International Journal of Early Childhood Special Education (INT-JECSE)
DOI:10.48047/intjecse/v16i3.30 ISSN: 1308-5581 Vol 16, Issue 03 2024

Optimizing DNN Accelerators for Edge-AI: A CORDIC-Based Approach

Shaik. Shafi,

MTech Student (VLSI & ES), Department of ECE, QIS College of Engineering and Technology, Ongole, Andhra Pradesh, India.

Dr. CH. Hima Bindu

MTech (Ph.D.), Professor &HOD, Department of ECE, QIS College of Engineering and Technology, Ongole, Andhra Pradesh, India.

Abstract:

Deep Learning, a part of Artificial Intelligence, uses deep neural networks. These networks need a lot of resources and efficient design to work well. They also require high memory and parallel processing. Building these networks is tough because they need large components like MAC units and activation functions. Edge-AI applications need high-speed accelerators, which use more space and power. To make a good DNN accelerator without losing much speed, we need to improve MAC design, activation functions, and network complexity. ASIC-based hardware designs for DNNs have limited space and flexibility. This study looks at creating low-power and efficient DNN accelerators using the CORDIC architecture for MAC and activation functions. CORDIC designs are efficient but slow. To fix this, we suggest a pipelined architecture for CORDIC-based MAC and activation functions. Pipelining uses more resources, so we study the balance between stages and accuracy to get high speed. We offer different designs for CORDIC-based MAC and activation functions with iterative and pipelined approaches. he proposed design's system parameters and hardware can be adjusted for specific applications. Iterative architecture is good for AI-enabled IoT applications with minimal speed loss. Pipeline architecture is better for Edge-AI, offering high speed but using more space and power. We developed a digital solution to design and implement a deep neural network for ASIC and FPGA platforms. DNNs work well with both digital and analog inputs. An ADC is needed to convert analog input to digital for processing. This study also focuses on designing a low-power, high-speed 4-bit Flash ADC to handle both analog and digital inputs. The ADC works effectively at a sampling rate of 2.4 GS/s and is used in analog-digital interface accelerators. Overall, the proposed designs use fewer hardware resources and less power, which is important for edge computing solutions.

Keywords: Multiply-and-Accumulate, activation function, Co-ordinate Rotation Digital Computer,

I. Introduction

In recent years, artificial intelligence (AI) has become extremely popular. AI and machine learning (ML) have transformed modern society. Over the past decade, there have been significant advancements in this field [1]. Innovations like self-driving cars, digital assistants, manufacturing robots, and smart cities show that creating intelligent machines is possible. AI has already changed many industries, such as retail, manufacturing, banking, healthcare, and journalism, and it continues to grow into new areas.

Machine learning, a part of AI, allows computers to learn new skills and knowledge without being explicitly programmed. It is essential for creating autonomous programs that can gather data and learn [2]. Like the human brain, machine learning uses inputs like knowledge graphs or training data to understand domains, entities, and their relationships. Once these entities are identified, deep learning can begin.

Deep learning is a type of machine learning inspired by how the human brain works. It uses large datasets to improve its algorithms. Like humans learning from experience, deep learning algorithms repeat tasks and make small adjustments each time to get better results.

Deep learning involves neural networks with many layers, called deep neural networks (DNNs). These networks are the backbone of many modern AI applications. According to computer scientist John McCarthy [3], the goal of DNNs is to create smart machines that can achieve human-like goals. DNNs perform well because they can extract important features from raw data after being trained on large amounts of information.

DNNs are used in various tasks like lane detection, pattern recognition, fault detection, and industry monitoring [4]. However, these tasks often need real-time processing, which requires a lot of computing power. To speed up DNN computations, platforms like GPUs, CPUs, ASICs, and FPGAs are used. CPUs and GPUs support processes like Multiply-Accumulate (MAC), but they consume a lot of power because they don't use their resources efficiently. In contrast, ASICs and FPGAs perform fast multiplication with less resource use and lower power consumption.

Different activation functions are useful in different situations. However, traditional architectures can't easily adjust the use of multiple activation functions on a chip. This chapter looks at how to optimize and design activation functions (AF) for deep neural networks (DNNs) [5]. One big challenge is making these functions flexible while keeping the chip size small, especially when using ASICs (Application-Specific Integrated Circuits).

To make the most of an ASIC's parallel processing power, we need a space-efficient, customizable architecture for AFs. Our proposed solution is the Coordinate Rotation Digital

Computer (CORDIC) method, which uses a shift-and-add technique. This adaptable AF design includes both tan hyperbolic and sigmoid functions.

Deep neural networks (DNNs) [6] are commonly used for tasks like pattern recognition, prediction, and classification. Key parts of a DNN include the computational unit, activation function (AF), mathematical precision, data formats, and design platform. There are three main types of hardware for DNNs: application-specific integrated circuits (ASICs), digital signal processors (DSPs), and field-programmable gate arrays (FPGAs). ASICs offer the best performance and space efficiency, but they can't be easily changed for different uses and require a lot of resources for large networks.

To achieve high accuracy with multiple activation functions, power-efficient designs are needed to lower the supply voltage. Hardware implementations of DNNs can use the network's parallelism to improve performance. Efficient VLSI designs have been developed for various DNN applications [7]. However, the accuracy and performance of a neural network depend heavily on data precision, which can increase power and space requirements when implemented in hardware. Flexible architectures are needed to balance this trade-off. FPGAs take up more space than ASICs but offer more customizable designs. Neural networks need adjustable activation functions like sigmoid, hyperbolic tangent (tanh), and exponential non-linear transformations.

Figure 1 shows the typical design of a neuron with several activation functions, as suggested. Among its many significant downsides are the following: higher data propagation time (because of MUX), power dissipation (static power dissipation) because of the unused hardware, and area overhead (path-A, path-B, and path-C) to separate activation functions. Since only one activation function can be enabled at a time, the separate hardware for each activation function is not the ideal option. To tackle this trade-off, we offer an optimized CORDIC-based architecture that allows neural networks to perform efficient yet adjustable computations.

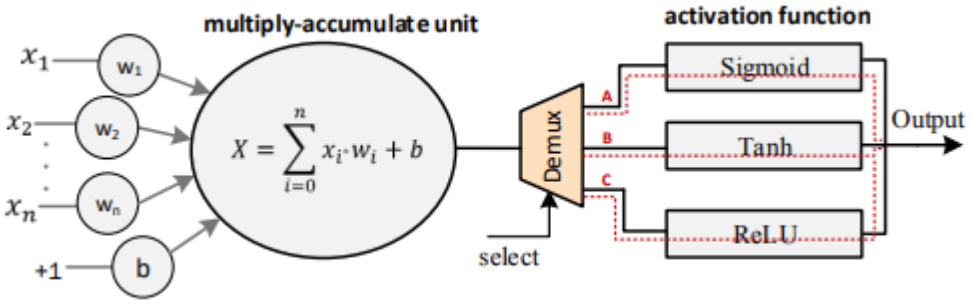


Figure 1: Typical design architecture of single neuron with configurable activation function

II. Literature Survey

Classical philosophy saw human thinking as the mechanical handling of symbols. For a long time, people have tried to create artificial beings with human-like intelligence, which was the start of artificial intelligence (AI). In 1950 [8-9], Alan Turing discussed the idea of making an intelligent machine and introduced the 'imitation game,' now known as the 'Turing test.' The Dartmouth summer research project on AI in 1956 is considered the official start of AI as a research field [10-11]. Over the years, AI has had its ups and downs. Recently, with the availability of big data and faster computers, AI has gained a lot of attention and investment. Machine learning (ML) methods have been successfully used to solve many problems in both academia and industry.

Machine learning (ML) algorithms, including those inspired by biology, originally aimed to mimic how the human brain works. The human brain is seen as the most intelligent 'machine' with very complex and efficient operations. In ML algorithms, two main units are synapses and neurons, similar to the biological nervous system. Synapses handle information processing, while neurons handle feature extraction. There are various neuron models like McCulloch–Pitts, sigmoid, ReLU, and Integrate-and-Fire, all of which have nonlinear characteristics needed for feature extraction and training neural networks (NNs) [12]. Later, 'biologically inspired' models were created as mathematical methods to achieve advanced functions. Modern ML algorithms are generally divided into two types: artificial neural networks (ANNs), which use numerical values, and spiking neural networks (SNNs), which use spikes to represent data.

The rapid increase in data for AI applications has made the need for specialized computing platforms more urgent. These platforms, designed specifically for AI, range from complements to traditional von Neumann platforms to essential, stand-alone solutions. Known as 'domain-specific computing,' these platforms are customized for AI tasks. They have achieved significant improvements in power and performance efficiency by overcoming challenges like the 'memory wall' and 'power wall.' Recent AI-specific computing systems, or AI accelerators, are built with many highly parallel computing and storage units [13]. These units are arranged in a two-dimensional (2D) layout to support common matrix-vector multiplications in neural networks.

In addition to traditional CMOS designs, new types of non-volatile memories like ReRAM are being used in AI accelerators. These new memories can store a lot of data and access it quickly, and they can also perform calculations directly within the memory. ReRAM arrays can store neural networks and perform matrix-vector multiplications in an analog way. Compared to the latest CMOS designs, ReRAM-based AI accelerators are much more efficient, using 3-4 times less power [14]. Although analog operations can be noisy, machine learning algorithms can handle this noise well. However, converting between analog and digital signals in these accelerators requires DACs and ADCs, which use a lot of power and space.

III. Proposed System

Using deep neural networks (DNN) for real-world problems requires a lot of hardware, high computational power, and more memory bandwidth. AI-enabled IoT applications need DNN engines that use resources efficiently. The hardware needed for DNNs increases with the depth of the neural network. In this chapter, we propose a design for a DNN engine that improves performance and uses resources efficiently. The design uses a layer-multiplexed DNN accelerator at the system level. Although reusing a single layer in the DNN can limit throughput, this issue is addressed by using a pipelined MAC design.

Types of Activation Functions and Design Techniques

Activation functions (AF) in deep neural networks (DNN) transform the output of the MAC (multiply-accumulate) operation. Because AFs are non-linear, they need more hardware resources, and resource usage increases a lot with higher precision. Implementing piecewise linear (PWL) AFs also requires extra memory, especially for 16-bit or higher precision. As precision increases, the memory needed grows a lot, making hardware usage very expensive. DNN applications use many types of non-linear transformations.

There are many types of activation functions, such as linear, sigmoid, tanh, rectified linear units (ReLU), parameterized ReLU, exponential, swish, and softmax. For back-propagation to work, activation functions need to be differentiable to calculate gradients. The best activation functions are differentiable, non-linear, and easy to use. You can choose between linear and non-linear activation functions, but non-linear ones are usually better. Common non-linear activation functions include Sigmoid, Tanh, ReLU, and Exponential. Depending on the application, other non-linear functions like Leaky ReLU, SeLU, and Softplus are also used. This chapter discusses several non-linear activation functions, including sigmoid, tanh, and ReLU. We have designed a flexible architecture using the CORDIC algorithm that can activate sigmoid, tanh, and ReLU with the same hardware.

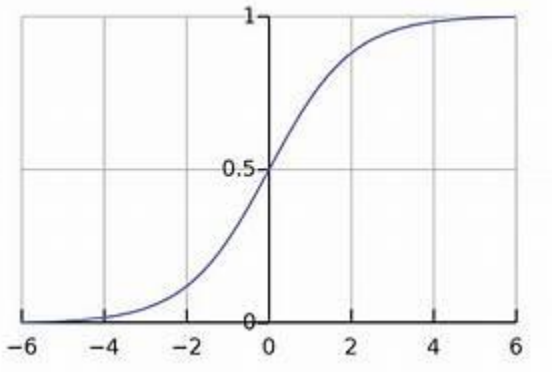


Figure 2: ReLU Activation

Exploring Hardware Implementation of Activation Functions

This section explains popular activation functions (AFs) like sigmoid, Tanh, and ReLU. While they are based on the same idea, each one uses different math processes.

For storing AF parameters locally, you can use different on-chip memory types like Lookup Tables (LUTs), Block RAM (BRAM), Distributed FPGA memory, or external DRAM. Here are some ways to implement them:

1. Using LUTs to store the function.
2. Using LUTs to store parameters.
3. Approximating calculations using base-2.
4. Using the Coordinate Rotation Digital Computer (CORDIC) algorithm.
5. Combinational logic-based implementation of the non-linear and continuous character of the AF makes a correct computation utilizing hardware implementation problematic.

Improving Activation Function Performance with Pipelining

The CORDIC algorithm's hyperbolic rotation mode is used in hardware designs for adjustable activation functions. These systems use less power and space than traditional memory-based devices but have lower throughput. To address this, we present an improved CORDIC-based hardware solution for high-throughput neural network applications with adjustable non-linear activation functions. Here are the main points:

- We analyze the trade-off between accuracy and the two CORDIC phases using Pareto analysis.
- For non-linear activation functions, we determine the optimal number of pipeline stages in a CORDIC-based architecture, considering space and power needs. Our research shows that a four-stage pipeline can increase throughput without losing accuracy.
- We examine the effects of computational approximation, which reduces the number of pipeline stages, by extracting error cost functions for the AF model.
- Using CMOS 45nm technology, we compare our design to state-of-the-art designs, discussing the circuit's physical properties like area, power, and critical delay for the derived Pareto points.

Efficient and Configurable Neuron Design Using CORDIC Architecture

As computational complexity increases in hardware implementations of artificial neural networks (ANNs), both area and performance can suffer. Fixed ASIC designs require extra effort to configure features like bit precision flexibility or different activation functions (e.g., sigmoid,

tanh). High-bit precision calculations (32-bit or 64-bit) also consume more power and space. Therefore, quicker response times and simpler technology (bit accuracy) are highly desired.

The classic ASIC-based neuron design includes several activation functions and a multiply-accumulate (MAC) unit. A MAC unit consists of a multi-adder tree and multiple multipliers. The activation function that processes the MAC unit's output is selected by a multiplexer based on the application.

Improving MAC Unit Performance with Pipelining

A major challenge in using modern Deep Neural Networks (DNNs) is the resource-heavy Multiply-Accumulate (MAC) unit. To make DNN accelerators work better, we need to improve computation efficiency and throughput. This project aims to design a MAC unit using the CORDIC architecture. Although CORDIC-based devices are efficient in size and power, they have low throughput. Our pipelined architecture for the MAC unit addresses this issue. We use Pareto analysis to examine accuracy variations at different precision levels and identify the critical pipeline steps for optimal performance.

Introduction to MAC and Performance-Enhancing Techniques

For high bit-precision calculations, the Multiply-Accumulate (MAC) block in Deep Neural Networks (DNNs) becomes very demanding due to its power-hungry multiplication. Researchers have proposed ways to customize ASIC and FPGA architectures, but these often compromise performance, throughput, and accuracy. Bandwidth constraints also pose challenges when implementing hardware accelerators.

Studies have examined performance metrics for hardware acceleration with various bit precisions (8, 16, 24, or 32 bits). The shift-and-add multiplication method uses fewer hardware resources and reduces complexity. The CORDIC architecture is suggested for efficient MAC design, but it has lower throughput due to its iterative nature. To achieve n-bit accuracy, an n-bit barrel shifter and n-1 additions are needed. Vedic multipliers offer efficient low-precision MAC designs, but they become unscalable for high-precision systems due to increased critical path and propagation time.

Resource-Saving Techniques for MAC Units

Researchers have explored the modified Booth's method for multiplication to save resources. They examined physical performance metrics after implementing a MAC design based on Wallace trees, which use efficient lower-precision AND and OR planes. However, as bit accuracy increases, the design complexity also rises.

Using an approximation multiplier in MAC calculations helps reduce power consumption and circuit latency, which is beneficial for error-tolerant applications. One study introduced a roughly accurate partial product accumulation tree, and the multiplier's error-prone logic compressors showed promising results for hardware implementation, enhancing power and energy efficiency.

Sharing weights simplifies MAC calculations and maintains consistent storage capacity. Research on a reversible logic structure high-performance array multiplier has shown improved throughput performance. However, current enhancement strategies do not cover multi-precision signed/unsigned computing.

This study evaluates a high-performance CORDIC-based MAC unit for DNN accelerators, addressing the trade-offs between area, power, and throughput.

Fixing Throughput Issues with CORDIC Phases

To solve the throughput problem, we use CORDIC phases and assess the necessary pipeline steps, which add some area overhead. Since accuracy and the number of pipeline stages are often at odds, we examined different stages of the CORDIC-based architecture to understand this trade-off. We also tested performance accuracy at various levels of mathematical precision.

Although there's a slight loss in accuracy, 8-bit precision computing is much faster than 16-bit precision and uses four times less memory bandwidth. Additionally, we reduced space usage without sacrificing throughput or accuracy, thanks to the Pareto analysis that evaluated the necessary number of phases.

Using a 45 nm technology node, the next section synthesizes the proposed design and describes the circuit's physical properties, including area, power, and critical delay.

Layer-Multiplexed High-Performance DNN Accelerator

Applying deep neural networks (DNN) to real-world problems requires a lot of memory bandwidth, processing power, and other hardware resources. Efficient DNN engines are essential for AI-powered Internet of Things applications. As the neural network gets deeper, more hardware resources are needed.

Our work proposes a new DNN engine architecture that is more cost-effective and performs better. We developed and deployed a layer-multiplexed DNN accelerator at the system level. By implementing the suggested pipelined MAC design, we overcome the throughput limits that occur when reusing a single layer within the DNN.

IV. Results and Discussion

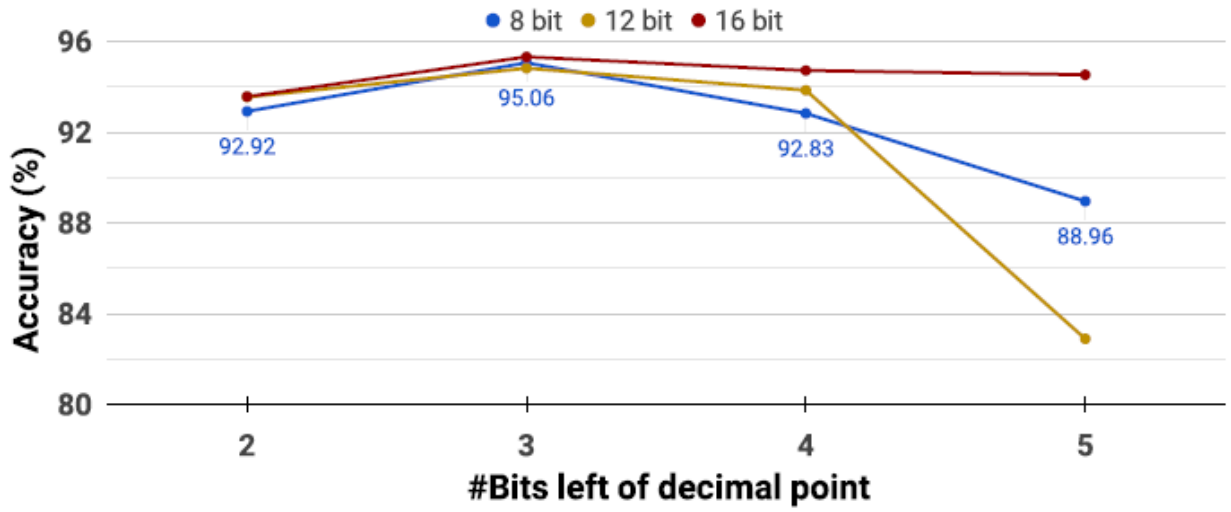
Strength of the Proposed Work

While much work has been done to improve the accuracy of deep neural networks, there has been little effort to design equivalent VLSI architecture for a chip or system solution. Our proposed work aims to create a single chip/system solution that amputated patients can use to perform daily activities easily, making them more self-dependent.

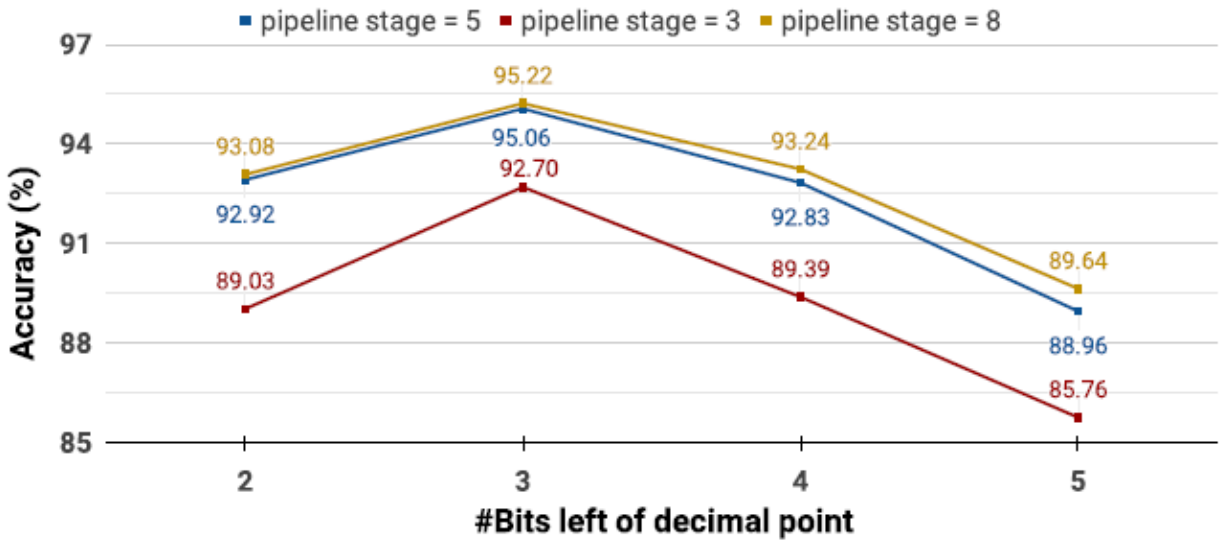
This section discusses the hardware performance and implementation results for a complete layer-reused system architecture and an enhanced CORDIC engine-based MAC unit. The

efficient layer-reused DNN configuration (196:64:32:32:10) with an enhanced MAC unit is designed using a hardware description language. The accuracy performance of the proposed design has been validated using Python with the standard TensorFlow computation for MNIST. Furthermore, we have simulated fixed-point behavior by quantizing all the operations and activation. Hence, our CORDIC-based Python implementation is framework-independent and faithfully replicates the hardware design to evaluate accuracy. The MNIST input image (28x28) is resized to 14x14, which increases our DNN performance, and the model has been trained for signed 8-bit precision fixed-point arithmetic, noting the inference accuracy. The extracted weights and biases are used to verify the hardware implementation. The Virtex-7 VC707 FPGA has targeted DNN implementation and evaluated results. The following experimental evaluation is performed for design validation and extracted results.

To improve the performance of the MAC unit, we optimized the architecture by introducing pipelining. The number of pipeline stages directly affects resource utilization, so we analyzed this through experiments. We used a five-stage pipeline for performance evaluation. Based on observed Pareto points, we implemented the enhanced MAC design and compared the results with state-of-the-art designs. In our experiment, we first compared performance parameters for signed 8-bit precision with the best designs available, as shown in Table 1. This table reports resource utilization, circuit critical delay per clock, and Power-Delay-Product (PDP). We used signal and logic power for PDP calculation. It can be seen that our proposed design has a 2.2 times lower PDP compared to



(a) Analyzing the effect of the number of integer bits for different bit-precisions, at Max Norm = 5.5 for the proposed enhanced MAC.



(b) Analyzing the accuracy versus the number of bits left of the decimal point for 8, 12, and 16 bit-precision, with Max Norm = 5.5 for the proposed enhanced MAC.

Figure 3: Analyzing the accuracy versus the number of bits

Secondly, we analyzed the impact of bit precision on hardware parameters. We compared our MAC implementation results with the multiply and accumulate IP from Xilinx, as shown in Table 2. We looked at different signed dynamic fixed-point representations using 6-, 9-, 13-, and 17-bit, which correspond to 5-, 8-, 12-, and 16-bit precision, respectively. One extra bit is used as a signed bit in the actual computation, as discussed in Section 5.4. Since our MAC uses five pipeline stages, the minimum bit precision is five magnitude bits. We observed that the proposed design shows only slight increases in resource utilization for higher precision. The rate of increase in flip-flop (FF) utilization is almost the same for both MACs. However, because our MAC has a pipelined structure, it uses more FFs, but the difference increases more or less linearly.

Table 1: compares resource utilization and performance parameters of the MAC with state-of-the-art techniques at fixed <6,8> precision.

Resources Utilization	**Slice LUTs**	**Slice Registers**	**Critical Path Delay (ns)**	**Power-delay Product (pJ)**
Vedic	159	245	4.48	5.86
IEEE	130	45	3.98	5.01
Wallace	105	112	2.59	3.13
Booth	83	61	3.08	2.77
Shift-add	75	58	5.44	3.97
Acc_App	62	59	2.87	2.04
proposed	54	88	1.52	0.91

Table 2: compares resource utilization and performance parameters for the MAC architecture at fixed <8, 6> precision.

Bit-precision signed	Slice LUTs	Slice Registers	DSP Util.	Delay (ns)		Power (mW)			PDP (pJ)
				Logic	Signal	Logic	Signal	Dynamic	
16-bit	369	76	-	2.268	6.783	1.95	1.95	13	117.7
	95	162	-	0.805	1,319	0.55	0.77	11	23.7
12-bit	244	60	-	0.966	3.485	1.26	1.26	9	40.1
	75	126	-	0.731	1.108	0.32	0.48	8	14.7
8-bit	130	44	-	0.921	2.895	0.66	0.60	6	22.9
	54	88	-	0.755	0.768	0.24	0.36	6	9.1
5-bit	53	28	-	0.813	2.277	0.27	0.21	3	9.37
	35	58	-	0.713	0.692	0l.16	0.16	4	5.62

Conclusion:

This paper presents efficient techniques for layer-multiplexed artificial neural network (ANN) engines that reduce hardware resources and improve performance. It addresses the low throughput of conventional multiplexed techniques by proposing an enhanced neuron architecture using pipelined CORDIC stages, optimizing the CORDIC architecture for area efficiency and reduced critical delay. The ANN engine, written in HDL, is implementable on both ASIC and FPGA, allowing users to configure deeper ANNs for object classification with user-defined parameters. Detailed analysis for selecting adaptable circuit design parameters is provided. The proposed techniques open new opportunities for low-area, low-power, and high-performance designs, especially in Edge-AI applications. The Virtex-7 FPGA platform was used to implement and verify the architecture for MNIST classification, showing improved area, power, and throughput performance. The project code will be made open-source to support reproducible results and hardware implementation for both ASIC and FPGA.

References:

1. A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, ‘‘A survey of quantization methods for efficient neural network inference,’’ in Low-Power Computer Vision: Improve the Efficiency of Artificial Intelligence. New York, NY, USA: Chapman & Hall, 2022, ch. 1.2.12, pp. 14–17.
2. L. Urbinati and M. R. Casu, ‘‘A reconfigurable 2D-convolution accelerator for DNNs quantized with mixed-precision,’’ in Proc. Appl. Electron. Pervading Ind., Environ. Soc. (ApplePies), Genoa, Italy, 2023, pp. 210–215.
3. McCarthy, John. "Artificial intelligence, logic, and formalising common sense." *Machine Learning and the City: Applications in Architecture and Urban Design* (2022): 69-90.
4. T. Hotfilter, J. Hoefer, P. Merz, F. Kreß, F. Kempf, T. Harbaum, and J. Becker, ‘‘Leveraging mixed-precision CNN inference for increased robustness and energy

efficiency,'' in Proc. IEEE 36th Int. System-on-Chip Conf. (SOCC), Santa Clara, CA, USA, Sep. 2023, pp. 1–6.

5. P. Dhilleswararao, S. Boppu, M. S. Manikandan, and L. R. Cenkeramaddi, "Efficient hardware architectures for Accelerating Deep Neural Networks: Survey," IEEE Access, vol. 10, pp. 131788–131828, 2022. doi:10.1109/access.2022.3229767
6. J. S. Walther, "The story of unified CORDIC," Journal of VLSI Signal Processing, vol. 25, no. 2, pp. 107–112, 2000.
7. A. Burrello, A. Garofalo, N. Bruschi, G. Tagliavini, D. Rossi, and F. Conti, ''DORY: Automatic end-to-end deployment of real-world DNNs on lowcost IoT MCUs,'' IEEE Trans. Comput., vol. 70, no. 8, pp. 1253–1268, Aug. 2021.
8. L. Mei, P. Houshmand, V. Jain, S. Giraldo, and M. Verhelst, "ZigZag: Enlarging joint architecture-mapping design space exploration for DNN accelerators," IEEE TC, vol. 70, no. 8, pp. 1160–1174, 2021.
9. Turing AM (1950) Computing machinery and intelligence. Mind 59(236):433–460
10. Bowen, J.P. (2024). Alan Turing: Breaking the Code, Computing, and Machine Intelligence. In: Giannini, T., Bowen, J.P. (eds) The Arts and Computational Culture: Real and Virtual Worlds. Springer Series on Cultural Computing. Springer, Cham.
11. McCarthy J, Minsky ML, Rochester N, Shannon CE. A proposal for the Dartmouth summer research project on artificial intelligence: August 31, 1955. AI Mag 2006;27(4):12.
12. Stahl BC (2021) Ethical issues of AI. In: Artificial Intelligence for a Better Future. Springer Briefs in Research and Innovation Governance.
13. Sijia Lu and Feng Xu. Linear leaky-integrate-and-fire neuron model based spiking neural networks and its mapping relationship to deep neural networks. Frontiers in neuroscience, page 1368, 2022.
14. Huanhao Li, Zhipeng Yu, Qi Zhao, Tianting Zhong, and Puxiang Lai, "Accelerating deep learning with high energy efficiency: from microchip to physical systems", The Innovation 3(4): 100252 (2022)
15. Cheng, Yuan, et al. "A Low-Power High-Throughput In-Memory CMOS-ReRAM Accelerator for Large-Scale Deep Residual Neural Networks." *2019 IEEE 13th International Conference on ASIC (ASICON)*. IEEE, 2019.