accelerators

Gopal Raut[1] · Jogesh Mukala[1] · Vishal Sharma[2] · Santosh Kumar Vishvakarma[1]

Received: 1 August 2022 / Revised: 11 April 2023 / Accepted: 12 April 2023 /
Published online: 16 May 2023

Abstract

In order to improve the performance of deep neural network (DNN) accelerators, it is necessary to optimize compute efficiency and operating frequency. However, the implementation of contemporary DNNs often requires excessive resources due to the heavy multiply-and-accumulate (MAC) computations. In this work proposes a MAC unit designed with a *Co-ordinate Rotation DIgital Computer* (CORDIC)-based architecture, which is both power and area-efficient for 8-bit and higher-bit precision. The CORDIC-based designs are typically associated with low throughput. To address this issue, a performance-centric pipelined architecture is investigated that increases throughput. The study conducts a detailed Pareto analysis of accuracy variation at different precision levels and required pipeline stages to achieve high performance. The proposed MAC unit's post-synthesis results at the 45*nm* technology node are provided, and performance is evaluated on a deep neural network using Vertex-7 FPGA board. The proposed fixed-point MAC architecture is scalable for all bit-precision and flexible for the decimal point implication. The study finds that the proposed Fixed $Q_{3.5}$ precision with five pipeline stage-based MAC shows better performance metrics compared to the recursive CORDIC-based MAC design. The proposed MAC design has a lower area-delay-product (ADP) which is 1.13×, and higher throughput of 2.73× compared to the recursive CORDIC-based MAC. The study evaluated the performance

✉ Santosh Kumar Vishvakarma
skvishvakarma@iiti.ac.in

Gopal Raut
gopalraut05@gmail.com

Jogesh Mukala
jogeshkumarmukala@gmail.com

Vishal Sharma
vishalfzd@gmail.com

1 Department of Electrical Engineering, IIT Indore, Indore, India

2 Nanyang Technological University, Singapore, Singapore

of the proposed MAC unit using the fully connected NN for the MNIST dataset and found that the throughput 1.89× better compared to the conventional MAC-based design.

Keywords Deep neural networks (DNNs) · Multiply-and-accumulate (MAC) unit · CORDIC-based architecture · Performance-centric pipelined architecture · Throughput

1 Introduction

In the last decade, deep learning has demonstrated remarkable progress [16]. Despite its potency, the cost of implementing a deep learning model can be exorbitant [31]. One of the most widely used learning networks is deep neural networks (DNNs), which have numerous applications in solving nonlinear detection problems, such as lane detection, pattern recognition, fault detection, and industry monitoring [17, 21]. Nonetheless, in edge devices where low power, minimum area, and high throughput are imperative, it is crucial to accelerate neural network inference while minimizing computational resources and reducing on-chip power consumption [3, 27, 40].

DNNs demand extensive multiply-and-accumulates (MACs) per image. To implement DNN accelerators, GPUs, FPGAs, and ASICs are used as platforms. GPU-based solutions exhibit good parallelism but suffer from inefficient resource utilization, resulting in high power consumption. On the other hand, FPGAs offer configurable architecture and superior resource utilization, making them a preferred choice for high-precision computation, given their digital signal processing (DSP) elements with fixed dynamic bit width [32]. For example, the Xilinx FPGA mainly utilizes `DSP48` blocks that come with 18-bit input precision and 48-bit output precision hard-coded hardware architecture [29]. However, in lower precision ($<$12-bit), the MAC architecture implementation will consume a complete 18-bit precision DSP block, leading to inefficient resource utilization and inherently higher power consumption. In contrast, CLBs-based implementations make better use of hardware resources. On the other hand, ASICs feature specialized architecture, resulting in efficient resource utilization and minimum power consumption.

DNNs have a resource-intensive architecture due to their high computational demands, and the throughput performance of DNNs is also a critical parameter that requires parallelism [19]. In edge-AI and mobile applications, area and power are restricted. To enhance the architecture, approximate computing techniques have been extensively researched in the field of DNN accelerators. Approximation enables reducing the area and power consumption of the DNN accelerator with minimal accuracy loss. To reduce on-chip area and power consumption while retaining high throughput, recursive Coordinate Rotation DIgital Computer (CORDIC)-based architecture has been investigated in [28]. This architecture employs the recursive CORDIC algorithm for MAC implementation, which performs computation iteratively. The CORDIC algorithm employs shift-and-add operation, which demands minimal hardware implementation area and power. The iterative CORDIC architecture necessitates N+1 clocks for signed N-bit precision computation, with each iterative calculation being mutually

exclusive [27]. Hence, the pipeline architecture provides an opportunity to improve the throughput performance at the expense of area and power overhead.

Several essential hardware parameters must be considered when developing deep learning networks, such as high main memory bandwidth, low latency, and fast registers [5]. Efficient processing engines in DNNs require multiple-and-accumulate units and associated register banks, which demand more hardware resources. Therefore, it is crucial to design DNN architectures based on the available resources. Layer-reused and approximation in arithmetic are suitable choices for constrained hardware resources, but they may result in accuracy and throughput loss. However, if resources are not a constraint, high-precision architecture, and a parallel processing engine can be adopted to achieve better throughput. Arithmetic precision, MAC operation, data parallelism, bit-quantization, and feature complexity are the primary determinants of hardware selection for implementation [34]. For edge-AI applications, the parallel computation can provide high throughput at the cost of increased hardware usage [4]. In contrast, IoT applications demand the least amount of hardware and limited power. Therefore, for IoT applications, the recursive computation can benefit from iterative MAC architecture [28], while pipeline MAC design can improve performance with large throughput for edge-AI applications.

Efforts to improve MAC performance have been the focus of research in the last decade. Efficient design techniques such as libraries [22], FINN [34], ShiftCNN [9], power reduction methods [5], and Sub-MACs [20] have been developed to achieve this goal. The implementation of CORDIC-based MAC has a high computational delay due to its recursive architecture, resulting in lower throughput [28]. Therefore, this article addresses high-throughput MAC architecture for DNN applications and proposes an enhanced performance CORDIC algorithm-based MAC hardware efficient pipeline architecture. To reduce the area and power overheads associated with the pipeline architecture, we provide a Pareto study to determine the necessary number of CORDIC pipeline stages. As the DNN algorithm is error-resilient, we conducted a performance-centric analysis to determine the optimal number of pipeline stages that would achieve area and power efficiency while maintaining accuracy. Our evaluation revealed that a minimum number of pipeline stages could significantly reduce the required area for implementation. Outlined below are the key contributions of our research:

- A performance-centric Pareto analysis is conducted to determine the optimal number of pipeline stages for the CORDIC pipelined architecture, which typically incurs area and power overhead. Our analysis focused on the MNIST and CIFAR-10 datasets, and we evaluated the impact of reducing the number of pipeline stages on overall output accuracy.
- Fixed $Q_{p.q}$ arithmetic is employed to evaluate our results and facilitate hardware implementation and Pareto study conducted to identify the optimal values of $\mathtt{p}$ and $\mathtt{q}$ for improved accuracy and hardware performance.
- An error metrics is evaluated for the proposed architecture with a reduced number of pipeline stages, precisely five stages. In addition, we implemented a fully connected DNN with dimensions $\mathtt{196:64:32:32:10}$ using the proposed MAC on an FPGA and analyzed the physical performance, throughput, and accuracy.
- The physical performance parameters are extracted for the proposed MAC with five pipeline CORDIC stages using 45nm technology and compared them to other

relevant metrics. Additionally, a post-layout Monte Carlo simulation is conducted to assess the impact of process variation and mismatch on power consumption.

The study presents an enhanced performance MAC and based DNN implementation. The DNN network is trained until maximum accuracy is achieved, and it shows a less than 1% accuracy loss when compared to accurate Tensor libraries for MNIST and CIFAR-10 datasets. The research also includes a Pareto study that focuses on variable notation for signed 9-bit Fixed $Q_{p.q}$ arithmetic, where 'p' represents the number of integer bits, 'q' indicates fractional bits, and one signed bit in a fixed-point 9-bit precision format. Furthermore, the study shows that fully connected NN implemented on Vertex-7 performs nearly two times better than Xilinx MAC-based computation. The physical parameters of the design are extracted for the 45nm node and compared with previous work. The experimental results show that the proposed pipeline architecture design has nearly five times higher throughput than the iterative architecture. Additionally, the area-latency-power (ALP) is 12% better than the iterative CORDIC and 21% better than other best state-of-the-art designs.

The remainder of this article is organized as follows. Section 2 provides a brief on related work and the motivation behind our research. Section 3 introduces CORDIC optimization and MAC realization. We discuss the tuning parameters for improving the performance of MAC computation in detail in Sect. 4. Section 5 describes the experimental setup, followed by the presentation of simulation results and discussion in Sect. 6. Finally, we conclude our findings in Sect. 7.

2 Related Work and Motivation

Deep neural networks, including fully connected and convolutional neural networks, have shown great promise in image recognition and classification tasks. However, the MAC computation used in these networks is resource-intensive and consumes significant power, especially for higher bit-precision computation. Previous studies have proposed efficient custom MAC designs for both ASICs and FPGAs. While ASIC-based implementation of DNN accelerators is efficient in terms of area, power, and throughput, it suffers from reconfigurability and development process limitations [39]. State-of-the-art techniques have trade-offs between physical performance parameters, throughput, and accuracy [13, 34]. Additionally, hardware implementation of accelerators faces a challenge with bandwidth limitations. Therefore, research has analyzed performance parameters for multi-bit precision (8, 16, 24, or 32-bit) design in hardware-based acceleration [28, 40].

The approach with iterative computation with shift-and-add multiplication method simplifies the logic arithmetic complexity and requires relatively fewer hardware resources [9]. Further, the Vedic multiplier-based efficient design of MAC unit for lower arithmetic precision has also been presented [35]; however, it is not scalable for high-precision architectures due to its increasing critical path and propagation delay. The modified Booth's algorithm for multiplication has been investigated to save resource utilization [5]. In addition, a Wallace tree-based MAC design has been implemented and analyzed for physical performance parameters [15]. However, its

architecture design using AND and OR planes becomes more complex with increasing bit precision. Another approach is an in-memory computing architecture that has better physical parameters but limits bit resolution [37]. Besides all, a CORDIC-based MAC design has been proposed [28], but it suffers from low throughput performance due to its iterative computations. An n-bit barrel shifter and n-1 additions are required for n-bit precision computation with iterative design [28]. Therefore, this article has introduced the CORDIC algorithm's pipeline architecture and discussed MAC for DNN in high-throughput applications.

Several techniques have been explored to improve power and energy efficiency in MAC computation, such as the use of an approximate partial product accumulation tree presented in [14], the application of integral stochastic computing for more straightforward arithmetic in DNN in [2], and the implementation of inaccurate logic compressors in the multiplier that show promising results, as discussed in [36]. Approximate computation is preferred for error-resilient applications, and an approximate multiplier in MAC computation can reduce circuit delay and on-chip power consumption, as reported in [8]. Additionally, the weight-sharing scheme is effective in reducing MAC computational complexity and constant storage capacity. A high-performance array multiplier with reversible logic structure has been developed, increasing the throughput performance, as reported in [38]. However, these techniques may lack generality for multi-precision signed/unsigned computation. The use of nested RNS (NRNS) in DNN for object detection is presented in [23], leading to high clock frequency with less area on FPGA. The author decomposed a MAC unit into 4 bits and realized it by LUTs, resulting in significantly improved performance-power efficiency in hardware implementation. In addition, stochastic computing MAC with optimization techniques has been proposed for DNN applications, showing better results than conventional techniques, as discussed in [30]. Nevertheless, the architecture's complexity may increase with precision scaling.

In addition to optimizing the MAC, there have been advancements in neural network architecture design. The fully connected layer circuit's memory bandwidth bottleneck has been addressed by proposing a sequential-input parallel-output circuit that uses neuron pruning to reduce weight memory and improve memory access speed and power consumption [6]. The computational and storage complexity of DNN has also been minimized by using Boolean logic for training [24]. Additionally, a comprehensive study on energy-efficient DNN implementation on micro-AI platforms has been conducted to understand recent developments in this area [19]. The floating-point arithmetic in MAC implementation can improve the system's accuracy, but it comes at the expense of increased area and power consumption [33]. However, it is possible to achieve accuracy comparable to floating-point arithmetic by using fixed-point arithmetic with a simple quantization method, which utilizes powers of 2 as scale factors in an optimal bit-width optimization algorithm [25]. The authors evaluated the performance accuracy at different bit-precision using fixed-point arithmetic. They found that an 8-bit precision computation shows significant performance improvement compared to 16-bit precision while saving 4× memory bandwidth at the cost of less than 1.5% accuracy loss [28].

The study also includes a comparison between different MAC architectures, such as array, boot, Wallace, Vedic, DSP-based, iterative CORDIC, and approximate MAC

Table 1 Relative comparison of state-of-the-art MAC performance parameters for VLSI implementations based on fixed-point arithmetic

MAC design architecture	Performance/speed	Power	Design complexity	Scalability for high precision	Accuracy in DNN inference
Array multiplier [8]	Low	High	Less	High	Average
Wallace Multiplier [15]	Medium	Medium	Medium	Medium	Average
Booth Multiplier [5]	Medium	Medium	Medium	Medium	Good
Vedic Multiplier [35]	Medium	High	Medium	Complex	Average
DSP Architecture [29]	High	High	Less	Medium	Good
Iterative CORDIC [28]	Very Low	Low	Very Less	High	Average

units, with DNN applications, as summarized in Table 1. Our research aimed to address the trade-off between area, power, and throughput for DNN accelerators by assessing the performance of an improved CORDIC-based MAC unit. We employed pipeline CORDIC stages to address the throughput issue and evaluated the optimal number of stages needed to manage area overhead. By varying the pipeline stages, we were able to understand the trade-off between accuracy and the number of pipeline stages required. Our study includes a Pareto analysis to determine the necessary number of CORDIC pipeline stages for reducing area and power overhead, given that pipeline architecture entails area and power overhead.

3 Multiply-and-accumulate (MAC) Architecture Using CORDIC Computation

This section has discussed the MAC architecture using CORDIC computation. The section includes sub-sections on background of CORDIC algorithm, MAC with iterative CORDIC, and a proposed architecture for an enhanced performance MAC design through pipelining. Further, we have discussed a Pareto study that was conducted to fix the optimum pipeline stage in order to achieve a performance-centric design.

3.1 Co-ordinate Rotation Digital Compute Algorithm (CORDIC)

A CORDIC is an iterative algorithm that calculates the two-dimensional vector rotation in the different coordinate systems to derive a wide range of mathematical relationships [1]. The basic CORDIC iteration describes a rotation of a plane vector (X_n,Y_n) to (X_{n+1}, Y_{n+1}). Z_n keeps track of the rotation angle, i.e., α_n. The real rotation of a plane vector follows a circular path instead of a linear one, making it tricky to calculate the coordinates. So, to easily find out (X_{n+1}, Y_{n+1}) from (X_n,Y_n), pseudo-rotations are used as they follow a linear path. Similarly, the CORDIC uses a pseudo-rotation calculation, a scaled version of real rotation [28]. The coordinate calculation equations for pseudo-rotation are given as follows:

$$\begin{aligned} \mathbf{X}_{(n+1)} &= \mathbf{X}_n - \mathbf{Y}_n \times tan\ \alpha_n \\ \mathbf{Y}_{(n+1)} &= \mathbf{Y}_n + \mathbf{X}_n \times tan\ \alpha_n \\ \mathbf{Z}_{(n+1)} &= \mathbf{Z}_n - \alpha_n \end{aligned} \tag{1}$$

The CORDIC Eq. 1 shows pseudo-rotation computation for the trigonometric calculations. Furthermore, pseudo-rotation coordinate equations converged to linear CORDIC form as shown in Eq. 2 [28].

$$\begin{aligned} \mathbf{X}_{n+1} &= \mathbf{X}_n - m \times d_n \times \mathbf{Y}_n \times 2^{-n} \\ \mathbf{Y}_{n+1} &= \mathbf{Y}_n + \mathbf{d}_n \times \mathbf{X}_n \times 2^{-n} \\ \mathbf{Z}_{n+1} &= \mathbf{Z}_n - d_n \times \mathbf{E}_n \\ d_n \in \{-1, +1\}; \quad n &= 0, 1, 2, ..., N \end{aligned} \tag{2}$$

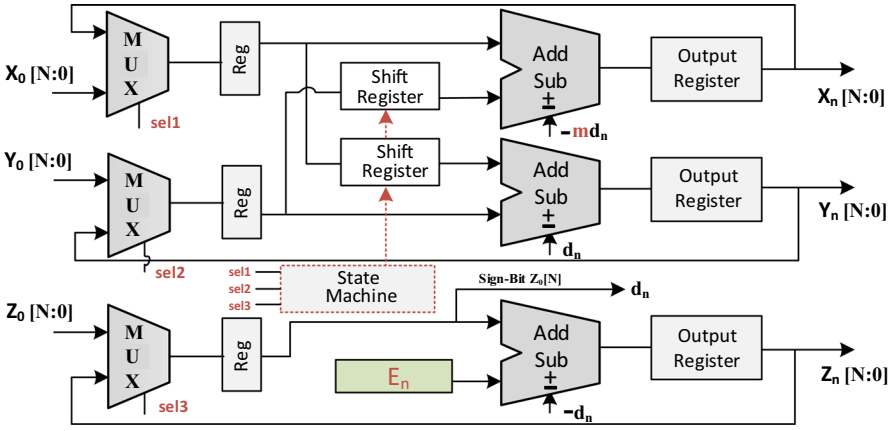


Fig. 1 Signed N-bit precision recursive CORDIC architecture realize Multiply-Accumulate computation in linear mode ($m = 0$ & $\mathbf{E}_i = 2^{-i}$) [28]

here mode $m \in \{1, 0, -1\}$ indicates a circular, linear, and hyperbolic coordinate system, respectively. The d_n signal generates from $\mathtt{sign}(\mathbf{Z}_n)$ which give the direction for either addition or subtraction. Further, $\mathbf{E}_n$ is the memory elements and for different modes is equal to $\mathtt{tan}^{-1}(2^{-n})$, 2^{-n}, and $\mathtt{tanh}^{-1}(2^{-n})$ for circular, linear and hyperbolic rotation computation mode, respectively, for n^{th} iterations.

One can observe from Eq. 2 that CORDIC hardware implementation requires an addition/subtraction, bit-shift operation, and memory elements. In linear mode, set as $m \in \{0\}$ and $\mathbf{E}_n$ is 2^{-n}, i.e., pre-calculated memory element at each iteration.

The equation presented in Eq. 2 can be realized using a generic iterative CORDIC hardware architecture, as illustrated in Fig. 1. Although this implementation requires fewer hardware resources, it suffers from low throughput due to its iterative nature, where the output of one iteration becomes the input for the next. To address this issue, a pipeline architecture has been employed. By simplifying Eq. 2, the hardware resources required for the pipeline architecture have been reduced. It should be noted that the hardware architecture used can perform all modes of CORDIC operations, but for our purpose, we have selected the linear mode of operation with $\mathtt{m} = 0$ and $\mathbf{E}_i = 2^{-i}$. To express the revised form of Eq. 2 in the linear mode of operation, we use the value of $\mathtt{m} = 0$, resulting in the new form of the equation, Eq. 3c. This revised equation is well-suited for performing the MAC computation, which is a critical component in DNN accelerators. We further discuss the linear mode of operation and its advantages for MAC computation.

$$\mathbf{X}_{n+1} = \mathbf{X}_n - 0 \times d_n \times \mathbf{Y}_n \times 2^{-n} \quad \Rightarrow \quad \mathbf{X}_{n+1} = \mathbf{X}_n \tag{3a}$$

$$\mathbf{Y}_{n+1} = \mathbf{Y}_n + d_n \times \mathbf{X}_n \times 2^{-n} \tag{3b}$$

$$\mathbf{Z}_{n+1} = \mathbf{Z}_n - d_n \times 2^{-n}$$

$$d_n \in \{-1, +1\} \ \& \quad \mathtt{n} \ = \ 0, 1, 2, ..., N \tag{3c}$$

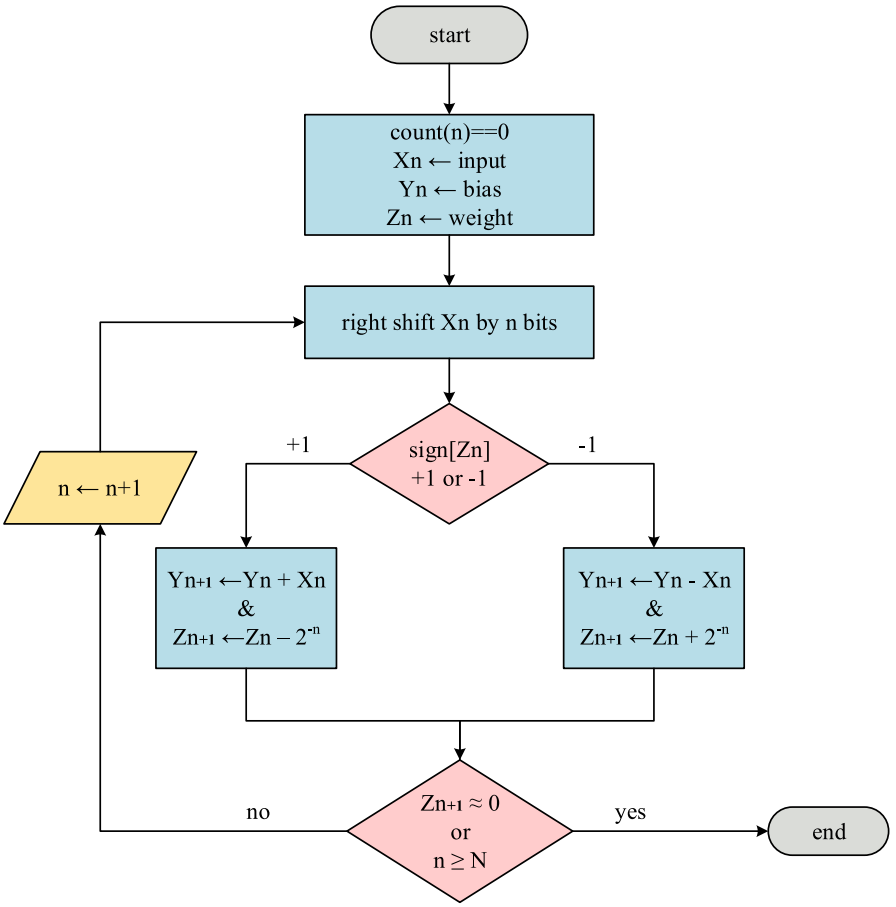


Fig. 2 The CORDIC-based MAC computation design flow, which stops when the conditions $Z_{n+1} \approx 0$ or $n \geq N$ are met

3.2 Iterative CORDIC-based multiply-and-accumulate (MAC) Computation

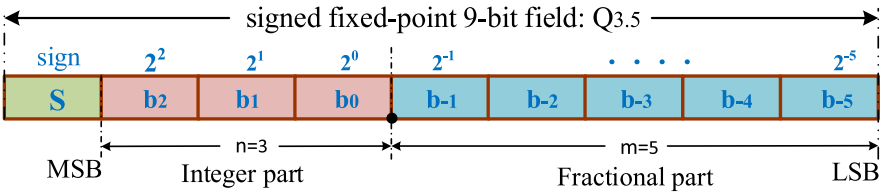
Computing architecture optimization for multiply and accumulate operation has been significantly explored. The algorithms can be either implemented iterative or in a parallel manner. The iterative nature consumes fewer resources and minimum power but often induces delay and affects the throughput. Sequential logic generally involves repetitions of single or multiple steps. Thus, on the contrary, the parallel or pipelined nature reduces latency, thus increasing the throughput at the cost of increased resource utilization [18]. Primarily, Neuron processing engine consists of a MAC unit followed by nonlinear transformation, whereas the arithmetic relation between input and output of a MAC in the l^{th} layer is given by Eq. 4.

$$\mathbf{x}^l = \sum_{j=1}^{J} \mathbf{w}_j^l \cdot \mathbf{x}_j{}^{(l-1)} + \mathbf{b}^l \tag{4}$$

Here, $\mathbf{x}_j$ is the input feature map, $\mathbf{w}_j$ is the weight for the j^{th} input, and b_n is a bias of the corresponding neuron in l^{th} layer. Note that the output of the layer $(l-1)^{\text{th}}$ is the input to the layer l.

CORDIC algorithm architecture uses multiplexer, shift-register, adder/subtractor, and memory constants. However, the proposed architecture realized for Eq. 3c where the output at $\mathbf{Y}_{n+1}$ performs the accumulation of Y_n and the shifted version of $\mathbf{X}_n$ for n^{th} iteration. As $\mathbf{Z}_n$ keeps track of the rotation angle, $\mathbf{Z}_n \rightarrow 0$ ensures final evaluation of the coordinates. Moreover, in CORDIC architecture, operations perform using a shift-and-add method, which allows maximum $N+1$ shifts for N-bit input precision. Further, accuracy has been evaluated for the lower to higher bit precision. We observed that the higher bit precision than 8-bit has not significantly helped improve the accuracy in simple and moderate feature classifications. In the CORDIC, the iterative computation has to be performed until the output at $\mathbf{Z}_n \rightarrow 0$, which means the number of iterations is purely dependent on $\mathbf{Z}_{in}$ and the maximum is $N+1$, where N is the input bit-precision. The MAC computation has been realized using CORDIC in a linear mode of operation [28]. From Eq. 3c, it can see that the output at $\mathbf{Y}_{n+1}$ is an accumulation of $\mathbf{Y}_n$ and the $\mathbf{X}_n$ shifted version for n^{th} iteration and the process will continue till the $\mathbf{Z}_n$ reaches to the zero. In order to perform the MAC operation CORDIC input parameters $\mathbf{X}_n$, $\mathbf{Y}_n$ and $\mathbf{Z}_n$ are taken as `input`, `bias` and corresponding `weight`, respectively. It is noted that right shift and accumulate of input $\mathbf{X}_n$ till $\mathbf{Z}_n \rightarrow 0$ performers the multiplication between input $\mathbf{X}_n$ and $\mathbf{Z}_n$ as shown in Eq. 3c. Therefore, the computation insights multiply-accumulate operation at $\mathbf{Y}_{n+1}$ is valid when the $\mathbf{Z}_0 \rightarrow 0$. The implementation technique is similar to the standard shift-and-add for multiplication except for the bit-shift direction, which supports both signed and unsigned calculations. In [28], results for MAC operation have reported using iterative architecture at the cost of throughput loss. The computation requires n clock cycles to evaluate the final output for N-bit precision operation.

The $\mathbf{X}_0$ have taken as a input feature, and $\mathbf{Z}_0$ will be the corresponding weight, whereas $\mathbf{Y}_0$ is the neurons' bias constant in the evaluation of CORDIC Eq. 3c. After n^{th} iteration, the final form of the equations is shown in Eq. 5 [28]. An n-bit precision computation needs to iterate a set of equations until we get $\mathbf{w}_0 \Rightarrow 0$ or a maximum of `n` iterations [28]. Thus, implementation is efficient for area and power at the cost of lower throughput. Therefore, we have optimized the CORDIC architecture firstly for MAC operation for lower area utilization. Secondly, we have designed a pipelined architecture for high throughput performance. Overall CORDIC equations in linear mode perform MAC computation where simplified form is shown in Eq. 3c. Further, a CORDIC-based MAC evaluation flowchart in a linear mode of operation is shown in Fig. 2. It observed that $\mathbf{X}_n$ right-shift and add iteratively with itself until $\mathbf{Z}_n \rightarrow 0$ at $\mathbf{Y}_n$ port. Finally, computation performs multiplication operations between $\mathbf{X}_n$ and $\mathbf{Z}_n$. Each CORDIC iteration produces 1-bit accuracy; it is important to notice that output accuracy depends on the input features' precision, i.e., n-bit precision CORDIC produces an error around $2^{-\text{n}}$ when $\mathbf{Z}_n$ reaching to zero.



(a) Signed fixed-point $Q_{3.5}$ precision representation used for the calculation

Variable	n=0 (1st stage)	n=1 (2nd stage)
$X_{n+1} = X_n$	$X_1 = 000.10101_2 = 0.65625_{10}$	$X_2 = 000.10101_2 = 0.65625_{10}$ X1 right-shift by n bits
$Y_{n+1} = Y_n + d_n * X_n * 2^{-n}$	$Y_1 = 000.10101_2 = 0.65625_{10}$	$Y_2 = 000.10101_2 + d1 * 000.01010_2 1$ lost bit $= 000.11111_2/0.96875_{10}$
$Z_{n+1} = Z_n - d_n * 2^{-n}$	$Z_1 = 000.00011_2 = 0.09375_{10}$	$Z_2 = 000.00011_2 - d1 * 000.10000_2 = -000.01101_2/-0.40625_{10}$

(b) Calculation for i^{th} iteration approach in linear mode

Fig. 3 Data representation with decimal point implication used for arithmetic calculation

$$\mathbf{X}_n = \mathbf{X}_0 \quad \Rightarrow \quad \mathbf{X}_{out} = \mathbf{X}_{in} \tag{5a}$$

$$\mathbf{Y}_n = \mathbf{Y}_0 + \mathbf{X}_0 * \mathbf{Z}_0 \quad \Rightarrow \quad \mathbf{Y}_{out} = \mathbf{Y}_{in} + \mathbf{X}_{in} * \mathbf{Z}_{in} \tag{5b}$$

$$\mathbf{Z}_n \simeq 0 \tag{5c}$$

3.3 Pipeline Architecture for Enhanced MAC Performance

The computations in the iterative CORDIC architecture are independent of each other. Further, architecture is area and power-efficient. However, the downside is that it requires N+1 clock cycles to produce the final output for an N-bit input, which is not ideal for achieving high throughput. To overcome this, pipeline stages have been introduced to improve the MAC performance at the cost of area and power overhead. This design is especially suitable for edge-AI and mobile applications where suffers from power and area overhead. Therefore, Pareto's points have been evaluated to optimize the performance, and the results are discussed in a subsequent section. A conventional n-stage pipeline architecture would require $n\times$ hardware compared to the iterative architecture. However, the proposed pipelined architecture eliminates need of n pipeline stage that obtained through Pareto analysis and also reduce the need for multiplexers, feedback registers, and barrel shifter blocks, making it more area-efficient.

The design and implementation of the CORDIC-based MAC computation utilize the signed fixed-point $Q_{3.5}$ arithmetic notation, as illustrated in Fig. 3a. The arithmetic computation for the first CORDIC stage is shown in Fig. 3b. In this computation, the most significant bit (MSB) of the fixed-point represents the signed bit used for

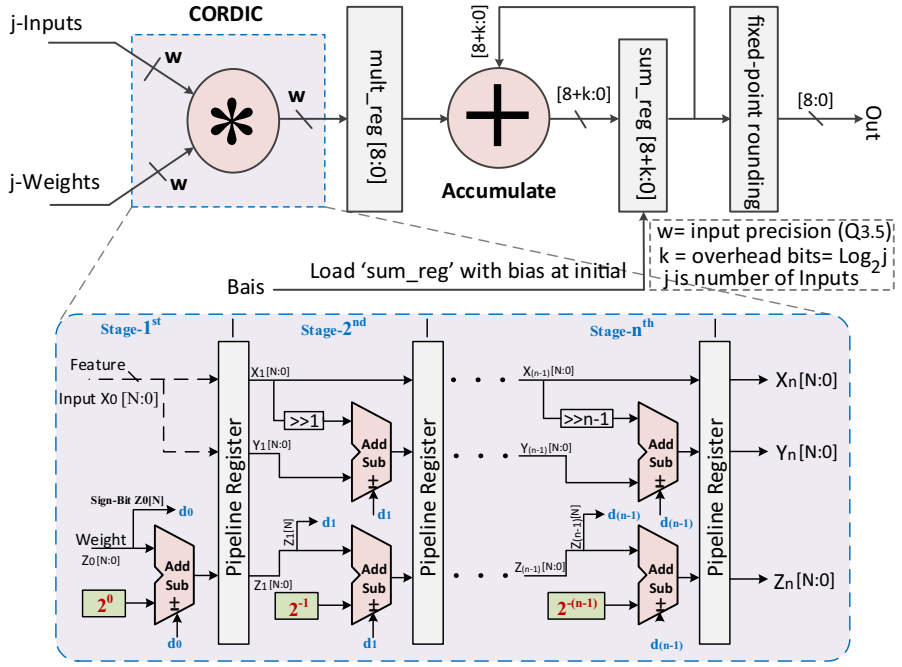


Fig. 4 Proposed pipeline CORDIC-based MAC architecture with single multiplier and adder, using signed N-bit, n-stage pipeline CORDIC architecture for enhanced performance. Bias constant loading required for fully connected DNN accelerator and `sum_reg` empty for first convolution computation

generating d_n in Eq. 3c. Additionally, in Fig. 3, the MSB bit of Z_n is utilized to calculate d_n, as per the CORDIC-based MAC computation design flow depicted in Fig. 2. Specifically, the value of d_n is +1 or -1, which is the sign of Z_n in the previous iteration. In Fig. 3b, the value of d_1 is +1, as the sign of Z_n is positive. Similarly, when Z_n is negative, it uses a value of 1 with a negative sign.

The proposed enhanced performance MAC architecture based on pipeline CORDIC is shown in Fig. 4. This pipeline architecture is relatively efficient in area and power consumption, as it does not require feedback registers and multiplexers, unlike iterative architecture [28]. Additionally, the proposed design uses an n-bit shift register instead of a barrel shifter, resulting in the desired output generated after the first N-clock cycles, providing n-times higher throughput computation compared to the iterative architecture.

The implementation of parallel multipliers and adder tree for MAC in neuron computation is costly and burdensome for ASICs and tiny FPGAs. Although a single multiplier followed by accumulation architecture is preferred due to its minimal resource utilization, it suffers from low throughput. The proposed MAC architecture of **w**-bit precision, as shown in Fig. 4, utilizes a single multiplier followed by accumulator. In DNN, the current layer output is an input to the subsequent next layer, and therefore, the same input and output format is preferred. We use $\mathbf{i}_{b_{in}} = \mathbf{i}_{b_{out}} = \mathbf{i}_b$, $\mathbf{f}_{b_{in}} = \mathbf{f}_{b_{out}} = \mathbf{f}_b$ and $\mathbf{w}_{in} = \mathbf{w}_{out} = \mathbf{w}$, where $\mathbf{i}_b$ represents the integer bits excluding the sign bit, $\mathbf{f}_b$

represents the fractional bits, and **w** represents the total number of bits at input/output. However, the final choice in the case of fixed-point format must be based on a target application that gives maximum accuracy with desired precision. The conventional w-bit precision MAC has $\lceil 2w + k \rceil$ output bits, where the w-bit multiplier generates the 2w-bit output bit-width. On the other hand, CORDIC-based design has the same input and output precision (w), as shown in Fig. 4. The CORDIC-based architecture has an (n-1) bit shift, which conventionally comes with 2n-bit precision. However, the proposed architecture does 1-bit shift operation in every iteration and has one bit lost at the same time, which inherently comes with n-bit precision. Moreover, due to the accumulation in the MAC, we have used extra overhead bits, i.e., k=$\lceil log_2 J(l) \rceil$, where $J(l)$ is the number of inputs to the MAC unit in the corresponding l^{th} layer of DNN.

The resources-efficient multiplication followed by successive accumulation in MAC has been successfully implemented, which computes the weighted sum in $j + n$ clocks, where j represents the number of inputs in the MAC computation and n represents the number of pipeline stages in the CORDIC computation. To resize the output bit size into **w**-bits (9-bits) with dynamic fixed-point arithmetic representation $Q_{3,5}$, a rounding scheme has been used at the MAC output, utilizing the *numeric_std* library package of VHDL. This allows for the same input and output precision for MAC implementation, as shown in Fig. 4. The proposed enhanced performance MAC is suitable for both convolution and fully connected layers. In fully connected layers, the bias must accumulate in the MAC computation, which is accomplished by loading it into the sum register for the first clock of MAC computation. The total clock overhead in the l^{th} layer, due to the performance *E*nhance CORDIC MAC architecture (T_E) in comparison with *C*onventional single clock multiplication evaluated architecture (T_C) [29], is expressed using Eq. 6, where n is the number of CORDIC pipeline stages.

$$T_E(l) = T_C(l) + (n - 1) \tag{6}$$

Therefore, the total clock overhead for complete DNN acceleration, due to the pipelined CORDIC-based MAC, is further dependent on the overall layers $l = 1, 2, ..., L$ available in the DNN accelerator. It is essential to notice that the enhanced performance CORDIC architecture has an initial L clock cycles overhead. After that, the circuit generates the output at every clock cycle. However, the critical delay of each stage is comparatively minimal, allowing it to operate at a higher frequency, enabling the MAC unit to be used for high-throughput applications. By applying the property of relative clocks evaluation, the simplified equation for comparable clock computation for the DNN with L-layers is expressed using Eq. 7.

$$\begin{aligned} T_E &= \sum_{l=1}^{L} T_E(l) \quad = \sum_{l=1}^{L} \Big(T_P(l) + n \Big) \\ &= \sum_{l=1}^{L} T_P(l) + L \cdot (n - 1) \end{aligned} \tag{7}$$

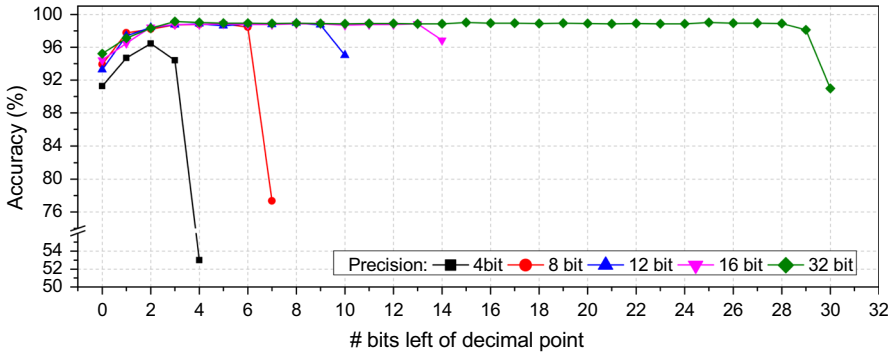


Fig. 5 Inference accuracy for MNIST dataset Tensor CNN model. Here, we observed the accuracy variation at different precision and different positioning of dynamic Fixed-point

4 Efficient MAC Design: Pareto-driven Pipeline Stages Selection

This section discusses two key aspects of the research article related to the design of efficient hardware for DNNs. The first part presents a Pareto study to determine the optimal integer bit-width for fixed-point arithmetic, which is crucial for achieving high accuracy in DNN applications. The second part focuses on identifying the minimum number of pipeline stages required for efficient multiplication operations in DNN hardware, while maintaining acceptable accuracy levels.

4.1 Pareto-study for Accuracy Variation with Dynamic Binary Point Implication

The various bit-precisions utilized include 8-bit, 12-bit, 16-bit, and 32-bit, for our experimental evaluation. For the representation of these precisions, we allocated one additional bit for the sign and three bits for the integer part. For instance, in the case of 8-bit precision, we adopted a 9-bit number format representation, where one bit was reserved for the sign, three bits for the integer part, and five bits for the fractional part. This representation is symbolically written as $Q_{3.5}$. We employed the $Q_{3.5}$ arithmetic to evaluate the proposed MAC performance. We designed the proposed MAC-based LeNET and Tensor CNN model on a software platform, and for accuracy evaluation and Pareto studies, we employed the MNIST dataset. By varying the position of the binary point implication for different arithmetic precisions, we observed the accuracy variation depicted in Fig. 5.

The use of dynamic fixed-point arithmetic with variable integer and fractional bits in DNN implementation can result in variable accuracy in object detection and classification applications [7]. Therefore, we performed a Pareto study to determine the optimal integer bit implication in the fixed-point arithmetic representation. In our analysis, we observed two significant conclusions from the Pareto study. Firstly, higher bit precisions yield nearly the same classification accuracy, which is particularly evident in larger datasets. However, this factor is less pronounced in simpler datasets such as MNIST, where even a 4-bit and 8-bit fixed-point precision model returns comparable

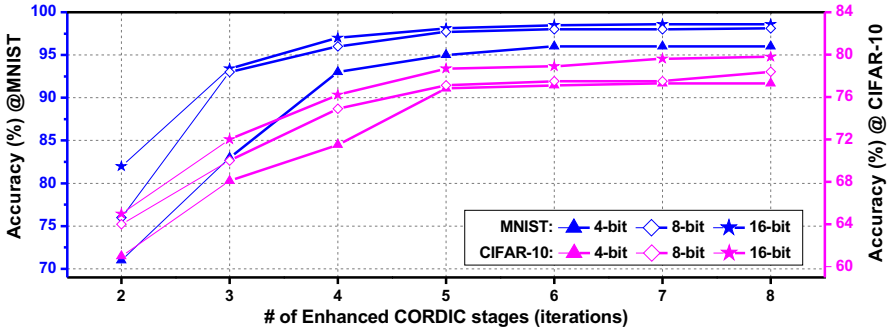


Fig. 6 Inference accuracy for MNIST dataset Tensor CNN model. Here, we observed the accuracy variation at different precision and different pipeline stages used within the MAC computation

accuracy. Secondly, increasing the integer bit width is not effective as the inference accuracy graph for integer bit widths indicates that 2-3 significant digits to the left of the decimal point are sufficient. More integer bits do not improve accuracy.

4.2 Identification of a Number of Pipelining Stages

The CORDIC-based MAC unit has designed for DNN applications and has enhanced throughput by implementing pipeline stages. Traditionally, N-bit precision multiplication operations require N+1 pipeline stages, but implementing the N+1 pipeline stages for 8-bit and higher precision computation is challenging and costly. The iterative CORDIC-based MAC has a hardware-efficient architecture, but it has a more critical delay that can be overcome by implementing a pipeline architecture. However, the pipeline architecture comes with larger resource utilization and more power consumption. As neural networks are error-resilient, systematic approximate computing can be employed to minimize pipeline stages and design efficient (area, power, delay) performance architecture, which inherently comes with computation approximation [10]. Therefore, we performed a Pareto study between the number of pipeline stages and accuracy and validated the results. The study helped us determine the minimum pipeline stage with optimum accuracy. Initially, we fixed the dynamic fixed-point implication for better performance, i.e., the number of integer bits and fractional bits found by the Pareto study, which is $Q_{3.5}$.

The research article demonstrates that the use of pipeline architecture in DNN computation with $Q_{3.5}$ arithmetic representation results in better accuracy. As the number of pipeline stages increases, the accuracy of computation improves until Z_{out} approaches zero, but at the cost of increased area overhead. Therefore, we performed Pareto analysis to determine the optimal number of CORDIC pipeline stages. Experimental results show that for all precision levels greater than or equal to 8-bit, MAC computation with five or more pipeline stages yields comparable accuracy, as depicted in Fig. 6. In this, the accuracy of the LeNet model on the CIFAR-10 dataset has been validated using a small sample set of data consisting of 1000 images. The results obtained are promising, with an accuracy of nearly 80%. To enhance the performance-centric

Table 2 A pipeline architecture for high-performance MAC computation using CORDIC in linear mode for fixed $Q_{3.5}$ representation, with iteration-level calculation shown at each n^{th} stage

n	$\mathbf{d_n}$	$\mathbf{X_{n+1} = X_n}$	$\mathbf{Y_{n+1} = Y_n + d_n X_n * 2^{-n}}$	$\mathbf{Z_{n+1} = Z_n - d_n * 2^{-n}}$
Pipelined	$(n+1)^{th}$	Initial Conditions/Inputs		
stage	iteration	Input = 0.65625_{10}	bias = 0.00_{10}	weight= 1.09375_{10}
initial	–	000.10101_2	000.00000_2	001.00011_2
0	+1	000.10101_2	000.10101_2	000.00011_2
1	+1	000.10101_2	000.11111_2	-000.01101_2
2	–1	000.10101_2	000.11010_2	-000.00101_2
3	–1	000.10101_2	000.11000_2	-000.00001_2
4	–1	000.10101_2	000.10111_2 /0.7187_{10}	000.00001_2 (≈ 0)

MAC unit, we implemented five pipeline CORDIC stages. Results are compared with state-of-the-art methods, and the use of a single-input CORDIC calculation with five pipeline stages is shown in Table 2. It can be observed that at the 4^{th} and 5^{th} stages, the results for Z_n closely approach zero.

5 Experimental Workflow and Evaluation

The performance-enhancing MAC for the DNN model is evaluated using Pareto analysis. The analysis is aimed at determining the minimum number of pipeline stages required for the CORDIC-based MAC unit. The model description is presented using HDL, and the $Virtex-7$ FPGA is utilized for hardware implementation and design validation. The design is further validated using Synopsys-$design\_vision$ [11], and post-synthesis results are presented to assess the scalability of the MAC architecture in ASIC. The following experiment is carried out to validate and compare the performance of the proposed design.

- An optimum pipeline stage for the proposed high-performance MAC using CORDIC has been evaluated using Pareto analysis. Since a limited number of pipeline stages come with approximation, error metrics have been evaluated for the optimum (five) stages-based MAC computation. Additionally, the accuracy has been analyzed on TensorFlow DNN models for MNIST and CIFAR datasets.
- A five-stage enhanced performance MAC was evaluated using Pareto analysis, and error metrics were analyzed for optimum computation. The performance of the MAC was compared to state of the art, and a fully parallel ANN model was developed and evaluated on Virtex-7 FPGA for MNIST and CIFAR datasets. The network had a five-layer configuration and was tested for ten output classes.
- The compatibility of the architecture for ASIC implementation has been evaluated, and the post-synthesis physical parameters have been assessed for the 45nm technology node.
- At lower technology modes, power is more sensitive to process variation and mismatch. To address this, the MAC RTL digital design was converted to a CMOS

design using the Siemens-$v2lvs$. The dynamic power variation was then projected using Monte Carlo simulations.

6 Performance Analysis and Discussion

The deep neural network's computationally dominant part is the MAC, which requires high computational power and more chip area. Moreover, the multiplier circuit has a significant critical delay, which increases in large scale for high-precision arithmetic. To address these issues, we propose an optimized MAC design architecture. The proposed MAC performance has been analyzed by extracting the results at different abstraction levels. An enhanced performance MAC unit has been introduced, which has better physical parameters at the cost of insignificant accuracy loss ($<1.5\%$). This section presents the results evaluation and comparison for both ASIC and FPGA-based implementations.

6.1 Performance Metric (Error Measures) and Accuracy Validation

The compatibility of the proposed architecture in a DNN model is verified by customizing with the LeNet model, using which inference accuracy is evaluated for MNIST and CIFAR-10 datasets. The different dynamic fixed-point representations with 4, 8, 16, and 32-bit precision are for the accuracy evaluation and comparison. The inference results show that there is an insignificant accuracy loss ($<1.5\%$) in moving for higher 32-bit to lower 8-bit precision computation in MNIST and CIFAR-10 dataset applications. In conventional combinational logic based designs, splitting higher precision multiplication operation (e.g., 16-bit), splitting two 16-bit operands into lower bit operands (8-bit) requires 4 (8 x 8) multiplications and adder circuitry for the addition. Hence, the area for 16-bit multiplier with conventional design is 4–5 times the area overhead than 8-bit multiplier. It means if 16-bit MAC architecture requires nearly $5\times$ hardware resources than 8-bit architecture. Further, bandwidth increases by $2\times$ which increases the power and reduces the throughput performance. Whereas, CORDIC-based MAC implementation has only 2 to $2.5\times$ resources overhead for 8-bit to 16-bit precision operand as reported [28].

The proposed architecture supports all fixed-point arithmetic precision with variable binary implications. Whereas, results focus on 8-bit precision design implementation and performance comparison. The inference accuracy of the proposed MAC and comparison with state-of-the-art is shown in Table 3. In combinational logic-based architecture implementation, we referred to exact computation in software and hardware platform-based results evaluation, which comes with superior accuracy at the cost of area power overhead. The proposed MAC uses approximation in arithmetic computation, saving huge hardware resources and improving other physical performance parameters at the cost of insignificant accuracy loss, which is nearly less than 1.5%. The CORDIC-based MAC architecture is efficient for higher precision since the scaling area overhead is less with increasing precision. It is observed that by limiting the number of pipeline stages of CORDIC, one can save huge hardware resources

Table 3 Inference accuracy metrics on multi-bit precision network with proposed MAC-based LeNet for the different datasets

Bit-precision Dynamic	Inference accuracy@LeNet (%) MNIST		CIFAR-10	
Fixed-point	Tensor [28]	PipeMAC	Tensor [28]	PipeMAC
32-bit	99.3	98.7	81.7	81.2
16-bit	98.9	98.5	81.2	79.8
12-bit	98.9	98.1	80.8	79.1
8-bit	98.8	97.9	80.7	78.4
4-bit	97.4	96.2	77.9	77.4

by compromising with a maximum of nearly 1.2% accuracy loss compared to exact computing in the case of MNIST and CIFAR-10. The number can be variable for different datasets, and therefore, before hardware implementation, the number of pipeline stages can be fixed using Pareto analysis.

The demand for efficient hardware design in terms of physical performance is high in real-time applications. Deep Neural Networks (DNNs) have an error-resilient feature that allows for approximation in computation. To address performance enhancement architecture, approximation computation has been attempted. Pareto analysis helped to determine the minimum number of pipeline stages required in MAC computation. The limited number of pipeline stages allows for significant savings in area resources and power consumption. As discussed in detail in Sect. 6.2, the proposed architecture has lower area and power consumption compared to the state of the art. Furthermore, the design is highly efficient for precision scaled architecture [28]. The design has been optimized for area and power by evaluating the required number of pipeline CORDIC stages in the MAC implementation. It has been observed that the five-stage pipeline architecture shows comparative results and reduces on-chip area utilization and power consumption. However, the reduced number of pipeline stages results in errors in output computations. To validate the output calculation, different error metric equations have been used, including mean square error, mean absolute error, average error, and standard deviation. The analysis in Table 4 is for the 8-bit precision CORDIC computation with five pipeline stages. The errors are considered for fixed-point $Q_{3.5}$ representation and were observed to have a negligible error count in the overall output results with five pipelines CORDIC stage-based MAC computation.

6.2 Hardware Resources Utilization and Comparison

The MAC unit and MAC-based DNN model have been implemented using Vivado-Xilinx for the purpose of evaluating the results. The Virtex-7 FPGA VC707 Evaluation Kit has been used to evaluate the physical parameters of resource utilization, power consumption, and delay analysis. Results extraction has been conducted using signed fixed-point $Q_{3.5}$ arithmetic notation and compared to state-of-the-art techniques. The proposed MAC design has been implemented using HDL for enhanced performance.

Table 4 Proposed MAC computations' error metrics, % error estimation and comparison with accurate model [29]

Performance metrics (error measure)				
Error metrics	Mean square error (MSE)	Mean absolute error (MAE)	Average error (AE)	Standard deviation (SD)
Metric equation	$\frac{1}{N}\sum_{i=1}^{N}(A-E)^2$	$\frac{1}{N}\sum_{i=1}^{N}\lvert(A-E)\rvert$	$\sum_{i=1}^{N}\frac{\frac{\lvert(A-E)\rvert}{\lvert E\rvert}}{N}$	$\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(A-A')^2}$
Measured Value	0.055%	1.82%	5.31%	0.434187

N=samples; *A=approx value*; *E=accurate value*; *A '=mean approx value;*

Post-implementation results have been extracted, demonstrated, and presented in Table 5.

The performance parameters of MAC architectures with various design techniques, including Vedic multiplier, Wallace tree, Booth algorithm, shift-add, and CORDIC-based implementation, were extracted and compared to state-of-the-art techniques. The results summary is presented in Table 5. We reported the MAC level results in Table 5 to validate the proposed MAC performance benefits. The proposed design utilizes less hardware compared to state-of-the-art techniques, and in addition, it has a minimum critical delay, providing high throughput and a lower power-delay product. The proposed MAC design employs CORDIC computation, which involves two fundamental operations: n-bit right-shift and n-bit addition, and use of pre-calculated memory constants in the calculation. The proposed design employs the right-shift function, enabling both signed and unsigned computation, unlike the left shift-and-add operation [9]. Furthermore, unlike conventional shift-and-add calculation [9], which requires n-clocks to generate the final desired output for n-bit precision computation, the proposed design needs only a single clock cycle.

The implementation report of the proposed architecture has been extracted and presented in Fig. 4. As previously noted, CORDIC-based computation traditionally required nine pipeline stages and resulted in area overhead with a single stage for n-bit precision. Therefore, we optimized the hardware architecture by reducing the required computing resources by limiting pipeline stage, as discussed in Section 3. We used Pareto analysis to determine the necessary pipelining, which revealed that five stages are sufficient for MAC evaluation. The initial output of the proposed pipelined design takes five clocks, and each additional clock generates the output specified in Sect. 3. We have demonstrated the results for the compute unit with 8-bit precision (i.e., $w = 8$). The iterative CORDIC-based architecture proposed in [28] has a smaller size and power budget than other state-of-the-art architectures, but each final calculation requires five clocks. The results have been obtained for signed fixed-point 8-bit precision representation. The MAC with five pipelined stages consumed 58 LUTs, while the recursive CORDIC architecture used 23 LUTs. The reason for the 2.3× resource scaling is the absence of feedback registers and multiplexers in the proposed design, unlike the recursive CORDIC architecture. Additionally, the proposed design does not require a barrel shifter, unlike conventional iterative CORDIC architectures. Therefore, the resource utilization did not increase proportionally with the number of pipeline stages. The MAC with pipeline CORDIC used 2.3× resource utilization compared to [28]; however, it improves the throughput performance by five times, and the Power-Delay product is lower compared to the iterative architecture. The critical delay of the recursive CORDIC architecture is reported for five iterations in Table 5. Moreover, the power-delay product was calculated to evaluate the overall performance which is 46% less compared to the iterative CORDIC-based architecture design in [28] and 67% less compared to the best state-of-the-art technique.

The proposed MAC design achieves higher performance throughput per watt, but with a trade-off of 0.35% accuracy loss compared to Xilinx IP accurate MAC computation, as shown in Table 6. The hardware resources utilization report is presented in Table 5, where it is observed that Xilinx IP [29] consumes 2.24× LUTs compared to the proposed architecture. Additionally, the iterative CORDIC [28] architecture

Table 5 Resource utilization and performance parameters at 'fixed-point $Q_{3.5}$' for MAC

Resources utilization	LUT (17600)	FF (35200)	Critical path delay (ns)	Power-delay product(pJ)
Vedic [35]	159	245	4.48	6.11
IEEE [29]	130	49	3.98	5.63
Wallace [15]	105	112	2.59	3.29
Booth [5]	83	61	3.08	3.07
Shift-add [9]	75	58	5.44	4.17
CORDIC [28]	23	22	9.06	1.90
Proposed	58	74	1.86	1.01

Table 6 Performance parameters for fully connected NN `196:64:32:32:10` at 'fixed-point $Q_{3.5}$' with different MAC unit implementation. Results are produced using Vertex-7 FPGA

Parameters	Xilinx IP MAC [29]	Iterative CORDIC [28]	Proposed pipeline CORDIC
On-chip power (W)	2.194	0.67	1.13
Computational time (in # clock cycles)	344	1640	360
Throughput (GOPS)	4.650	0.977	4.450
Performance (GOPs/W)	2.11	1.45	3.94
Accuracy (%)	95.41	95.06	95.06

consumes fewer resources than the proposed design. Both designs are evaluated for overall architecture-level performance in Table 6. The conventional designs require 344 clock cycles, whereas the pipeline CORDIC-based architecture needs four extra clocks in each layer, resulting in 360 cycles for final evaluation. However, the proposed MAC-based DNNs can be operated at a higher clock rate, with nearly half the critical path delay. The proposed DNN design architecture computes 15,963 Multiplication operations, 15,963 addition operations, and 138 AF operations at 50MHz in these 344 cycles, giving about 4.65 GOPs. Similarly, for proposed pipeline-based architecture requires 360 clock cycles. Here, Table 6 reports the throughput for the first image inference. However, after the first image inference, the Xilinx IP-based architecture will take 201 clock cycles, whereas in the case of the pipeline architecture, it will take 205 clock cycles. Based on these values, the proposed design achieves 1.89× better performance throughput per watt than conventional MAC-based fully parallel neural network.

6.3 Physical Performance Parameters Evaluation of CORDIC-based MAC and Comparison with State of the Art

The proposed architecture has been validated for the ASIC design development process, and physical parameters have been evaluated and compared with state-of-the-art methods. The architecture's performance parameters, including area, on-chip power, and critical logic delay at the 45*nm* technology node, have been examined. The post-synthesis parameters for the proposed work and state-of-the-art designs are reported in Table 7, using Synopsys-*design_vision*. In this table, N represents the number of input features discussed in Sect. 3. To reduce area overhead compared to the iterative architecture [28], an n-bit shifter was used instead of a barrel-shifter in this design. Although the shift-and-add method [9] has fewer physical parameters, it requires N-clock cycles for N-bit *(8-clocks for 8-bits)* precision computation, which reduces the throughput performance and renders it unsuitable for edge computing applications.

The proposed design employs a pipeline architecture, while the state-of-the-art design uses iterative computation that takes N clock cycles for each computation, resulting in high latency [28] (nearly 5 times that of the five pipeline stages), and a relative ADP of 1.13 times. The iterative architecture is better suited for situations where area and power are highly constrained, and low throughput is acceptable. However, the proposed design uses a pipeline architecture, which inherently has more area, but the overall performance is relatively good (such as ADP) due to hardware architecture optimization. The relative area overhead in this design is smaller than that in the iterative architecture because we used an n-bit shifter instead of a barrel shifter and removed feedback registers. Despite having better physical parameter reports, the shift-and-add approach [9] requires N-clock cycles for each N-bit *(8 clocks for 8 bits)* of precision computing, which reduces throughput performance and makes the method less efficient and incompetent in DNN accelerators.

The high-precision computation in DNN returns better accuracy for more extensive and complex datasets. The CORDIC-based MAC in DNN implementation is an admirable choice for 8-bit and higher precision computation [28]. Whereas, in the worst-case scenario with 8-bit precision, physical parameters are evaluated as shown in Table 7. In this table, N represents the number of accumulations performed by the MAC, which further depends on the number of input features at the input of the MAC unit. To evaluate the overall chip cost, we have used the figure-of-merit, the area, latency, and power product *(ALP=area* $\times$ latency $\times$ power). The ALP is 12% and 21% less compared to the iterative CORDIC-based architecture proposed in [28] and other best of state-of-the-art works [9], respectively. It is essential to notice that the iterative CORDIC-based MAC requires 5-clock cycles for each multiplication, and therefore, in each neuron, we need $5n$ clock cycles for n multiplications and accumulation. Consequently, such designs can be helpful where area and power are on a tight budget and throughput can be compromised. The proposed architecture uses 5 clocks for the first multiplication due to its pipelined nature, and afterward, it computes each multiplication at every clock cycle as depicted in Eq. 7. It is helpful for many modern AI applications, which would be better in terms of physical performance along with

Table 7 Resource utilization and performance parameters for multiply-accumulate unit with 'fixed ⟨8, 5⟩' arithmetic

Performance parameters	Area (μm^2)	Dy. Power (μW)	St. Power (μW)	Delay (ns)	Clock cycles	Area-delay-power(ADP)
Vedic [35]	1164	484.8	4.93	7.66	N	6.10×
IEEE [29]	832	495.5	3.81	6.71	N	3.89×
Wallace [15]	838	498.0	3.88	6.44	N	3.79×
Booth [5]	771	146.7	6.56	7.42	N	1.23×
Shift-add [9]	501	119.5	2.86	4.27	$8N$	2.92×
CORDIC [28]	307	139.2	4.72	3.62	$5N$	1.13×
Proposed	749	322.7	10.5	2.83	$N+5$	1.00×

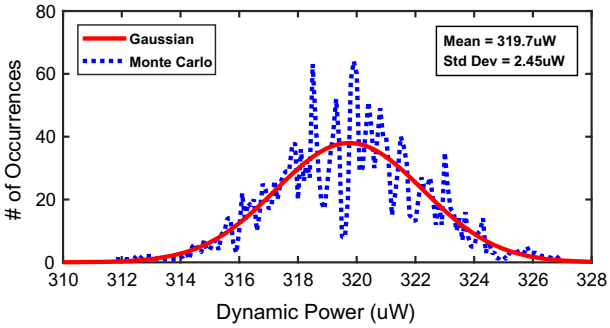


Fig. 7 Dynamic Power consumption by CORDIC based MAC unit having five pipeline stages. The Monte Carlo with process variation and mismatch simulation is perform for 2500 samples

enhanced throughput, applications like speech recognition, image recognition, video analysis, medical data analysis, robotics, etc.

6.4 Process Variation and Mismatch Analysis

In the context of integrated circuit manufacturing, process variation refers to the properties of the device and peripheral components such as length, breadth, and oxide thickness. At process nodes smaller than 65nm, process variation becomes more apparent as it constitutes a larger proportion of the device's overall length or breadth, and because feature sizes are closer to the basic dimensions used for lithography masks. As a result, process variation and device mismatch become crucial in determining the stability and reliability of physical circuit characteristics at lower technology nodes. Static current varies significantly due to process variation and mismatch, and power-delay product at lower technology nodes is more sensitive to naturally occurring variation. To observe this variation, Monte Carlo simulation calculates the probabilistic distribution of dynamic power variation due to process variation and device mismatch in the characteristics of similar design devices, occurring during the manufacturing of ICs. Hence, we carried out Monte Carlo simulation for 10,000 samples to validate the power variation due to process and mismatch. The RTL design was extracted into custom-CMOS circuit design using *v2lvs-Mentor Graphics*. We performed the Monte Carlo simulation in *Virtuoso-Cadence* for 2500 samples as shown in Fig. 7. The simulation result was obtained at the 45nm technology node. The proposed design exhibits less dynamic power variation and standard deviation. The mean dynamic power and σ deviation at the 45nm node are $319.7uW$ and $2.45uW$.

7 Conclusion

This study has demonstrated the effectiveness of a pipeline CORDIC-based approach with minimal pipeline stages and less critical delay, leading to improved MAC performance in DNN applications. Our results have shown that limiting the pipeline stages

can lead to erroneous computation. However, by adopting the proposed approach, we observed better overall performance. Software implementation of DNN with five pipeline stages in the MAC showed less than 1% accuracy loss in DNN inference. Furthermore, we have demonstrated that the proposed MAC architecture allows for both signed and unsigned computations, and our comparison with alternative design approaches demonstrated better relative performance. The proposed MAC unit can adapt to various inferencing applications with high throughput computations by selecting appropriate circuit design parameters such as bit-precision, # of integer bits, and # of pipeline stages. The implemented CORDIC MAC-based fully connected neural network operated at 66MHz with better performance parameters. The standalone MAC unit could run at 460MHz, indicating that the pipeline MAC architecture has mitigated the low throughput issue present in the iterative architecture. Our proposed technique has outperformed earlier designs for implementing and evaluating FPGA and ASIC. The proposed system architectural methodology for performance optimization opens up new possibilities for the reduced area and power with better performance design, particularly in Edge-AI DNN accelerators.

Acknowledgements The authors would like to thank the University Grant Commission (UGC) New Delhi, Government of India under the SRF scheme with award no. 22745/(NET-DEC. 2015) for providing financial support. And Special Manpower Development Program Chip to System Design (SMDP), Department of Electronics and Information Technology (DeitY) under the Ministry of Communication and Information Technology, Government of India for providing necessary Research Facilities.

Data Availability Data sharing is not applicable to this article as no data sets were generated or analyzed during the current study, and detailed circuit simulation results are given in the manuscript.

Declarations

Conflict of interest There is no conflict of interest from the authors

References

1. R. Andraka, A survey of CORDIC algorithms for FPGA based computers. In: Proceedings of the 1998 ACM/SIGDA sixth international symposium on field programmable gate arrays, pp. 191-200 (1998)
2. A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, VLSI implementation of deep neural network using integral stochastic computing. IEEE Trans Very Large Scale Integr (VLSI) Syst 25(10): 2688-2699 (2017)
3. Z. Carmichael, H.F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson, and D. Kudithipudi, Performancefficiency trade-off of low-precision numerical formats in deep neural networks. In: Proceedings of the conference for next generation arithmetic, pp. 1-9 (2019)
4. H. Chhajed, G. Raut, N. Dhakad, S. Vishwakarma, and S. K. Vishvakarma, BitMAC: bit-serial computation-based efficient multiply-accumulate unit for DNN accelerator. Circuits Syst Signal Process, pp. 1-16(2022)
5. F.U.D. Farrukh, C. Zhang, Y. Jiang, Z. Zhang, Z. Wang, Z. Wang, and H. Jiang, Power efficient tiny yolo CNN using reduced hardware resources based on booth multiplier and wallace tree adders. IEEE Open J. Circuits Syst. 1 (2020): 76-87
6. T. Fujii, S. Sato, H. Nakahara, and M. Motomura, An FPGA realization of a deep convolutional neural network using a threshold neuron pruning. In: Applied reconfigurable computing: 13th international symposium, ARC 2017, Delft, The Netherlands, April 3-7, 2017, Proceedings 13, pp. 268-280. Springer International Publishing (2017)

7. M. Gao, Q. Wang, and G. Qu, Energy and error reduction using variable bit-width optimization on dynamic fixed point format. In: 2019 IEEE computer society annual symposium on VLSI (ISVLSI), pp. 152-157. IEEE (2019)
8. J. Garland, D. Gregg, Low complexity multiply-accumulate units for convolutional neural networks with weight-sharing. ACM Trans. Architect. Code Optim. (TACO) **15**(3), 1–24 (2018)
9. D.A. Gudovskiy, and L. Rigazio, Shiftcnn: generalized low-precision architecture for inference of convolutional neural networks. arXiv preprint arXiv:1706.02393 (2017)
10. M.A. Hanif, R. Hafiz, and M. Shafique, Error resilience analysis for systematically employing approximate computing in convolutional neural networks. In: 2018 design, automation and test in Europe conference and exhibition (DATE), pp. 913-916. IEEE (2018)
11. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/design-compiler-graphical.html
12. ISO/IEC/IEEE International Standard - Floating-point arithmetic, ISO/IEC 60559:2020(E) IEEE Std 754-2019, pp.1-86 (2020)
13. S. Jain, S. Venkataramani, V. Srinivasan, J. Choi, P. Chuang, and L. Chang, Compensated-DNN: energy efficient low-precision deep neural networks by compensating quantization errors. In: Proceedings of the 55th annual design automation conference, pp. 1-6 (2018)
14. H. Jiang, C. Liu, F. Lombardi, J. Han, Low-power approximate unsigned multipliers with configurable error recovery. IEEE Trans. Circuits Syst. I Regul. Pap. **66**(1), 189–202 (2018)
15. R.B. S. Kesava, B. L. Rao, K. B. Sindhuri, and N. U. Kumar, Low power and area efficient Wallace tree multiplier using carry select adder with binary to excess-1 converter. In: 2016 conference on advances in signal processing (CASP), pp. 248-253. IEEE (2016)
16. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436–444 (2015)
17. W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications. Neurocomputing **234**, 11–26 (2017)
18. M. Masadeh, O. Hasan, S. Tahar, Input-conscious approximate multiply-accumulate (mac) unit for energy-efficiency. IEEE Access **7**, 147129–147142 (2019)
19. A.N. Mazumder, J. Meng, H.A. Rashid, U. Kallakuri, X. Zhang, J.S. Seo, T. Mohsenin, A survey on the optimization of neural network accelerators for micro-ai on-device inference. IEEE J. Emerging Sel. Topics Circuits Syst. **11**(4), 532–547 (2021)
20. L. Mei, M. Dandekar, D. Rodopoulos, J. Constantin, P. Debacker, R. Lauwereins, and M. Verhelst, Sub-word parallel precision-scalable MAC engines for efficient embedded DNN inference. In: 2019 IEEE international conference on artificial intelligence circuits and systems (AICAS), pp. 6-10. IEEE (2019)
21. E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, FPGAs in industrial control applications. IEEE Trans. Ind. Inf. 7(2): 224-243 (2011)
22. V. Mrazek, L. Sekanina, and Z. Vasicek, Libraries of approximate circuits: automated design and application in CNN accelerators. IEEE J. Emerging Sel. Topics Circuits Syst. 10(4): 406-418 (2020)
23. H. Nakahara, and T. Sasao, A high-speed low-power deep neural network on an FPGA based on the nested RNS: applied to an object detector. In: 2018 IEEE international symposium on circuits and systems (ISCAS), pp. 1-5. IEEE (2018)
24. M. Nazemi, G. Pasandi, and M. Pedram, Energy-efficient, low-latency realization of neural networks through Boolean logic minimization. In: Proceedings of the 24th Asia and South Pacific design automation conference, pp. 274-279 (2019)
25. V. Rajagopal, C. K. Ramasamy, A. Vishnoi, R. N. Gadde, N. R. Miniskar, and S. K. Pasupuleti, Accurate and efficient fixed point inference for deep neural networks. In: 2018 25th IEEE international conference on image processing (ICIP), pp. 1847-1851. IEEE (2018)
26. G. Raut, S. Rai, S. K. Vishvakarma, and A. Kumar, A CORDIC based configurable activation function for ANN applications. In: 2020 IEEE computer society annual symposium on VLSI (ISVLSI), pp. 78-83. IEEE (2020)
27. G. Raut, A. Biasizzo, N. Dhakad, N. Gupta, G. Papa, and S. K. Vishvakarma, Data multiplexed and hardware reused architecture for deep neural network accelerator. Neurocomputing 486: 147-159 (2022)
28. G. Raut, S. Rai, S.K. Vishvakarma, A. Kumar, RECON: resource-efficient CORDIC-based neuron architecture. IEEE Open J. Circuits Syst. **2**, 170–181 (2021)

29. T. Sato, and T. Ukezono, A dynamically configurable approximate array multiplier with exact mode. In: 2020 5th international conference on computer and communication systems (ICCCS), pp. 917-921. IEEE (2020)
30. H. Sim, J. Lee, Cost-effective stochastic MAC circuits for deep neural networks. Neural Netw. **117**, 152–162 (2019)
31. V. Sze, Y.-H. Chen, T.-J. Yang, J.S. Emer, Efficient processing of deep neural networks: a tutorial and survey. Proc. IEEE **105**(12), 2295–2329 (2017)
32. R. Thomas, V. DeBrunner, and L. DeBrunner, Fixed-point implementation of discrete Hirschman transform. In: 2018 52nd Asilomar conference on signals, systems, and computers, pp. 1507-1511. IEEE (2018)
33. A. Thomas, 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), pp. 1-8. IEEE (2021)
34. Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, Finn: a framework for fast, scalable binarized neural network inference. In: Proceedings of the 2017 ACM/SIGDA international symposium on field-programmable gate arrays, pp. 65-74 (2017)
35. A. Vamsi, S. Krishna, and S. R. Ramesh, An efficient design of 16 bit mac unit using vedic mathematics. In: 2019 international conference on communication and signal processing (ICCSP), pp. 0319-0322. IEEE (2019)
36. N. Van Toan, J.-G. Lee, FPGA-based multi-level approximate multipliers for high-performance error-resilient applications. IEEE Access **8**, 25481–25497 (2020)
37. S. Yin, Z. Jiang, J.-S. Seo, M. Seok, XNOR-SRAM: in-memory computing SRAM macro for binary/ternary deep neural networks. IEEE J. Solid-State Circuits **55**(6), 1733–1743 (2020)
38. Yugandhar, K., V. Ganesh Raja, M. Tejkumar, and D. Siva. "High performance array multiplier using reversible logic structure." In 2018 international conference on current trends towards converging technologies (ICCTCT), pp. 1-5. IEEE, 2018
39. S. M. A. Zeinolabedin, F. M. Schüffny, R. George, F. Kelber, H. Bauer, S. Scholze, S. Hänzsche et al. A 16-channel fully configurable neural SoC with 1.52μ W/Ch signal acquisition, 2.79μ W/Ch real-time spike classifier, and 1.79 TOPS/W deep neural network accelerator in 22 nm FDSOI. IEEE Trans. Biomed. Circuits Syst. 16(1): 94-107 (2022)
40. H. Zhang, D. Chen, S.-B. Ko, New flexible multiple-precision multiply-accumulate unit for deep neural network training and inference. IEEE Trans. Comput. **69**(1), 26–38 (2019)